



École Nationale Supérieure d'Informatique et d'Analyse des Systèmes  
Centre d'Études Doctorales en Sciences des Technologies de l'Information et de l'Ingénieur

## THÈSE DE DOCTORAT

---

# MARKOV PROCESSES FOR PARTICLE SWARM OPTIMIZATION CONTROL AND STOCHASTIC MODELING: APPLICATION TO AIR TRANSPORT PROBLEMS

---

Présentée par

**Oussama AOUN**

Le 27/03/2019

**Formation doctorale : Informatique**  
**Structure de recherche : Smart Systems Laboratory (SSL)**

### JURY

**Professeur Bouchaib BOUNABAT**

PES, ENSIAS, Université Mohammed V, Rabat

**Professeur Abdellatif EL AFIA**

PES, ENSIAS, Université Mohammed V, Rabat

**Professeur Hassan QJIDAA**

PES, FSDM, Université Sidi Mohamed Ben Abdellah, Fès

**Professeur Rachid ELLAIA**

PES, EMI, Université Mohammed V, Rabat

**Professeur Raddouane CHIHEB**

PES, ENSIAS, Université Mohammed V de Rabat

**Professeur Mohamed ETTAOUIL**

PES, FST, Université Sidi Mohamed Ben Abdellah, Fès

**Professeur El-Ghazali TALBI**

Professeur, Université de Lille, France

**Président**

**Directeur de thèse**

**Rapporteur**

**Rapporteur**

**Rapporteur**

**Examineur**

**Examineur**

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

*In Memory of my Father*

*To my Mother*

*With Love and Eternal Appreciation*

---

# RÉSUMÉ

Ces dernières années, les modèles de Markov ont couvert plusieurs applications importantes dans la vie réelle en tant que modèles probabilistes basés sur l'apprentissage automatique. En théorie, le modèle stochastique de Markov se réfère simplement à représenter des systèmes qui changent de façon aléatoire où l'état suivant dépend uniquement de l'état actuel. En pratique, la propriété de Markov est de plus en plus utilisée dans l'analyse de la dynamique des processus stochastiques et de nombreuses variantes ont été proposées. Dans cette thèse, les modèles de Markov seront utilisés dans trois types d'application différents pour représenter la diversité et le bénéfice précieux de l'application de cet apprentissage automatique probabiliste.

D'abord, le modèle de Markov caché (MMC) est utilisé pour constituer un modèle générique pour l'amélioration de la performance de l'algorithme d'optimisation par essaim de particules (OEP). En effet, l'analyse du OEP en tant qu'algorithme méta-heuristique basé sur une population stochastique et selon son comportement stochastique conduit à la construction d'une chaîne de Markov sur les réalisations du OEP. Ainsi, nous considérons différentes sélections optimales de paramètres selon le changement d'états de Markov du OEP, et ce en fonction de la classification donnée par le modèle MMC. Nous proposons ensuite une nouvelle méthode permettant d'optimiser automatiquement les performances de l'algorithme OEP. L'algorithme OEP est amélioré par la configuration hors ligne et en ligne de ses paramètres à l'aide de MMC. Tout d'abord, une méthode est conçue spécifiquement pour des classes particulières d'instances de problèmes, ce qui produit de meilleures performances dans des applications dans le monde réel. Aussi, il sera conçu pour les mécanismes de contrôle en ligne de l'algorithme OEP qui adaptent les paramétrages dans l'exécution. Le cas des configurations OEP homogènes et hétérogènes sont considérées. L'analyse empirique d'un ensemble de plusieurs fonctions de référence montre des performances remarquables par rapport à d'autres variantes de OEP dans la littérature.

Deuxièmement, l'apprentissage par renforcement est analysé et discuté. L'utilisation du formalisme de processus décisionnel de Markov (PDM) est apparue dans des applications importantes de prise de décision dans divers domaines, il est utilisé pour aider à prendre des décisions dans un environnement stochastique. Cependant, les méthodes classiques de résolution des PDMs surdimensionnés souffrent du problème de dimensionnalité de Bellman et du manque d'informations dans le modèle. Par conséquent, une nouvelle méthode de résolution est proposée dans le contexte des algorithmes d'apprentissage par renforcement en se basant sur un libre-modèle. En fonction de l'algorithme OEP amélioré, nous établissons des règles de coopération entre les apprenants en renforcement indépendants afin d'accélérer la convergence vers des solutions optimales. La résolution du processus d'apprentissage prend en compte deux étapes : l'apprentissage indépendant par un algorithme Q-learning et la stratégie de partage des valeurs de Q par un algorithme OEP coopératif. Ce Q-learning coopératif basé sur OEP donne les meilleurs résultats ; cela est dû à l'aspect coopératif de OEP pour améliorer la recherche de la valeur Q.

Enfin, le transport aérien est choisi comme domaine d'application dans cette thèse en tant qu'environnement opérationnel dynamique et complexe. Le modèle de processus de décision de Markov est également appliqué dans ce contexte. Ce modèle stochastique est étudié dans différentes compagnies aériennes qui rencontrent des problèmes de gestion et d'affectation des ressources des aéroports et des compagnies aériennes de manière efficace et efficiente. Notre approche proposée traite les éventuelles perturbations dans le transport aérien. La solution fournie des décisions qui pourraient être prises au moment de l'horizon de planification des opérations de vol. Ce type de modèle prend en compte les perturbations stochastiques et permet de gérer les risques potentiels en tenant compte les éventuelles incertitudes.

**Mots clés :** Intelligence par essaim ; Optimisation par essais de particules ; Gestion de risque ; Apprentissage par renforcement ; Optimisation stochastique ; Apprentissage probabiliste ; Modèle de Markov caché ; Processus décisionnel de Markov ; Transport aérien.

---

# ABSTRACT

In recent years, Markov models have found important applications in real life as a probabilistic machine-learning-based model. In theory, Markov stochastic model simply refers to represent randomly changing systems where the next state depends only on the current state. In practice, the Markov property is increasingly being used in the analysis of the dynamics of stochastic processes and many variants have been proposed. In this thesis, Markov models will be used in three different application types to represent the diversity and the valuable profit of using this probabilistic machine learning.

First, a Hidden Markov model (HMM) is used to constitute a generic model integrated as a performance enhancement of the particle swarm optimization algorithm (PSO). In fact, the analysis of the PSO as a stochastic population-based metaheuristic algorithm, and according to its stochastic behavior, leads to the construct of a Markov chain on PSO achievements. So, we consider a different optimal selection of parameters during the changed Markov states of PSO based on the classification given by the HMM model. Then, we propose a new method that allows optimizing the performance of the PSO algorithm automatically. PSO algorithm is improved by both offline and online configurations of its parameters using HMM. Firstly, a method is designed specifically for particular classes of problem instances, which produces better performances in real-world applications. Secondly, it will be designed for online PSO algorithm control mechanisms that adapt parameter settings within the execution. The case of homogeneous and heterogeneous PSO configurations are both considered. Empirical analysis of a set of several benchmark functions shows remarkable performances in comparison with other PSO variants from literature.

Second, Reinforcement learning is analyzed and discussed. The use of Markov decision process (MDP) formalism has found important applications to decision making in various domains, where it is used to help to make decisions on a stochastic environment. However, classical methods for solving large MDPs suffer from Bellman's curse of dimensionality, and lack of model information. Therefore, a new resolution method is proposed in the context of model-free-based reinforcement learning algorithms. Depending on the enhanced PSO algorithm, we are building cooperation rules between independent reinforcement learners to accelerate convergence to optimal solutions. Two stages are considered for resolution in the learning process: independent learning by Q-learning algorithm and Q-values sharing strategy by a cooperative PSO algorithm. This PSO-based cooperative Q-learning gives the best results; this is due to the cooperative aspect of PSO to enhance the Q-value search.

Finally, airline transport is chosen as an application field in this thesis as a challenging dynamic operational environment. Markov decision process model is also applied in this context. This stochastic model is investigated in different airline problems dealing with managing and allocating airport and airline resources effectively and efficiently. Our proposed approach is dealing with disturbances in airline transport. The provided solution to the decisions that could be made at the time of the planning horizon of flights operations. This kind of model takes into account stochastic disturbances, and it can manage the potentially risks by handling possible uncertainties.

**Keywords:** Swarm intelligence; Particle Swarm Optimization; Risk management; Reinforcement learning; Stochastic optimization; Probabilistic learning; Hidden Markov Model; Markov decision process; Airline transport.

---

# ACKNOWLEDGEMENTS

Praise be to Allah who gave me health, strength and patience to conduct this work.

Acknowledgment is a thanks to the many people who directly or indirectly supported me in this thesis. I apologize to all who are not mentioned in this section but should be!

Thanks to my parents who helped and supported me during all the years, always ready to give and ask for nothing.

To my beloved father's soul **EL MOSTAFA AOUN** who died during the preparation of this thesis. He taught me the meaning of life.

To my mother for her unlimited support, love and patience.

To my two sisters, for their continuous support.

To my uncle MOSTAFA who helped me in every possible way, for which I am eternally grateful.

Special thanks to my supervisor **Dr. Abdellatif El Afia** for his consistent supervision and patience, invaluable advice, and guidance throughout the course of this thesis.

I am deeply grateful to all members of the jury for agreeing to read the manuscript and to participate in the defense of this thesis.

Rabat, December 2018

---

# CONTENTS

<b>LIST OF FIGURES .....</b>	<b>12</b>
<b>LIST OF TABLES .....</b>	<b>14</b>
<b>LIST OF ALGORITHMS .....</b>	<b>15</b>
<b>INTRODUCTION .....</b>	<b>16</b>
<b>CHAPTER I : PARAMETERS SETTING OF HOMOGENEOUS PARTICLE SWARM OPTIMIZATION .....</b>	<b>19</b>
<b>1. Background .....</b>	<b>20</b>
1.1 Nature-inspired algorithms .....	20
1.2 Swarm intelligence .....	21
<b>2. Particle Swarm Optimization Definition .....</b>	<b>23</b>
2.1 PSO Basic algorithm .....	23
2.2 The Parameters of PSO .....	25
2.2.1 Number of iterations .....	26
2.2.2 Swarm size .....	26
2.2.3 Acceleration coefficients: .....	27
2.2.4 Neighborhood Topologies .....	28
2.3 Convergence Analysis of PSO .....	29
2.4 Advantages and Drawbacks .....	30
2.5 PSO improvements .....	31
<b>3. Improved PSO variants from literature.....</b>	<b>32</b>
3.1 Population size .....	33
3.2 PSO topology.....	34
3.3 Inertia weight .....	35
3.4 Acceleration coefficients.....	36
<b>4. PSO parameters setting using HMM.....</b>	<b>37</b>
4.1 The Hidden Markov Model .....	37
4.1.1 Model definition .....	37
4.1.2 Hidden Markov Model Learning evaluation .....	38
4.2 Parameter Tuning (Offline) .....	39
4.2.1 HMM-based tuner Model for PSO.....	40
a. Generation phase .....	41
b. Test phase.....	41
c. Performance evaluation phase.....	42
4.2.2 Performance evaluation of HMM-based tuner .....	43
4.3 Automated parameter control by HMM.....	44
4.3.1 A generic model.....	44
a. Markov Chain on Particle Swarm Optimizer .....	44
b. A model for adaptive parameters .....	45
c. PSO hidden states.....	46
d. Model of state classification.....	47
e. Online parameters estimation.....	50
4.3.2 Parameter control of Homogeneous PSO .....	51
a. Control of acceleration coefficients .....	52
a.1 Coefficients update by state .....	52
a.2 HMM adaptation of acceleration factors (HMM-APSO).....	53

b.	Control of Population Size .....	54
b.1	Population update strategy by state .....	54
b.2	Generation and elimination of particles .....	56
b.3	HMM control of Population size (HMM-PPSO) .....	57
c.	Control of inertia weight .....	58
c.1	Inertia weight control by state .....	59
c.2	HMM control of inertia (HMM-wPSO) .....	59
d.	Empirical evaluation .....	60
d.1	Parameters setting .....	60
d.2	Examination of HMM-wPSO with other PSO variants .....	63
d.3	Examination of HMM-PPSO with other PSO variants .....	67
d.4	Examination of HMM-APSO with other PSO variants .....	71
d.5	Comparison of the HMM-based approaches .....	74
<b>5.</b>	<b>Conclusion .....</b>	<b>78</b>
<b>CHAPTER II: ONLINE PARAMETERS CONTROL OF HETEROGENEOUS PARTICLE SWARM OPTIMIZATION.....</b>		<b>80</b>
<b>1.</b>	<b>Improved PSO variants from literature.....</b>	<b>81</b>
<b>2.</b>	<b>parameter control of heterogeneous PSO .....</b>	<b>82</b>
2.1	Self-state identification for PSO .....	83
2.2	Cooperative Multi-swarm .....	86
2.2.1	Sub-swarms constitution .....	86
2.2.2	Multi-swarms cooperation .....	87
a.	Master/Slave scheme .....	87
b.	Adaptive cooperation .....	88
2.3	Experimentation .....	92
2.3.1	Performance evaluation .....	92
2.3.2	Convergence speed .....	93
2.3.3	Statistical tests.....	94
<b>3.</b>	<b>Conclusion .....</b>	<b>95</b>
<b>CHAPTER III: MULTI-AGENT REINFORCEMENT LEARNING BY HMM-BASED PSO .....</b>		<b>96</b>
<b>1.</b>	<b>Intelligent Agents .....</b>	<b>97</b>
1.1	Agent Systems .....	98
1.1.1	Single-agent Systems.....	98
1.1.2	Multi-agent Systems .....	98
1.2	Multi-agent Systems Classifications .....	99
1.3	Multi-agent Learning.....	99
1.3.1	Cooperative Multi-agent Learning Methods.....	99
1.3.2	Team Learning .....	100
a.	Homogeneous Team Learning .....	100
b.	Heterogeneous Team Learning.....	100
c.	Concurrent Learning .....	100
<b>2.</b>	<b>Markov decision process .....</b>	<b>101</b>
2.1	Definition of MDP .....	101
2.2	Finite-horizon MDP vs Infinite-horizon MDP .....	102
2.3	BELLMAN'S EQUATION .....	103
2.4	Solution Methods .....	103
2.4.1	Policy Iteration.....	104
2.4.2	Value Iteration.....	105
2.4.3	Other Methods .....	106
2.5	MDP variants.....	106
2.5.1	Time-Dependent Markov Decision Processes .....	106
2.5.2	Multi-Agent Markov Decision Processes .....	108



2.5.3	Time-Dependent Multi-Agent Markov Decision Processes .....	110
2.6	Limits of Markov Decision Processes .....	111
2.6.1	Curse of dimensionality .....	111
2.6.2	Memory Requirement .....	111
2.6.3	Stationary Assumption .....	112
2.6.4	Large Markov Decision Processes .....	112
<b>3.</b>	<b>Reinforcement Learning fundamentals .....</b>	<b>112</b>
3.1	RL basic model .....	112
3.2	Policy types .....	114
3.3	Action Selection Policies .....	114
3.4	Exploration/Exploitation paradigm.....	114
3.5	Q-learning Algorithm.....	115
3.6	Multi-agent Reinforcement Learning .....	116
3.6.1	Benefits of Multi-agent Reinforcement Learning.....	116
a.	Parallel Computation .....	117
b.	Sharing of Knowledge.....	117
c.	Robustness .....	117
3.6.2	Recent developments in Multi-agent Reinforcement Learning .....	117
a.	Combinational Reinforcement Learning .....	117
b.	Swarm Reinforcement Learning.....	118
<b>4.</b>	<b>PSO Cooperative reinforcement learning based HMM .....</b>	<b>119</b>
4.1	Q-value HMM Sharing swarm Strategies.....	119
4.2	HMM-QPSO algorithm .....	120
4.3	Experimentation .....	122
4.3.1	Experience settings.....	122
4.3.2	Experimental Results .....	123
<b>5.</b>	<b>Conclusion .....</b>	<b>124</b>
<b>CHAPTER IV: STOCHASTIC MODEL OF THE GATE ASSIGNMENT PROBLEM .....</b>		<b>126</b>
<b>1.</b>	<b>The gate assignment Problem .....</b>	<b>127</b>
1.1	Problematic .....	127
1.2	Problem statement .....	128
1.2.1	Airports Description .....	129
1.2.2	Terminal description and configurations.....	130
1.2.3	Example of Mohammed V airport of Casablanca .....	132
<b>2.</b>	<b>Literature review.....</b>	<b>134</b>
2.1	Constraints and Objectives .....	134
2.1.1	Constraints .....	135
2.1.2	Objectives.....	136
2.2	Models of the GAP.....	137
2.2.1	Basic Mathematical Programming model .....	138
2.2.2	Uncertainty managing Models of the GAP .....	139
a.	Simulation Approach to GAP .....	140
b.	Fuzzy Logic Application.....	140
c.	Expert Systems for the GAP.....	141
d.	The Gate Re-assignment Problem .....	141
2.2.3	Datasets.....	142
<b>3.</b>	<b>Stochastic Proposed Approach of the GAP .....</b>	<b>142</b>
3.1	MDP Model .....	142
3.1.1	Aircraft size constraints model.....	143
3.1.2	Airport Configuration .....	144
3.1.3	MDP parameters .....	145
3.2	Multi-agent MDP Formalism .....	146

3.2.1	Multi-agent control mechanism .....	147
3.2.2	Multi-agent MDP model .....	148
3.3	Time-dependent Multi-agent GAP formulation .....	151
3.4	Experiment .....	153
3.4.1	Single Agent model experiment .....	153
3.4.2	Multi-Agent Model experiment .....	155
3.4.3	Time-Dependent Multi-Agent Model experiment .....	157
<b>4.</b>	<b>Conclusion .....</b>	<b>159</b>
<b>CHAPTER V: STOCHASTIC MODEL OF THE CREW PAIRING PROBLEM AND A TUNED PSO SOLUTION .....</b>		<b>160</b>
<b>1.</b>	<b>The Crew Pairing Problem .....</b>	<b>161</b>
1.1.	Problematic .....	161
1.2.	Airline resource planning process .....	162
1.3.	Crew Pairing .....	164
1.3.1	PROBLEM DESCRIPTION .....	164
1.3.2	Crew restrictions .....	165
1.3.3	Pairing Generation .....	165
1.3.4	Crew Pairing Optimization .....	166
1.4.	Robust Scheduling .....	167
<b>2.</b>	<b>Literature review .....</b>	<b>168</b>
2.1	Works from literature .....	169
2.2	Basic Models of literature: .....	170
2.2.1	Deterministic model .....	171
2.2.2	Stochastic model .....	171
<b>3.</b>	<b>Tuning resolution Method by PSO .....</b>	<b>172</b>
3.1	Experimental Model and tuning algorithm .....	172
3.1.1	The crew scheduling problem: Crew pairing .....	173
3.1.2	Binary particle swarm optimization .....	173
3.2	Generation of instances and configurations of crew pairing .....	173
3.3	Tuning Test phase .....	174
3.4	Tuning Evaluation phase .....	175
<b>4.</b>	<b>Markov decision Process Model of crew pairing .....</b>	<b>177</b>
4.1	Why MDP Model for the crew pairing .....	177
4.2	The MDP Model .....	177
4.3	Experimentation .....	179
<b>5.</b>	<b>Conclusion .....</b>	<b>180</b>
<b>CHAPTER VI: A REINFORCEMENT LEARNING MODEL OF THE MAINTENANCE ROOTING PROBLEM .....</b>		<b>182</b>
<b>1.</b>	<b>Aircraft maintenance Problem .....</b>	<b>183</b>
1.1.	Problematic .....	184
1.2.	Airline Planning Process of maintenance .....	184
1.3.	Maintenance Problem description .....	185
1.4.	Maintenance types .....	186
1.5.	Resources management in maintenance .....	187
1.6.	Maintenance tasks .....	188
<b>2.</b>	<b>Maintenance Literature .....</b>	<b>189</b>
2.1	General maintenance rooting model .....	189
2.2	Aircraft maintenance studies .....	191
<b>3.</b>	<b>Markov decision process Model .....</b>	<b>193</b>
3.1	Flight delays propagation .....	193
3.2	Delays detection from data .....	194
3.3	MDP model data based .....	195

3.4	Reinforcement learning model based data.....	196
3.5	Experimentation.....	198
<b>4.</b>	<b>CONCLUSIONS.....</b>	<b>199</b>
	<b><i>Conclusion</i> .....</b>	<b>201</b>
	<b><i>References</i>.....</b>	<b>203</b>
	<b><i>Appendix: Thesis publications</i> .....</b>	<b>219</b>

---

# LIST OF FIGURES

Figure 1	Flow process of Nature algorithms design .....	20
Figure 2	Different observed swarms .....	22
Figure 3	particle position update .....	24
Figure 4	Effect of population size increase on HMM-APSO [18].....	27
Figure 5	Common neighborhood topologies of PSO. ....	28
Figure 6	ring topology (left) and star topology (right) .....	29
Figure 7	Stochastic Particle Trajectory for $w = 0.9$ and $c_1 = c_2 = 2$ .....	29
Figure 8	PSO improvement methods .....	31
Figure 9	The graphical model for a hidden Markov model.....	37
Figure 10	Description of the HMM-based tuner.....	41
Figure 11	Markov Chain of PSO states.....	46
Figure 12	Fuzzy functions of evolutionary factor by particles state .....	48
Figure 13	HMM classifications .....	51
Figure 14	Comparison on convergence speed on fitness functions .....	66
Figure 15	Performance comparison with different functions .....	70
Figure 16	Comparison on Benchmark functions. ....	73
Figure 17	HMM based approaches comparison of convergence speed .....	77
Figure 18	Sub-swarms and possible particle movements .....	86
Figure 19	Sub-swarms and the master/slave interactions.....	87
Figure 20	Sub-swarms and its cooperation diagram.....	89
Figure 21	MsAHMM-PSO Convergence speed comparaison .....	93
Figure 22	Markov Decision Process.....	101
Figure 23	Elementary example of TMDP.....	107
Figure 24	Representation of probability density function types .....	107
Figure 25	Centralized control in MMDP.....	109
Figure 26	TMMDP policy representation .....	111
Figure 27	: Reinforcement learning model . ....	113
Figure 28	The AMRLS aggregation design .....	118
Figure 29	Example of grid world for $n=10$ .....	123
Figure 30	Comparison of Reinforcement learning algorithms .....	124
Figure 31	Example of gating at Mohamed V airport .....	127
Figure 32	Area of interest of the gate assignment problem. ....	129
Figure 33	Schematic presentation of an airport .....	130
Figure 34	Airside ground resources optimization at an airport.....	131
Figure 35	Terminal configurations [181].....	132
Figure 36	Mohammed V Airport Infrastructure [ONDA].....	133
Figure 37	Mohammed V configuration.....	133
Figure 38	Example of gate disposition .....	143
Figure 39	Graph representation of gates.....	143

Figure 40 Centralized control in MMDP .....	147
Figure 41 Agents representation .....	149
Figure 42 Agents distribution and temporal planning .....	152
Figure 43 Initial policy .....	155
Figure 44 given solution .....	155
Figure 45 Transitions and rewards matrixes .....	156
Figure 46 State transition diagram.....	158
Figure 47 probability density functions of $\mu_2$ .....	158
Figure 48 The resource planning process in airlines .....	162
Figure 49 Example of flight tree for a flight .....	166
Figure 50 Robustness in air scheduling .....	168
Figure 51 Comparison of default and tuned configurations by cost and runtime (instance 7).....	176
Figure 52 typical airline planning scheme .....	185
Figure 53 Maintenance types .....	187
Figure 54 Maintenance Resources management.....	188
Figure 55 Maintenance tasks.....	189
Figure 56 Delay propagation .....	193
Figure 57 Reinforcement learning iterating from data. ....	197

---

# LIST OF TABLES

Table 1 Description of Benchmark functions .....	61
Table 2 PSO variants from literature .....	62
Table 3 Results comparisons with other variants of PSO .....	64
Table 4 T-test comparison .....	67
Table 5 Comparison of results .....	68
Table 6 Statistical tests .....	71
Table 7 Results comparisons with the variants of PSO.....	72
Table 8 Statistical tests .....	74
Table 9 Results comparison .....	75
Table 10 Statistical tests .....	78
Table 11 Results comparison .....	92
Table 12 T-test comparison .....	94
Table 13 Data from Hong Kong international airport .....	153
Table 14 Matrix of $E(p(s,s'))$ for numerical example .....	154
Table 15 Stochastic Matrix of $\psi s, s'$ .....	154
Table 16 Initial policy without disruption .....	156
Table 17 Conflicting assignment in initial policy due to delay .....	156
Table 18 Optimal policy .....	157
Table 19 Instances of the crew problem .....	174
Table 20 Configuration parameters .....	174
Table 21 Example of one test output .....	175
Table 22 Representation of probabilities for each metric .....	176
Table 23 FLIGHT SCHEDULE.....	179
Table 24 Legal pairings.....	180
Table 25 Flights data sample.....	198
Table 26 Propagated delay diminution.....	199

---

# LIST OF ALGORITHMS

Algorithm 1: Basic PSO algorithm .....	24
Algorithm 2: Tuning Test .....	41
Algorithm 3 : Baum-welch algorithm.....	48
Algorithm 4: Viterbi algorithm .....	49
Algorithm 5 : Expectations-Maximization algorithm .....	50
Algorithm 6: Update by state of acceleration coefficients.....	51
Algorithm 7: HMM-APSO .....	53
Algorithm 8: Population size control.....	54
Algorithm 9: Generation and elimination strategy.....	55
Algorithm 10: HMM-PPSO .....	57
Algorithm 11: Adaptive inertia weight control.....	58
Algorithm 12: HMM-wPSO .....	59
Algorithm 13: self-adaptive inertia weight control by particle.....	83
Algorithm 14: SelfHMM-APSO algorithm.....	84
Algorithm 15: SelfHMM-wPSO .....	84
Algorithm 16: MsHMM-PSO .....	87
Algorithm 17: Adaptive acceleration update for swarms.....	89
Algorithm 18: Adaptive inertia weight control by swarm.....	90
Algorithm 19: MsAHMM-PSO .....	90
Algorithm 20 : The Classical Policy Iteration Algorithm.....	104
Algorithm 21 : Value Iteration Algorithm.....	105
Algorithm 22: Q-learning .....	115
Algorithm 23: HMM-QPSO algorithm.....	121
Algorithm 24 : Algorithm for computing probabilities.....	145
Algorithm 25: Delays extraction from DATA.....	194



---

# INTRODUCTION

---

Swarm intelligence has emerged from the behavior of a group of social animals or insects that provide insight into metaheuristics [1]. The Swarm movements is a key hypothesis in the production of the particle swarm optimization algorithm (PSO). It was first introduced by [2], where, the exchange of information between individuals of the same varieties supplies an evolutionary advantage. The PSO is a well-known bioinspired algorithm applied to optimization problems, which basically involves a machine-learning technique inspired by birds freely flocking in search of food. More specifically, it contains a number of particles that collectively move in the search space looking for the global optimum.

The main advantage of working with the PSO is its simplicity: its concept and capability to be implemented in a small number of lines of code. Moreover, PSO additionally possesses a short-term memory, which allows the particles to move through the local best and the global best positions. Other choices, just like genetic algorithms, tend to be complex and, most of the time, they do not take into account the past iteration or even the collective emergent performance. As an illustration, in the genetic algorithm, if a chromosome is not picked, the information covered by that chromosome is definitely lost. In spite of its good facilities, a common problem with the PSO, much like other optimization algorithms that are not exhaustive methods, for example, the brute-force search [3], is that of getting stuck in a local optimum, or suboptimal solution, in a way that it could work well on one problem but yet fail on another problem.

Recently, several distinct approaches have appeared to be able to solve much larger optimization problems effectively. Multiple heuristic techniques have been applied in a parametric setting before the PSO, such as Genetic Algorithms [4], Ant Colony Optimization [5], and Tabu Search. In these approaches, there is a selection of parameters that require to be tuned to have an efficient enhanced algorithm. A number of these parameters have been studied earlier from diverse aspects. Nevertheless, such parameters analyses have to be taken into consideration to set up an effective PSO tuned model. The primary intent behind parameters setting approaches of PSO is to incite complex global behaviors through local communications by sharing information between diverse agents and enhancing the learning capacity. Further, it may enable the swarm to adapt to unexpected variations (such as in the dynamic optimization) when they are interacting with more agents. Based on the theoretical results of [6] and [7], the further extension carried out in this thesis, is the analysis of the relationship between transition probability and parameter setting by the estimation of the



transition probability with the selection of the optimum parameter set in each state of PSO based Markov chain. An efficient instrument could be making use of computer simulations. This approach can maximize the value of adaptive parameters selection rather than classical approaches which are based on the iteration number.

Markov models have many variants from literature. Each one has a specific utilization context and applications. Markov Decision Processes provide a mathematical framework for decision making. MDPs are one tool of artificial intelligence (AI) that can be used to get optimal action policies under a stochastic domain. They are widely used for the solving of real-world problems of both planning and control as they are surprisingly capable of capturing the essence of purposeful activity in a variety of situations. For those reasons, they have formed the basis on which many important studies in the field of learning, planning, and optimization have been built. As a result, several different techniques have been developed for their solution. As a popular framework for designing agents that interact with their environment, the execution of actions gives some feedback signals, which indicate how good the actions are. This learning process enables to solve a specific task through these repeated interactions. Different solution methods are presented in the literature, as well as different action selection strategies, which can be used by the agents during the learning process. The idea is to analyze its weaknesses and show ways to fix them by Reinforcement learning techniques to cover an interesting class of real-world scenarios. One way could be by improving cooperation between independent reinforcement learners to a best cooperative Q-learning algorithms based on the cooperation performances of the PSO algorithm. In this case, particles can constitute cooperative learners based on their speed of convergence to an optimal solution.

As an application framework, airline transport constitutes an interesting dynamical environment to apply the attained results from both optimization and stochastic approaches presented in this thesis: namely the Markov decision process as a formulation model and the particle swarm optimization method. More attention in recent years has been accorded to developing advanced techniques in the context of air traffic. This is due to the progress of air transport traffic (doubled since the early 1980s). The main objectives are the best allocation and management of airport and airline resources in the best way effectively and efficiently. Due to the dynamic stochastic operational environment of air transport, the scheduling problems nowadays faced by airport and airline managers have led to complex planning problems that require new models and methods. This is caused by the large variety of resource modules that have to be considered like terminals, flights, crews, baggage, etc., and they are highly interdependent. In the real world, stochastic disturbances in air traffic increased problem complexity. This is more considered in the latest research.

Airline operations are exposed to diverse sources of disruptions like bad weather conditions, airspace congestion, or some technical breakdowns, etc. In such situations, the resource schedules could be disrupted, so that these schedules are potential to become infeasible. Disruptions need fast recovery measures that result in flight delays or cancellations to recover many resources that are essential for operating flights (see Clausen et al. [8]). Instead of taking the risk of online disruption management, robust planning has been considered by some airlines as an efficient way of managing

possible disruptions in their schedules. New models and methods are given in this thesis with the objective to deal with those problems.

The contribution of this thesis will be presented in three main points:

Markov model is used to constitute a generic model integrated as a performance enhancement of the particle swarm algorithm. PSO algorithm is improved by both offline and online configuration of its parameters using HMM. The Homogeneous case is addressed in **Chapter I** and the heterogeneous case of PSO variants is presented in **Chapter II**.

In **Chapter III**, a new solution method of Markov decision process is given based on a model free approach. Thus, depending on the enhanced PSO algorithm of chapter II, we assign cooperation rules between independent reinforcement learners to accelerate convergence to optimal solutions. The proposed new PSO based cooperative Q-learning gives best results by making use of the cooperative aspect of PSO to enhance the Q-value search.

**Chapter IV, V and VI** are dedicated to applications. New methods and resolution strategies based on the results of the previous chapters are used to solve different problems of airline transport; primarily, to treat the cases of stochastic disturbances in the airline traffic.

---



# CHAPTER I :

## PARAMETERS SETTING OF HOMOGENEOUS PARTICLE SWARM OPTIMIZATION

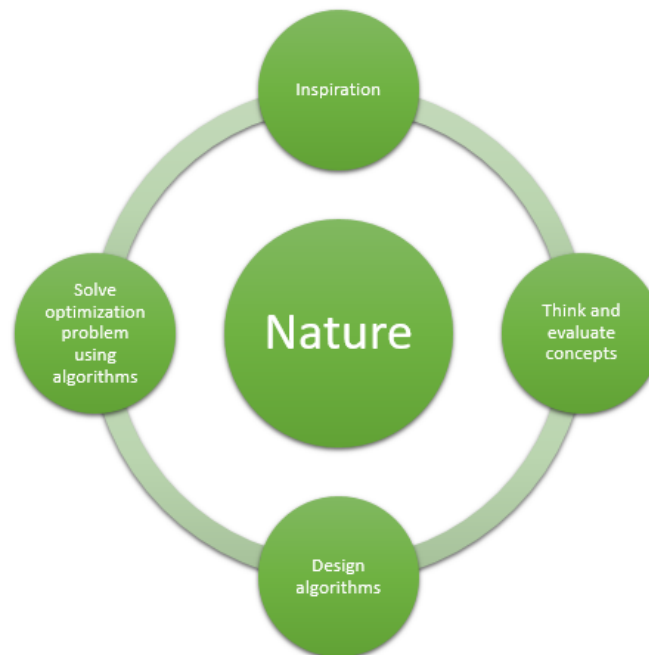
---

In this chapter, Particle Swarm Optimization is introduced and analyzed. Particle swarm optimization is a stochastic population-based metaheuristic algorithm. It has been successful in solving a high range of real-world problems. The classical PSO algorithm can be affected with premature convergence when it comes to more complex optimization problems; the resolution of PSO can easily be trapped into local optima. The primary concern is to accelerate the convergence speed and to prevent the local optima solutions. To defeat these weaknesses and to enhance the overall performances, a new technique is offered in this chapter building a machine learning technique for the parameter settings of the homogeneous particle swarm optimization. The homogeneous PSO has the same search behaviors of all particles, which are assumed to have been inspired by models of social influence guided by homogeneous individuals. Each main parameter of PSO is analyzed and enhanced either online (parameter control) or offline (parameter tuning). An HMM classification is used to enhance PSO performance. That is, we integrate the Hidden Markov Model (HMM) to have a stochastic parameter setting mechanism of PSO. The approaches are simulated and compared by experimental tests to the best-known state of the art.

## 1. BACKGROUND

### 1.1 NATURE-INSPIRED ALGORITHMS

Biologically Inspired Approaches from Animal groups offer paradigmatic illustrations of collective phenomena that have repeated interactions between individuals to generate dynamic behavior and reactions on a scale much bigger when compared to individuals themselves [9]. Many of the samples surrounding us contain synchronized movements of fish and birds in a school or a flock, the creation of vortices for bacterial colonies, the coordinated march of wingless locusts, and the synchronized flashing of fireflies. Many more cases can be observed around all of us. Greater computational studies of biological phenomena are affecting our comprehension of various aspects of computing itself and are changing how we comprehend computing properly. All those collective behaviors of natural mechanisms are discoveries of the pervasiveness of swarming in the natural world. Therefore, a huge range of awe-inspiring solutions supplied by nature is dealing with swarms (or schools, colonies, flocks, etc. [10]; the common term “swarm” will be employed throughout this thesis). The process of designing nature-inspired algorithms is depicted in Figure 1.



*Figure 1 Flow process of Nature algorithms design*

Swarms from nature represent typical illustrations of systems where aggregated behaviors are found, creating remarkable, synchronized moves without collision. In this kind of systems, the behavior of every group member depends on simple built-in responses, though their result is very complex from a macroscopic perception.

Particularly, Particle systems from nature can reveal a complex behavior if an algorithm could handle every particle's movement. Algorithms can also consider

several conditions present in the environment. By this means, the particles behavior could be reactive.

Reynolds [11] is the first to have applied a particle system to imitate the collective behavior of a flock of birds working with a reactive procedure. In the simulation, particles (symbolizing birds) had a "body" produced from a polygon mesh. The originality in Reynolds' work was that it took into consideration particles as an element of the environment. As the behavior of a particle was handled by the algorithm, and each particle was independent of the rest, the interaction between particles is also provided. The result exhibited an "emergent" collective behavior that was similar to a flock of birds. This behavioral rules discovered by Reynolds helped as a beginning point for the design of PSO. In Reynolds' model, there are three types of behavioral rules. In decreasing priority order, they are [11]:

- Collision Avoidance: This enables a particle to prevent collisions with their particular neighbors.
- Velocity Matching: Particle tries to match their neighbors.
- Centering : To keep the particle aggregated.

The Swarm movements is a crucial hypothesis in the production of the particle swarm optimization algorithm. It also represents the exchange of information between individuals of the same varieties that supplies an evolutionary advantage [2]. Serious research in systems where collective phenomena are obtained predisposed the basis for the development of swarm intelligence, lightly discussed in the next paragraph.

## 1.2 SWARM INTELLIGENCE

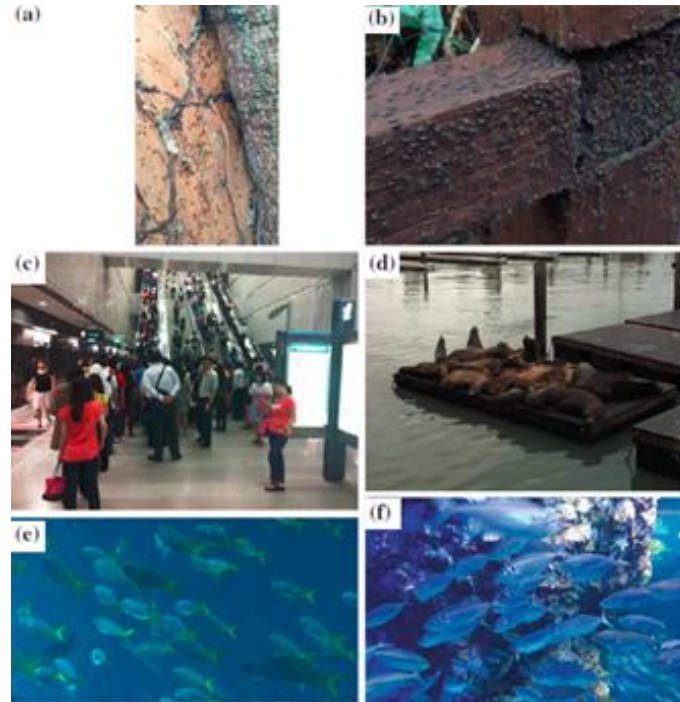
The swarm intelligence (also called collective intelligence) has emerged from the behavior of a group of social animals or insects that provide insight into metaheuristics [1]. For example, in the society of insects, there is very modest individual effectiveness, but various complex activities are performed for their survival. Problems, like getting and saving foods and picking up materials for future usage, need complete planning and can be performed by insect groups without any sort of controller or supervisor. Since the birth of swarm intelligence algorithms in the 1980s, it has continued to be studied and utilized extensively in domains like biology, industry, economics, and decision-making.

We call "swarm" in a general meaning to refer to any loosely structured collection of agents that interact. Some typical example of swarms are the swam of ants and bees; however, the metaphor of a swarm can be expanded to other systems with an equivalent structure. A flock of birds represents a swarm which their agents are birds, car traffic is normally a swarm of vehicles, a crowd at the street is simply a swarm of humans, an immune system is also a swarm composed of molecules and cells, and a market is another swarm including economic agents. Though the concept of a swarm implies a feature of collective movements in space, such as the swarm of a flock of birds, we have an interest in all kinds of collective behavior, not only spatial motion.

An illustration of a notably successful research algorithm in swarm intelligence is the particle swarm optimization (PSO), which is a very well-known swarm intelligence

algorithm for global optimization over continuous search spaces [12]. From its foundation in 1995, PSO has seduced the interest of several researchers, generating many variants of the original algorithm and many parameter automation approaches for the algorithm.

Swarm intelligence refers to a category of algorithms that imitate natural and artificial systems made up of many individuals that may coordinate by way of self-organization and decentralized control. Figure 2 [13] shows examples of observed swarms from nature, where a. Lane of aphids; b. Swarm of insects; c. Lane formation in human crowds ; d. Collective napping of sea lions; e, f. School of fish.



*Figure 2 Different observed swarms [13]*

A swarm algorithm works on the collective behaviors that derive from the local interactions of the individuals amongst each other and with the environment. Some typical examples of systems operating by swarm intelligence are colonies of birds flock, fish schools, animal herds, and ants. A standard swarm intelligence system contains these particular properties [13]:

- It is composed of multiple individuals.
- The individuals are comparatively homogeneous (i.e. they are either all similar or they are a part of only a few typologies).
- The interactions among the individuals depend on simple behavioral rules that make use of just local information that the individuals directly share or through the environment.
- The global behavior of the system comes from the local interactions of individuals together and with their environment.

The fundamental characteristic of a swarm intelligence system is its capability to work in a coordinated state without any existence of a coordinator or an exterior controller. Regardless of the absence of individuals in control of the group, the swarm in its

entirety does reveal intelligent behavior. This is the consequence of interactions of spatially located neighboring individuals working with simple rules.

The swarm intelligence had been applied in many diverse areas from the time it was initially introduced. The research and applications of swarm intelligence are primarily interest in flocking and schooling in birds and fish, ant colony optimization, the clustering behavior of ants, nest-building behavior of wasps and termites, particle swarm optimization, cooperative behavior in swarms of robots, swarm-based network management, etc.

## 2. PARTICLE SWARM OPTIMIZATION DEFINITION

Common well-known bioinspired algorithms applied to optimization problems is particle swarm optimization (PSO), which basically involves a machine-learning technique inspired by birds flocking in search of food loosely. More specifically, it contains a number of particles that collectively move on the search space looking for the global optimum.

Particle Swarm Optimization (PSO) includes swarming behaviors discovered in flocks of birds, swarms of bees, or schools of fish, as well as in human social behavior, from which the concept [14, 15, 16] has appeared as showed in the previous paragraph. PSO is a population-based optimization tool, which can be implemented and applied smoothly to deal with many different function optimization problems or the problems that could be converted to function optimization problems. The principle of PSO is an end result of combining the ideas shown in the previous paragraph. In a PSO algorithm, a multitude of solutions to an optimization problem are updated depending on several forms of interaction between them. This can be rather equivalent to what occurs in a particle system of the flock of birds; those rules were designed by Reynolds [11]. Reynolds's simulation inspired the concepts that affected the update rules in a PSO algorithm and provided the historical rules of the PSO algorithm. We will give more details about the PSO model in the following paragraphs.

### 2.1 PSO BASIC ALGORITHM

Introduced by Kennedy and Eberhart [2], PSO is a metaheuristic algorithm; the canonical PSO model comprises a swarm of particles, which are initialized with a population of a random selection of solutions. They move iteratively throughout the N-dimension problem space to look up the new solutions, where the fitness,  $f$ , can be measured as some qualities measure.

The learning procedure of the standard PSO algorithm is dependent on a particle's own experience and the experience of the most effective particle. A swarm of N particles is defined for an optimization problem of D variables, where each particle is given a random position in the D-dimensional space being a candidate solution. Every single particle has its specific trajectory, specifically position  $X_i$  and velocity  $V_i$ , and its movement in the search space is conducted by updating its trajectory at each iteration.

All of the particles change their trajectories based on their own experience and the experience of other particles. All particles possess fitness values that are measured by the fitness function  $f$  to be optimized.

The algorithm consists at each time step of changing the velocity (accelerating) of each particle toward its pBest (personal best) and gBest (global best) locations (see figure 3).

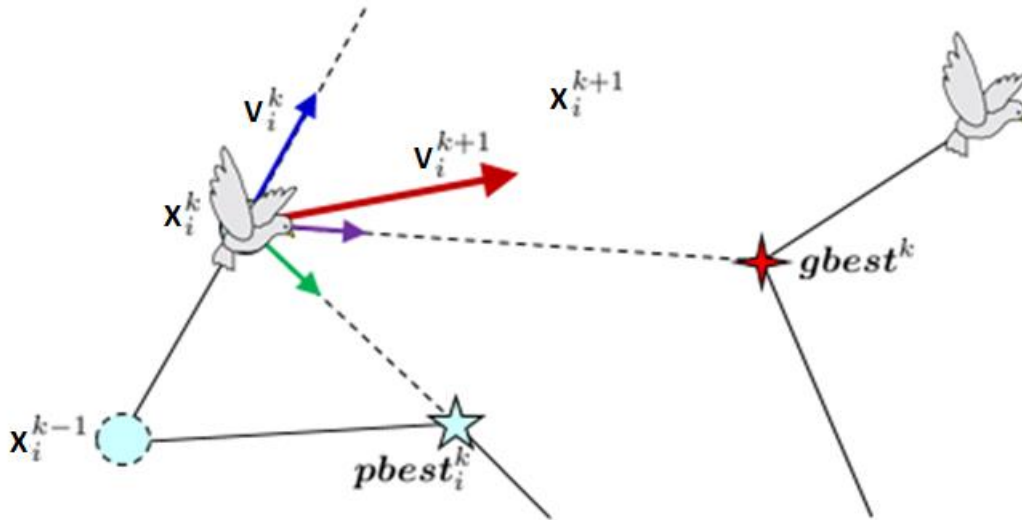


Figure 3 particle position update

The classical version of PSO adopts global topology where any two particles are connected, and each particle is informed by the gbest particle [2]. Then, each particle  $i$  has two vectors: its velocity vector  $V_i$  and its position vector  $X_i$ ; Those vectors are updated at each iteration  $t$  according to Eqs. (1) and (2):

$$V_i(t+1) = w V_i(t) + c_1 r_1 (pbest_i - X_i(t)) + c_2 r_2 (gbest - X_i(t)) \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

where:

- $N$  number of particles and  $i < N$  a particle index.
- $X_i$  position vector of particle  $i$  of dimension  $D$  :  $X_i = (x_i^1, x_i^2, \dots, x_i^D)$ .
- $V_i$  velocity vector of particle  $i$  of dimension  $D$  :  $V_i = (v_i^1, v_i^2, \dots, v_i^D)$ .
- $r_1$  and  $r_2$  are two  $D$ -dimensional uniformly distributed random vectors (generated at each iteration) in which every component varies in the interval  $[0,1]$ . Those two vectors are used to maintain the diversity of the population.
- $w$  is the inertia weight; its value is typically set up to vary linearly from 1 to near 0 during iterations.
- $c_1$  and  $c_2$  are acceleration factors. On the original version, they are set to a value of 2.



The pseudo-code of the particle swarm optimization algorithm is detailed in Algorithm 1.

<b>Algorithm 1:</b> Basic PSO algorithm
<p><b>Data:</b> The fitness function, the dimension of the problem <math>D</math> and the number of particles <math>N</math></p> <p><b>Initialization:</b> Initialize positions of particles <math>(X_1, \dots, X_N)</math>, velocities <math>(V_1, \dots, V_N)</math>, the fitness <math>fit</math>, pbest and gbest ; Set <math>t</math> value to 1 ;</p> <pre> <b>while</b> <math>t \leq t_{max}</math> <b>do for</b> <math>i = 1</math> to <math>N</math> <b>do</b>   <b>for</b> <math>j = 1</math> to <math>D</math> <b>do</b>     Compute the velocity <math>v_i^j(t)</math> ;     Compute the position <math>x_i^j(t)</math> ; <b>end</b>   Evaluate the fitness <math>fit_i(t)</math> the particle <math>i</math> ;   <b>if</b> <math>(fit_i(t) \leq f(pbest_i))</math> <b>then</b>     <math>(pbest_i \leftarrow X_i(t)) \&amp; (f(pbest_i) \leftarrow fit_i(t))</math> <b>end</b>   <b>if</b> <math>(fit_i(t) \leq f(gbest))</math> <b>then</b>     <math>(gbest \leftarrow X_i(t)) \&amp; (f(gbest) \leftarrow fit_i(t))</math> <b>end</b>   <math>t \leftarrow t + 1</math> <b>end</b> </pre> <p><b>Result:</b> The position of the best particle in the population <math>gbest</math></p>

The particle looks for the solutions in the problem space having a range  $[-x_{max}, x_{max}]$  (When the range is not symmetrical, it can be converted to its matching symmetrical range).

To be able to guide the particles efficiently in the search space, the maximum distance of particle one movement during a single iteration need to be clamped in between the maximum velocity  $[-v_{max}, v_{max}]$  provided by the Equation 3:

$$v_{ij} = \text{sign}(v_i^j) \min(|v_i^j|, v_{max}) \quad (3)$$

Where :  $v_{max} = p \times x_{max}$ , with  $0.1 \leq p \leq 1.0$ .

Generally  $v_{max}$  is selected as :  $v_{max} = x_{max}$ , i.e.  $p = 1$ .

More details about PSO parameters and other variants will be covered in the next paragraphs.

## 2.2 THE PARAMETERS OF PSO

The basic PSO algorithm is influenced by multiple control parameters, which are the dimension of the problem, number of particles, inertia weight, acceleration coefficients namely the cognitive and the social components, neighborhood size, number of

iterations, and the random vector values that scale the impact of the acceleration coefficients. In addition, if velocity constriction is applied, the maximum velocity and constriction coefficient as well influence the performance of the PSO. The following paragraph will focus on the analysis of the main PSO parameters which are: number of iterations, swarm size, acceleration coefficients and neighborhood topologies.

### 2.2.1 Number of iterations

This parameter is usually configured depending on the following criteria:

- Maximum number of iterations.
- Number of iterations without improvement.
- Minimum objective function error.

The number of iterations to achieve a good solution can be a problem-dependent. Too little iterations may stop the search too early. A too big number of iterations has the effect of unnecessary further computational complexity in the case of that the number of iterations is selected as a stopping condition.

### 2.2.2 Swarm size

This parameter refers to the number of particles in the swarm: the greater the number of particles in the swarm, the larger the initial diversity of the swarm. Given that a good uniform initialization method has to be used to initialize the particles, a large swarm enables larger areas of the search space to be covered at each iteration. However, more particles rise the per-iteration computational complexity, and the search degrades to a parallel random search. It is as well the case that more particles may lead to fewer numbers of iterations to reach a good solution, in comparison to smaller swarms.

We experiment the effect of a simple population increase over iterations in order to analyze the corresponding effect on convergence. Figure 4 represents the effect on varying population size in PSO. It shows that, by increasing population size with just random positions and velocities, fitness is ameliorated and improved, but did not get improved after a certain population size. Hence, as also deduced in [17] for the simple PSO, the population diversity is ruined when the particles increase. With a large number of particles, the optimization ability will not be improved; even if we increase population size, the fitness function is still constant after iteration number: 600 (see Figure 4). So using fewer particles improves the search ability. In contrast, CPU time increase fast while the population grows.

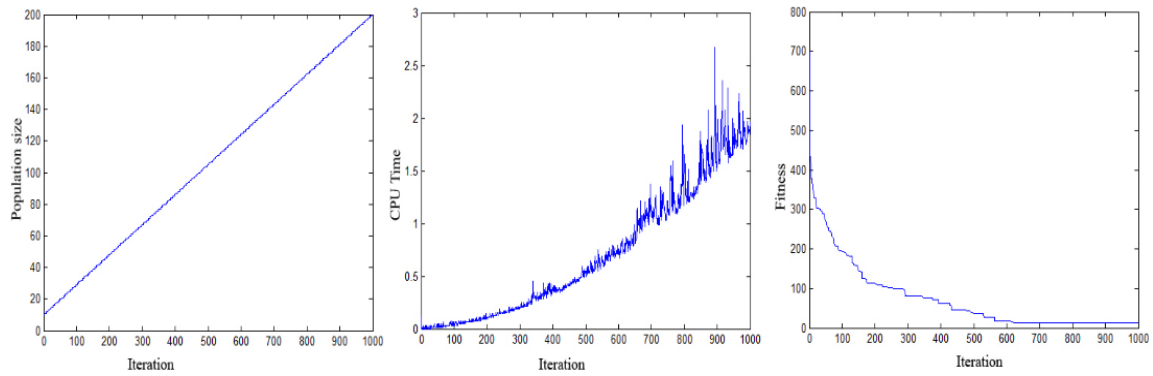


Figure 4 Effect of population size increase on HMM-APSO [18]

Because population variations have a significant effect on PSO regarding convergence speed and accuracy. Smaller population size will increase the probability of being trapped in a local optimum. However, added particles result in the rise of computing cost. However, it has been revealed in some empirical studies that the PSO possesses the potential to find optimal solutions with small swarm sizes of 10 to 30 particles [19, 20]. Even in [21], Success has even been attained for fewer than 10 particles. Even though empirical studies provide a general number of  $N \in [10, 30]$ , the optimal swarm size is commonly problem-dependent. A simple search space will need a smaller amount of particles than a rough surface to locate optimal solutions. The best choice of this parameter is to find a compromise between increasing or decreasing population and also the way how the population is increased or decreased according to the problem.

### 2.2.3 Acceleration coefficients:

The acceleration coefficients,  $c_1$  and  $c_2$ , combined with the random vectors  $r_1$  and  $r_2$ , control the stochastic impact of the cognitive and social parts on the entire velocity of a particle. The constants  $c_1$  and  $c_2$  are also known as trust parameters, where  $c_1$  presents how much self-confidence a particle possesses in itself, when  $c_2$  presents how much confidence a particle has got in its neighbors. Several combinations of values can be affected to  $c_1$  and  $c_2$  with different effects:

- If  $c_1 = c_2 = 0$ , particles maintain flying at their current speed until finally, they reach a boundary of the search space (presuming no inertia).
- If  $c_1 > 0$  and  $c_2 = 0$ , all of the particles are independent hill-climbers. Every single particle discovers the best position in its neighborhood by updating the current best position if the new position is better. Particles conduct a local search.
- If  $c_2 > 0$  and  $c_1 = 0$ , the whole swarm is attracted to a one specific point,  $g_{best}$ . The swarm becomes one stochastic hill-climber. Particles provide their potential from their cooperative nature, and are most effective when personal ( $c_1$ ) and social ( $c_2$ ) coexist in a proper balance, i.e.  $c_1 \approx c_2$  ( $c_1$  and  $c_2$  are equal or approximatively equal).

- If  $c_1 = c_2$ , particles are driven towards the average of pbest and gbest [22, 21]. Even while many applications work with  $c_1 = c_2$ , the proportion amongst these constants is problem-dependent.
- If  $c_1 \gg c_2$  ( $c_2$  negligible in front of  $c_1$ ), each particle is noticeably further attracted to its own personal best position, leading to excessive wandering.
- If  $c_2 \gg c_1$  ( $c_1$  negligible in front of  $c_2$ ), particles are considerably more highly attracted to the global best position, producing particles to move too rapidly towards optima.

However, for unimodal problems that have a smooth search space, a higher social component can be efficient, while complex multi-modal search spaces will find a larger cognitive component more beneficial. Low values for  $c_1$  and  $c_2$  lead to smooth particle trajectories, enabling particles to run away from good regions to explore before being taken back towards good regions. Large values induce more acceleration, with abrupt movement towards or past good regions [20].

Commonly,  $c_1$  and  $c_2$  are static, with their optimized values being determined empirically. Incorrect initialization of  $c_1$  and  $c_2$  may cause divergent or cyclic behavior [22, 21].

#### 2.2.4 Neighborhood Topologies

An essential component of the PSO algorithm is social information sharing between the neighborhoods. Common neighborhood topologies are gbest, lbest and the von-Neumann neighborhood, as displayed in figure 5 (where nodes illustrate particles and edges depict inter-particle influences). A particle's neighborhood is made up of all the particles accessible by an edge. Other topologies like figure 6 also given in the literature (ring and star topologies). The most simple neighbor structure is often the ring structure. Basic PSO functions with gbest topology, where the neighborhood includes the whole swarm, and thus all the particles possess the information of the globally discovered best solution. Every particle is a neighbor of every other particle. The lbest neighborhood comes with ring lattice topology: each particle produces a neighborhood composed of itself and its two or more immediate neighbors. The neighbors may not be near to the generating particle either relating to the objective function values or the positions; instead, they are selected by their adjacent indices.

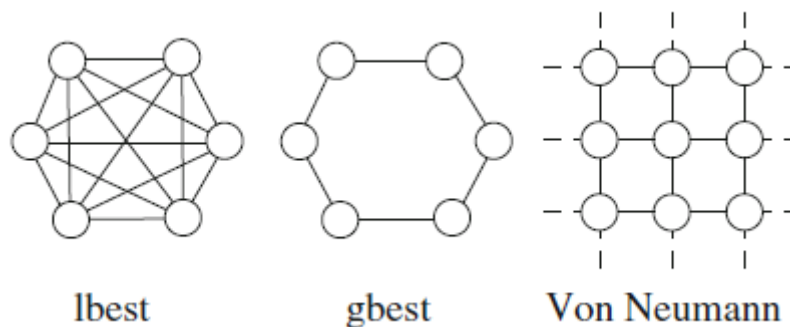


Figure 5 Common neighborhood topologies of PSO.

On behalf of the von-Neumann neighborhood, each particle has four neighbors on a two-dimensional lattice that is wrapped on all four sides (torus), and a particle is in the center of its four neighbors. The possible particle number is limited to four.

According to the tests on many social network structures, PSO using a small neighborhood seems to perform better on complex problems, while PSO with a large neighborhood could perform considerably better on simple problems [23, 24].

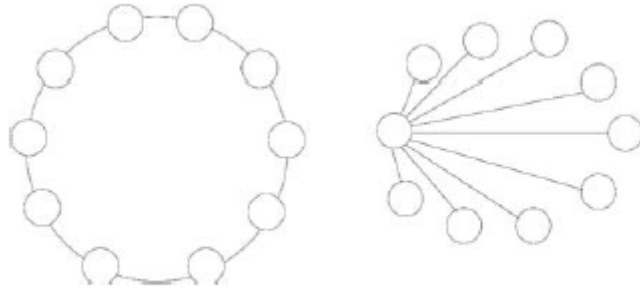


Figure 6 ring topology (left) and star topology (right)

## 2.3 CONVERGENCE ANALYSIS OF PSO

Initial empirical analyses of the basic PSO reveal that PSO is an effective optimization approach. Some research has demonstrated that the basic PSO enhances the performance of other stochastic population-based optimization algorithms just like genetic algorithms [21, 25, 26]. The particle follows a convergent trajectory for the majority of the time steps [20]. Figure 7 is an illustration of such behavior. However, it was as well shown that the basic PSO has some considerable defects which can cause stagnation [27].

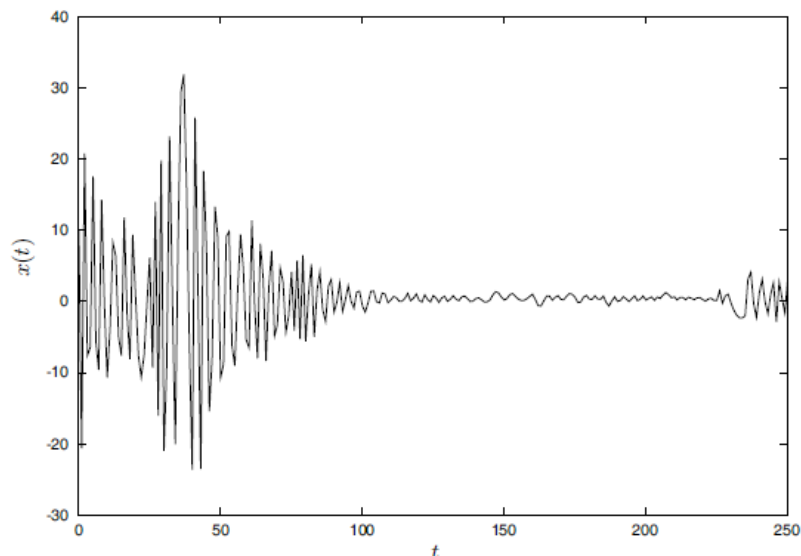


Figure 7 Stochastic Particle Trajectory for  $w = 0.9$  and  $c_1 = c_2 = 2$

Since its start, PSO analysis was of an empirical aspect. There was no vision regarding how the algorithm definitely worked well or if it was qualified to converge in anyway.

In the beginning, it was clear that (occasionally) particles raised their velocities extremely fast. This induced the particle swarm to explode [14]. The first way to deal with this problem was to clamp the maximum permitted velocity to a fixed value. Empirical studies recommended that a good selection was to set:

$$v_{max} = x_{max} \quad (4)$$

where  $x_{max}$  is the limit of the search range [28].

It is essential to note at this point that in case the trajectory of the particle converges, it will conduct so towards a value derived from the line around its personal best position and the global best particle's position (see equation below). In theory, each particle in PSO is demonstrated to converge to the weighted average of  $pbest(i)$  and  $gbest$  [22]:

$$\lim_{t \rightarrow \infty} x_i(t) = \frac{c_1 pbest_i + c_2 gbest(t)}{c_1 + c_2} \quad (5)$$

where:  $c_1$  and  $c_2$  are the acceleration factors of PSO.

The key guarantee for the convergence of the PSO algorithm is the good selection of its parameters. Special analysis [22, 21, 14, 29] indicates that the convergence and the overall performances of the PSO are very sensitive to the values of its control parameters. As an illustration, [29] provided a methodology to determine convergent behavior and some suggestions to the choices and the acceleration coefficients and the constriction factor.

Further works from literature given as a conclusion, even though there are some interesting theoretical results in PSO convergence, empirical work is always required to tune the parameters of a PSO algorithm to solve any specific problem. The next paragraphs will address this issue.

## 2.4 ADVANTAGES AND DRAWBACKS

The main advantage of working with the PSO is its simplicity: its concept and capability to be implemented in a small number of lines of code. Even more, PSO additionally possesses a short-term memory (it will be analyzed in the next section), which allows the particles to move through the local best and the global best positions. Other choices, just like genetic algorithms (GA), tend to be complex and, the majority of the time, they do not take into account the past iteration or even the collective emergent performance. As an illustration, in GA, if a chromosome is not picked, the information covered by that individual is definitely lost.

The PSO is much like other optimization algorithms that are not exhaustive methods, for example, the brute-force search [3]. Despite their good facilities, a common problem with this type of algorithms is that of getting stuck in a local optimum, or suboptimal solution. As a result, PSO could work well on one problem but yet fail on another problem.

Generally, the primary downsides of PSO could be summarized in the following:

- Swarm Premature convergence: Particles of the swarm try to converge to a singular point, positioned on a line around the personal best positions and global best. This point is not necessarily assured for a local optimum [30]. One more purpose could possibly be the fast amount of information circulation between particles, which causes producing similar particles. This leads to a decrease in diversity and the risk of being trapped in local optima is elevated [31].
- Parameter settings dependence: This invokes the high-performance variances for a stochastic search algorithm [32]. In most cases, there is not any particular set of parameters for different problems. For instance, through simple observation of Equation (1), raising the inertia weight  $w$  increases the velocity of the particles and induces more exploration (global search) and less exploitation (local search). Consequently, selecting the best set of parameters is not a simple task, and it could be distinct from one problem to another [32].

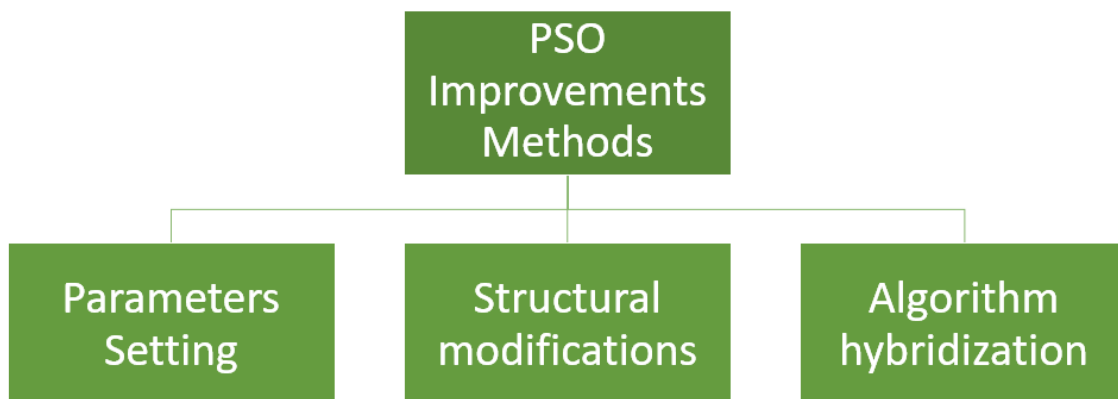
With a purpose to defeat the PSO trend to get stuck in undesirable solutions and enhance its convergence, various authors have recommended other adjustments to the parameter combinations of the PSO algorithm. A review of enhancement methods will be drawn in the next section.

## 2.5 PSO IMPROVEMENTS

Three distinct classes of techniques were reported to be improving PSO (figure 8):

- Setting parameters.
- Modifying components of the algorithm
- Pairing the algorithm with other algorithms.

Parameters setting refers to setting the different parameters of PSO just like the topology, acceleration coefficients, inertia weight, and population size. Modifying components represents modifications of the velocity or position update rule (which includes likewise creating new components; changing the manner they are computed). Combining the algorithm with other algorithms offers hybridization of PSO with other techniques.



*Figure 8 PSO improvement methods*

Our interest is given to the parameter setting due to its proven impact on PSO performances (see the two last paragraphs). The calculation of the movement inside PSO constantly incorporates several numerical parameters. In the basic case, these parameters are constant and defined by the user. Several variants have been offered in this way. In most classical variants, the parameter values depend just on the number of iterations [33, 4, 34]. Further advanced variants adjust the values depending on the information that is gathered during the run. A specific area of research is how to establish an algorithm that is adaptive as possible, to ensure that the user does not need to tune any component [34, 35, 36]. In the next section, we are interested in PSO variants, especially, which have improved the overall performance of PSO by using particular settings of its parameters.

### **3. IMPROVED PSO VARIANTS FROM LITERATURE**

With the purpose to defeat the PSO trend to get stuck in undesirable solutions and enhance its convergence, many authors have suggested other modifications to the parameters of the PSO algorithm. Various models that differ in their exploration and exploitation behaviors are included.

Among the mainly active areas of studies in PSO has been relating to algorithmic modifications. This has triggered multiple algorithmic variants of the original PSO algorithm. In this subsection, we will summarize the algorithmic variations that are an element of the empirical comparison with the proposed approach. In this state of the art, the analysis of the most effective and promising algorithmic modifications related to parameters setting has been a challenging task in iterative optimization methods in recent years [37]. Two distinct methods for setting parameters of an optimization algorithm could be conceived: parameter tuning and parameter control. Parameter tuning represents the setting parameters of an algorithm by using just experiments to some constant values. On the other hand, Parameter control relates to designing approaches which vary the value of parameters during the execution. Parameter control techniques are categorized as well into three groups: deterministic, adaptive, and self-adaptive. In deterministic parameter control, a rule (named time-varying rule) is developed to determine the value of a parameter depending on the iteration number. In adaptive parameter control methods, a function is produced that maps some feedback from the run right into the value of the parameter. In a self-adaptive parameter control methodology, the parameters are encoded into individuals and are altered during the execution by the optimization algorithm.

Besides, parameters control of PSO can be done in the context of the homogeneous or heterogeneous swarm. In the first one, all the swarm adopts the same behavior and the second refers to multiple behaviors during the same run of the PSO. In this chapter we consider the case of homogeneous PSO; the next chapter will deal with the heterogeneous one.



In this section, articles that have examined distinct parameters for PSO are reviewed. The arrangement of description is chronological and/or hierarchical (to some scope). This will serve as a background to introduce the ideas that provide the proposed PSO enhancement method.

The state of art of parameters control of PSO is presented in terms of homogeneous PSO swarm. The standard PSO and the majority of its improvements [38] utilize homogeneous swarms where every one of the particles adopts specifically the same behavior. That is, particles use the same velocity and position update rules. The result is that particles have the same exploration and/or exploitation properties. Several homogeneous variants have studied the control of the different PSO parameters.

The parameter control mechanism has many advantages. It can enable to use appropriate parameter values in different stages of the search process, to make use of the accumulating information to enhance performance in later stages and to liberate the user from the charge of selecting parameter values [39]. For each optimization problem, a number of these parameter's values and selection has a big effect on the efficiency of the PSO algorithm [40]. Three major tactics can be found in literature, which are available to categorize PSO parameters. The first is to vary this parameter randomly as a constant value, and the second is that the parameter varies according to the iteration number. In the third one, the value of this parameter at each iteration changes according to the results acquired by the particles till the current iteration [41], where the value of this parameter at each iteration, varies according to the results obtained by the particles until this iteration. More details of the parameter setting of PSO are presented per parameter as follow:

### 3.1 POPULATION SIZE

Concerning the parameter configuration of PSO population size, a proposed approach to enhance the adaptivity of PSO is to vary the population size [42], which should be well chosen in order to achieve good results [42]. Some paper has been interested in the definition of the best configuration of PSO. For instance, one the one hand, concerning the population size, [43] proposed the following formula to set the swarm size as a fellow:

$$N = \text{Int} (10 + 2\sqrt{D}) \quad (6)$$

where D is the dimension of the problem, and Int is the integer part function.

An adaptive approach has been proposed in [42]. The authors defined two strategies which are the increasing and decreasing strategies. Moreover, we can notice from the literature that adaptive population size approaches have been proposed in other forms and other terms (dynamic, incremental, varying population size, etc). For instance, a similar idea has been introduced by [17] in which the authors inspired by the notion of birth and death of particles in nature to increase and decrease the population size. The authors defined three functions (Damping function, Sine function, Sine Attenuation function) to update the population. Furthermore, [44] and [45] interested especially in the diminution of the worst particles in the population, they found that this idea may reduce the computational complexity of PSO. [46] also proposed a

strategy of augmentation of the population to enhance the exploration of the algorithm and diminution of it for exploitation purposes. Another population size approach has been proposed in [46] in which the neighborhood is used to allow varying the size of the dynamic population, and has also been integrated into the multi-objective PSO [47]. Moreover, population size variation has been applied in some fields such as power systems [48].

Furthermore, [49] integrated the adaptive population size concept into the learning mechanism of PSO. For this purpose, the incremental social learning which aims to facilitate the scalability of systems composed of multiple learning agents has been used for enhancing the performance of PSO. That is, every time a new agent is added to the population, it should learn socially from a subset of the most experienced agents.

### 3.2 PSO TOPOLOGY

Additionally, other PSO improvement variants involve the modification of PSO topology. In [50], the authors have investigated and tested several social network structures for the PSO topology. Moreover, a fully informed particle swarm optimization (FIPSO) has been offered based on the learning of the best topology as in social networks.

In particular, some papers have adopted the learning concept to adjust PSO parameters. The parameters that have to be defined are the velocity clamping, the inertia weight ( $w$ ) and the acceleration factors (cognitive attraction and social attraction). Thus, a number of methods have been proposed to learn the best values of these factors. This problem can be formulated as a learning process in which each particle learns from the obtained data and predict the values of the parameters in accordance with the history of its values and the values of other particles.

Another commonly used algorithm is the comprehensive learning which has been introduced by [51] and extended by [52]. In this case, each particle learns from another particle that is chosen according to a learning probability. This approach offers good performance on complex multimodal functions at the expense of the convergence speed for unimodal functions.

The basic idea behind the orthogonal learning PSO proposed by Zhan et al. [53] is to determine the best combination of historical values of the particle itself and other particles.

Another approach is the feedback learning which has been introduced by Tang et al. [54]. In the mentioned work, the feedback fitness information of each particle (described especially by each particle's history best fitness) is used to determine the learning probabilities. These probabilities affect the acceleration parameters of PSO.

In [55], the learning has been done by different examples instead of one. The convergence has been analyzed theoretically by considering a Markov process of the PSO algorithm. Also, [56] proposed the fully informed PSO which is based on the learning of the best topology as in social networks.

### 3.3 INERTIA WEIGHT

Inertia weight  $\omega$  was used to manipulate the effect of the particle's velocity; it consequently impacts global and local search capabilities of the particle inside the swarm. A suitable value selection of  $w$  could balance global and local search capacities, which allows the PSO algorithm to give better performances. A smaller  $w$  may fortify local search capacities. However, the higher  $w$  value may accelerate the convergence speed [28]. Due to the importance of this parameter, a variety of diverse approaches for selecting the value of inertia weight at each iteration has been proposed.

In [57], an analysis of the impact of the inertia weight and maximum velocity permitted on the overall performance of PSO. A time-varying inertia weight offers a higher performance fixing a maximum velocity.

Furthermore, in [58], the inertia weight was also dynamically adjusted at each time step by taking into account the distance between the particles and  $gBest$ .

The classical way to define the inertia weight was proposed by Shi and Eberhart [59]. It consists of linearly decreasing  $w$  with the iterative generations using a generation number. It gives good results. However, some papers proposed other variants of  $\omega$ . For instance, Tang et al. [54] proposed a quadratic decreasing and Zhan et al. [60] chose a sigmoid decreasing of this parameter. These time-varying methods of the inertia weight are chosen in order to control the exploitation and exploration of PSO.

Concerning the values that have to be assigned to the inertia weight and acceleration parameters, according to [61], the algorithm can converge if the following conditions are satisfied:

$$0 < c1 + c2 < 4 \quad \text{and} \quad \frac{c1 + c2}{2} < w < 1 \quad (7)$$

On the other hand, concerning the inertia weight parameters, various adaptive strategies have been proposed to adapt it. In [36], a fuzzy system is implemented to adapt the inertia weight dynamically.

Yang et al. [62] defined a strategy in which the inertia weight dynamically changes based on the run, and evolution state Different nonlinear variations are tested with different modulation index.

Also, Chatterjee & Siarry [63] presented a nonlinear way for varying the inertia weight which employs aggressive, coarse tuning during initial iterations to achieve better search of the solution space to quickly arrive near an optimum solution and to apply mild, fine-tuning during later iterations gradually.

Furthermore, Ting et al. [64] used a similar concept of the adaptive crossover used in differential evolution algorithm for adapting the inertia weight. For instance, Ding et al. [65] employed a stochastic local search to adjust the inertia weight in terms of keeping a balance between the diversity and the convergence speed.

Suresh et al. [35] incorporated two modifications into the classical PSO which are the modulation of the inertia factor and the modification of the position update. The purpose of the first one is to adapt the inertia weight over different fitness landscapes.

Moreover, Kentzoglanakis & Poole [66] proposed three oscillating inertia weight functions of time that to be approached with better accuracy, where the value of inertia weight was adaptively changed during the search process.

Furthermore, other parameters have been proposed to replace this parameter. For instance, another coefficient has been proposed and incorporated into velocity by Eberhart & Shi [67] for the same purpose of the inertia.

Concerning the use of probabilistic approaches, Zhang et al. [41] proposed a Bayesian PSO in which the inertia weight vector is determined on the basis of the past particle position by maximizing the posterior probability density function of the weight.

Also, in [68], a dynamically decreased particle velocity limit strategy has been used instead of the inertia weight. More generally, De Oca et al. [69] presented a study on the effects of using different schedules (using both time-decreasing and time-increasing inertia weight strategies).

The setting of the inertia weight can depend on the particle itself as in [70]. That is, based on the success of particles, each particle affects the adjustment of the inertia weight. A time-decreasing inertia weight is used to allow the user to tune the algorithm's exploration/exploitation capabilities.

### 3.4 ACCELERATION COEFFICIENTS

Enhanced PSO algorithm can be achieved just by using adapted coefficients  $c_1$  and  $c_2$ . That is, it is known that the accuracy of PSO depends on the selection of the appropriate values of parameters and their values through the search process (see for instance [71]).

The most used adaptive strategy of  $c_1$  and  $c_2$  has been formulated by Zhan et al. [60]. It consists in updating its values according to four defined states which are: exploration, exploitation, convergence and jumping out. Also, Fuzzy approaches have been used to adapt other PSO parameters such as acceleration factors [60] and the velocity climbing.

Other papers are interested in the relation between the learning behavior of PSO and acceleration parameters. For instance, Kamalapur and Patil [72] examined the link between these parameters and the topology of PSO. Moreover, Subbaraj et al. [73] interested in choosing the best values of these parameters.

In [74], the acceleration coefficients are dynamically updated through PSO iterations. Epitropakis et al. [75] studied the effect of the social and cognitive parameters on PSO convergence and used differential evolution to enhance it. In [76], the learning process of fitness information is used to control the parameters of PSO.

On the other hand, [77] proposed four operators which play similar roles as the four states the adaptive PSO defined in [60]. Their approach is based on the idea of assigning to each particle one among different operator values based on their rewards.

## 4. PSO PARAMETERS SETTING USING HMM

The configuration of algorithms is one of the main challenges of the PSO metaheuristic; the reason lies in the fact that the PSO performances depend heavily on the chosen parameter values. Concerning metaheuristics, their parameters setting can be done in two ways [78]. The off-line configuration involves the adjustment of parameters before running the algorithm while the online configuration consists of adjusting the algorithm parameters while solving the problem (also called automated configuration). The first problem is known as the ‘parameter tuning problem’ and the second one as the “parameter control problem”.

In this section, we propose a new method that makes it possible to optimize the performance of the PSO algorithm automatically. Firstly, a method is designed specifically for particular classes of problem instances, creating the most likely significant better performance in real-world applications. Secondly, it will be designed for online PSO algorithm control mechanisms that adapt parameter settings within the execution that can produce enhanced performance than the fixed algorithm configuration techniques. Before giving the proposed PSO parameters setting technique, the used machine learning, which is the Hidden Markov Model, is presented.

### 4.1 THE HIDDEN MARKOV MODEL

#### 4.1.1 Model definition

The full state  $S_t$  of the system rarely uses straight observable once modeling complex operations in the real world. This is often caused by normal occlusions or imperfect sensor observations. A Hidden Markov Model (HMM) [79] can model related cases, as displayed in Figure 9. In this model, the complete system state is regarded as hidden, and therefore it exists only an access to sensor observations  $Y_t$ .

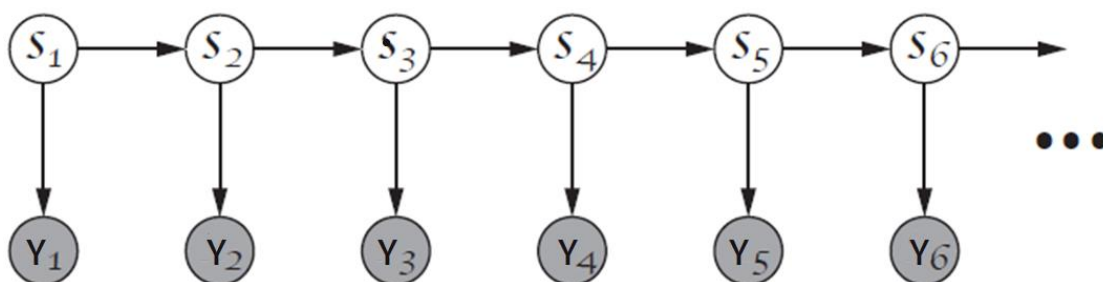


Figure 9 The graphical model for a hidden Markov model

The main considered problem is then to estimate  $p(S_t | Y_{1:t})$ . The value of the hidden state supplied the sequence of observations about the given point. The process of inferring the state is the task of the system perception. Instances of such perception systems are autonomous, obstacle detectors, localization subsystems, etc. In this

chapter, we discuss the task of learning a hidden Markov model for dynamic configuration of particle swarm PSO.

As a definition, A hidden Markov Model (HMM) is a triple  $\lambda = (\Pi, A, B)$  where :

- $\Pi = (\pi_i)$ ;
- $A=(a_{ij}), P(S_t = i|S_{t-1} = j), i, j \in [1, N]$
- $B = (b_{jk}), P(Y_t = k|S_t = j), j \in [1, N], k \in [1, N]$ .

where:

- $S$  is a set of states,
- $Y$  is a set of observations,
- $P$  is a state transition probability  $p(S_{t+1}|S_t)$  that state  $S_t$  will lead to state  $S_{t+1}$ , is a conditional probability  $p(Y|S)$  of observing  $Y$  at state  $S$ .

Also, with the many observations, the precise value of the hidden state could not be generally established with certainty. It is typically desired to approximate it and preserve its complete distribution. This distribution is known as Belief state  $B_t p(S_t|Y_{1:t})$ .

Its essential property is that the sequence of belief states forms a Markov chain; i.e., the knowledge of a belief state  $B_t$  captures the full information about the past observations. Learning algorithms for the HMM model namely, the Viterbi and Baum-Welch will be presented in the proposed approach.

### 4.1.2 Hidden Markov Model Learning evaluation

There are different major methods for evaluating a Hidden Markov Model (HMM).

- Likelihood of test data:

In this approach, one will need to retain some test data and calculate the likelihood of the test sequences by using the forward algorithm.

- Errors Prediction from data:

A significant prediction task relies on the application. For example, the task could be considering predicting the future. In such a case, the forward algorithm is used to track the state at  $t$  of the HMM and use that to predict the expected observation at some future time  $t+k$ . This is done for all  $t$  to calculate the mean error.

- Evaluate the hidden Markov chain structure:

An additional concern is the choice of an appropriate model structure. To be able to use the HMM model for a given problem, the analyst must speculate primary values for the entries of the state transition matrix  $A$ . In general, one may use  $a_{ij} \neq 0$  for all  $i$  and  $j$ . However, according to the particular problem, the analyst may have a good grasp on the structure of the underlined Markov chain. But the hidden Markov chain initial structure can be maintained throughout the iterations.

Equivalent comments apply to the initial choice of the conditional probabilities of symbol emission. The probability of observation when in a state can be very low and probably could be ignored; however, it can affect the whole observation sequence. These forms of structures could be known from some prior knowledge of the model and can be forced within the analysis.

Thus, a good initial structure may lead to more accurate and efficient parameter estimation result for the problem under investigation. In the same direction, we address PSO based HMM for parameters setting.

## 4.2 PARAMETER TUNING (OFFLINE)

The tuning approach consists in finding the most suitable configuration of an algorithm for solving a given problem. Machine learning methods are usually used to automate this process [80]. They may enable to construct robust autonomous artifacts whose behavior becomes increasingly expert.

[78] has surveyed the main approaches that can be used for parameters tuning of optimization algorithms. In [81], the tuning approaches have been classified into four categories depending on the number of parameters and the number of functions. [82] has presented the three most used procedures for tuning algorithms which are: Racing Procedures, ParamILS, and Sequential Model-Based Optimization. The racing approaches have been well investigated in the past, and many variants have been proposed (sampling F-Race, iterative, F-Race, tNO-Race, etc). In [83], the authors showed how the training sample could be classified into positive and negative examples. This classification may enable us to use a supervised machine learning method.

Moreover, metaheuristics have been used for tuning metaheuristics. Indeed, the off-line configuration of the algorithm can be formulated as an optimization which aims to minimize the objective function. This idea has been introduced for evolutionary algorithms as meta-evolutionary algorithms. In other terms, an evolutionary algorithm is used to configure another one. A similar idea of the meta-evolutionary has been proposed in [84]. That is, the algorithm that has to be tuned can be used to tune the algorithm itself. The specificity of the paper is that the authors proposed to use a multi-objective approach. The firefly algorithm has been used to examine the proposed framework.

The tuning of metaheuristics is related to the generic notion of hyper-heuristics which consists of finding the most suitable configuration of heuristic algorithms such as local searches (simulated annealing, tabou search, etc). Machine learning has been used also to deal with hyper-heuristics [85]. Furthermore, [82] proposed a hyper-heuristic solver based on a choice function which combines various numbers of strategies to learn the weighted mixture of heuristics for a given problem class. Also, [86] proposed another choice function which tends to rank the heuristics according to their ability to properly solve an instance of the problem and PSO has been used then for the tuning of the choice function parameters.

The use of machine learning for the tuning problem has been popularized by [87]. It consists of learning from problem instances. In particular, HMM has been successfully applied in a number of problems which have similarity with the tuning problem. The current section will use the hidden Markov model to find the best configuration of the PSO algorithm based on the estimation of the most likely state. The experiment consists of finding the best parameter values of the particle swarm optimization algorithm for a given problem.

#### 4.2.1 HMM-based tuner Model for PSO

Our interest in this article is given to the automated algorithms tuning by machine learning. This issue can be considered as a special case of tuning metaheuristics. Solutions of the parameter tuning problem are parameters with maximum utility [88], where utility is based on definition of algorithm performance, some objective functions or problem instances. We associate this utility calculation to a metric function  $m$ .

In this approach, we consider the case of non-iterative tuners, where we execute the generation step only once, during initialization, thus creating a fixed set of configuration vectors. Each of those vectors is then tested during the test phase; then a final machine learning phase using HMM is to find the best vector in the given set. Our method of tuning differs from the iterative one, where the set of vectors starts with a small initial set and creates new vectors iteratively during execution.

The following parameters can define the PSO configuration or parameter tuning problem of PSO:

Given

- an algorithm  $A$  with a number  $l \in \mathbb{N}$  of parameters  $p_1, \dots, p_l$  that affect its behaviour,
- A space  $C$  of possible configurations for the algorithm  $A$ , where each configuration  $c \in C$  specifies values for  $A$ 's parameters where  $A$ 's behavior on a given problem instance is completely specified.
- A set of problem instances  $I$ ,
- A performance metric  $m$  that measures the performance of  $A$  on instance set  $I$  for a given configuration  $c \in C$
- A rank function  $rank(m) \in \mathbb{N}$  over the metric values set, this function orders the metric values obtained on instance  $i \in I$  for a given configuration  $c \in C$ .

The objective of our tuner problem is :

To find a configuration  $c^* \in C$  that results in optimal performance of  $A$  on all instances set  $I$  according to the rank value  $k$  returned by the  $rank()$  function of the evaluation of a metric  $m$ .

For the algorithm whose performance is to be optimized or also the target algorithm, we use  $A(c)$  to denote target algorithm  $A$  under a specific configuration  $c$ .

We identify three phases of our tuning problem: Generation phase, test phase, and evaluation phase.



In most cases of tuning algorithms, these three phases can be merged or executed in parallel as in the iterative tuner, where it starts with a small initial set of parameters and generates new parameters vectors iteratively during the test phase.

In this approach, every step is performed only one time after its previous step is finished. So, if a step is executed, it will not be performed again. Our tuning problem is designed with the following template (see figure 10):

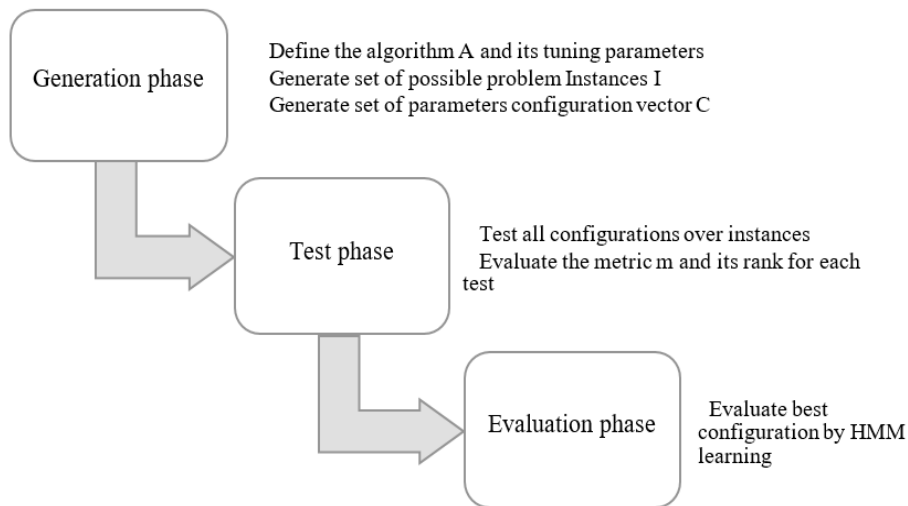


Figure 10 Description of the HMM-based tuner.

#### a. Generation phase

In this phase, according to the algorithm A to be tuned, and according to the problem subject to tuning, we generate all required elements for tuning by machine learning.

For parameters generation, this step is about designing the algorithm A for a given application amounts to selecting good values for the parameters. Two main classes of the configurations are identified in the literature and can be found as in [89]: selection operators and variation operators. Also, we can distinguish between qualitative and quantitative parameters.

A configuration  $c$  will be a vector of a combination of those operators. Let  $N$  the number of generated configurations to test in the test phase:  $C = (c_1, \dots, c_N)$ , and  $T$  the number of generated instances of the considered problem:  $E = (e_1, \dots, e_T)$ .

#### b. Test phase

The objective of this test phase is to evaluate all configurations in the set C on instances I. Then, each result of an execution of a configuration:  $c \in C$  on an instance  $e \in E$  will be evaluated and ranked according to some metric  $m$  (considered as an evaluation test for performances comparison).

First, we define a Markov process  $\{S_k\}_{k \in [1, N]}$  on the simulation test as a stochastic process which returns a number from 1 to N, where N is the number of evaluated configurations.

We define also observed evaluation values from executions  $\{Y_k\}_{k \in [1, T]}$ . It indicates to the observed result of simulations and corresponds to the rank value of the executed configuration  $c$ , or also as defined, the rank value of the process  $\{S_k\}_{k \in [1, N]}$ .

We define firstly two additional vectors where results from all executions will be stored, which are the observation sequence  $O$  and state sequence  $Q$ :

$$O = (o_1, o_2, \dots, o_T), \quad Q = (q_1, q_2, \dots, q_T) \quad , T \in \mathbb{N} \quad (8)$$

where :

$T$ : number of test runs, also indicates the length of observation sequence  $O$  and state sequence  $Q$ .

And:

$$o_i = \text{rank}(m_{ij}) \in \{Y_k\}_{k \in [1, T]} \quad (9)$$

$$q_i \in \{S_k\}_{k \in [1, N]}, \quad i, j \in [1, T], [1, N] \quad (10)$$

$m_{ij}$  : Metric value by executing configuration  $j$  on the instance  $e_i$ .

$o_i$  : rank of the metric value of executing configuration  $c_j$  on instance  $s_i$ , is the rank order over all configuration of metric value on the instance  $e_i$ .

$q_i$  : corresponds to the executed configuration, called state sequence.

The pseudocode of the test phase is depicted in Algorithm 2:

<b>Algorithm 2: Tuning Test</b>
<p><b>Input:</b> Algorithm <math>A</math> , configurations <math>C</math> of length <math>N</math> and instances <math>I</math> of length <math>T</math>.</p> <p>for each instance <math>s_i, i</math> from 1 to <math>T</math> do</p> <p style="padding-left: 2em;"><b>for</b> each configuration <math>c_j, j</math> from 1 to <math>N</math> do</p> <p style="padding-left: 4em;">Execute instance <math>e_i</math> with configuration <math>c_j</math></p> <p style="padding-left: 4em;">Evaluate the metric value <math>m_{ij}</math>.</p> <p style="padding-left: 2em;"><b>end</b></p> <p style="padding-left: 2em;">Evaluate rank values of metrics on instance <math>e_i</math>.</p> <p><b>end</b></p> <p><b>return</b> : Observation sequence <math>O</math>, state sequence <math>Q</math></p>

### c. Performance evaluation phase

Evaluation of the best configuration  $c$  is done in this thesis with a machine learning technique, which uses execution data from the test phase to evaluate the best configuration  $c^*$  of the algorithm  $A$  for the problem, with consideration of all test instances.

Machine learning algorithms will be able to extract target knowledge (in our case) from training examples, models of desired behavior drawn from experience. Thus, it is important that the learner not overfit, that is, not getting information so close to the training examples. So, we choose the hidden Markov model (HMM) as an adequate machine learning technique to extract knowledge from all collected data in the test

phase. Our motivation beyond the use of HMM is the successful applications in many fields, which are similar to our problem as detailed in the literature section. The specificity of HMM is that it can include stochastic information on transitions between states and also observations to give the most likely state of the parameter values.

According to the best of our knowledge, the HMM algorithm has not been yet investigated for the tuning problem.

We define the Hidden Markov Model by a triple  $\lambda = (\Pi, A, B)$ , where all processes are defined on a probability space. The triple components are respectively the vector of the initial state probabilities, the state transition matrix and, the emission matrix.

- $\Pi = (\pi_i)$ ;
- $A = (a_{ij}), P(S_t = i | S_{t-1} = j), i, j \in [1, N]$
- $B = (b_{jk}), P(Y_t = k | S_t = j), j \in [1, N], k \in [1, N]$ .

We suppose that a configuration  $c$  is identified as the state of HMM, then:

- The state  $\{S_i\}_{i \in [1, N]}$  takes values from the set  $C = \{c_i\}_{i \in [1, N]}$ .  $N$  is the number of configurations.

Moreover, for every configuration of the algorithm  $A$ , we observe from tests the rank of each execution, then:

- The observed parameters of this hidden chain  $\{o_i\}_{i \in [1, N]}$  take values from the set of values of a rank function:  $\{1, \dots, N\}$ .
- The output of the test phase is considered as observation sequences and state sequence for our HMM. It forms a stream of observed ranks for configurations set  $C$ .

This model will serve to execute the Baum-Welch re-estimation algorithm [90] on the two output sequences of the test phase: sequence  $O$  of ranks and sequence  $Q$  of states. The result will be learned HMM parameters: the state transitions  $A = (a_{ij})$  and emissions  $= (b_{jk})$ .

$c^*$  is the best evaluated configuration; it is given by the formula :

$$c^* = \max_j (b_{j1}) \quad (11)$$

The best configuration  $c^*$  is the configuration that has the maximum probability to emit rank 1. Especially, it is the most likely configuration to be the best over all instances.

#### 4.2.2 Performance evaluation of HMM-based tuner

By adapting the formulation of the HMM-based tuner to some defined problem we can evaluate the effectiveness of our used tuning method. Therefore, we depict some applications for air transport problem in the second part of this thesis, which is dedicated to this issue. PSO algorithm is used to solve each model of air transport with related experimentations to prove the efficiency of the proposed HMM tuner.

### 4.3 AUTOMATED PARAMETER CONTROL BY HMM

The parameter control mechanism has many advantages. It can enable to auto-select, one of the appropriate parameter values in different stages of the search process, to make use of the accumulating information to enhance performance in later stages, and to liberate the user from the charge of selecting parameter values [39]. In this section, a new generic model for parameter control will be defined according to the analysis of the PSO behavior search. Then, the proposed control scheme is applied to different PSO parameters with different PSO search structure. The two cases of PSO structure will be liable to the generic model: homogeneous and heterogeneous PSO.

#### 4.3.1 A generic model

##### *a. Markov Chain on Particle Swarm Optimizer*

As a complex stochastic process, The Particle Swarm Optimizer (PSO) can be analyzed according to its stochastic behavior. When it comes to this type of investigation, the majority of the important research on PSO is centered on simulations and empirical study and only a little is determined by theoretical methodology. In this paragraph, theoretical results from [6] are described to inspect the stochastic behavior of PSO.

We consider a swarm of  $N$  particles on a space  $S$ , and  $i$  is the index of particle and  $t$  the index of iteration;  $X_i$  represents the position and velocity vector of particle  $i$ ,  $Pbest_i$  the personal best position of particle  $i$ , and  $gbest$  denotes the global best position.

The state of the PSO defined by [7] includes as many details as possible that are involved in the process. PSO state has a memory-less property which is also proven by [6]. The state at time  $t$  is defined as :

$$W(t) = (X(t), pbest(t), V(t), gbest(t) ) \quad (12)$$

The state Markov chain identified of PSO states is stationary (time homogeneous).  $W(t)$  has the information required for the future movements, and varies depending just on the present state. The influence of the present state on the next states is independent of the historical state. The execution of PSO is based only on the current state without including the past history.

Furthermore, to assess the achievement (good results) of the state  $W$ , an index for the state to represent the actual achievement of PSO is additionally defined in [6]. The achievement can only be position-dependent. This index is strongly related to the positions and depending on a concept of probability because of stochastic movements of particles. Likewise, it can be defined by different measures based on particles positions (because that reflects the achievements). Since the already defined states  $W$  are merged into a countable number of classes that refers to its achievement, they are identified as levels. The stochastic process of the levels is defined as :

$$\{L(W(t)), t = 1, 2 \dots\} \quad (13)$$

$L(t)$  constitutes also a Markov chain on PSO levels [6]. Then, it constitutes also a stationary Markov chain. As a result of this state classification, as  $t$  raises, the state may earn a significantly greater achievement, but it is not guaranteed. If for any particular sample process, as  $t$  raises while the state can't get any noticeable achievement. Therefore, the future state distribution will not likely be favored even if  $t$  is highly larger. Consequently, it appears enhanced to consider different parameters selection during different state stages rather than time stage as the earlier research managed. The transition probability matrix can be denoted by:

$$P \equiv [p_{j,k}], \quad j < m, k < n \quad (14)$$

The fact that the state (level) transition probabilities are independent of  $t$  justifies the fact that parameter selection needs to be a function of state (representing achievement) rather than a function of iteration.

Based on the theoretical results of [6] and [7], the further extension which is carried out in this thesis could be the analysis of the relationship between transition probability and parameter set estimating the transition probability with the selection of the optimum parameter set in each state. An efficient instrument could be making use of computer simulations. This approach can maximize the value of adaptive parameters selection rather than classical approaches which are based on the iteration number. This will likely constitute a preliminary basis to build in the next paragraph a generic model for adapting PSO parameters.

*b. A model for adaptive parameters*

Based on the theoretical results of [6] presented above, it exists a Markov chain on PSO levels. These levels defined by the degree of achievements of the search which must be defined related to the position of particles. We define in this section parameters adaptation in the light of the last paragraph and the results given by the APSO [60]. Some notions are presented before we address our parameters adaptation control by HMM.

This proposed method uses the same global topology as the standard PSO algorithm. As recognized, this topology converges quickly; nevertheless, it can simply be trapped into local optima. While PSO, which has a local topology, it has more chances to find the global optimum, but by a slow convergence speed [91]. Consequently, global and local variations of PSO algorithms both possess their advantages and disadvantages. Considering these matters, a global topology is designed with an adaptive parameter control of PSO by HMM classification in order to improve the overall performance of the algorithm. Thus, the developed approach based on HMM aimed to possess effective global search capability using the associated adaptation strategy, such as local versions of PSO, preserving good convergence speed as it is a global variant of PSO.

c. *PSO hidden states*

To define our approach for classification of PSO states using a hidden Markov model, we inspire from the same definition of particle states in APSO, and we use classification capabilities of HMM to enhance the PSO algorithm.

HMM is a stochastic method where we need to associate several probabilities in the model definition; Transitions between states are governed by a set of probabilities named transition probabilities. In a particular state, an observation can be generated according to the associated probability distribution. These observations are visible in contrast with their corresponding states which are hidden. After model parameters definition, a resolution algorithm is used to build the HMM classification process.

The contribution of using HMM inside PSO is to benefit from these features to better explain PSO behavior and then to better identify the corresponding state at each iteration. Indeed, HMM can enable most robust estimation of the state based on the evolutionary factor than the fuzzy classification (HMM can give better results as shown in [92] even if it is more computationally expensive). That is, in HMM, we can predict from the evolutionary factors (which correspond to observations in HMM) the most probable of the four states (which are the hidden states of HMM).

We follow a similar state definition as the one proposed by [60] which is also used by [58] to enhance PSO adaptivity. It consists of identifying one of four evolutionary states: exploration, exploitation, convergence, and jumping out in order to enable the automatic control of acceleration coefficients. The objective is to use HMM to identify the proper state (class) at each iteration. So, we can generate the Markov Chain of PSO states as described in figure 11.

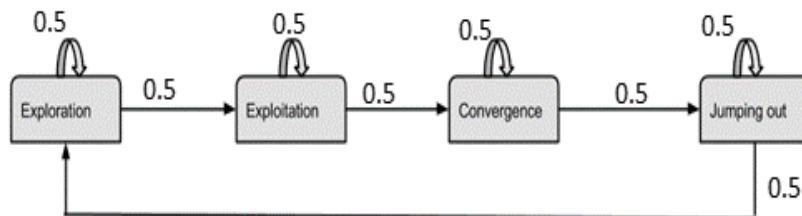


Figure 11 Markov Chain of PSO states

PSO parameters are updated according to the classified state at every iteration. Then, the same as [60] an elitist learning strategy is performed when the evolutionary state is classified as a convergence state. We take all possible  $i$  and  $j$  transitions as mentioned in figure 11 a transition probability of 0.5.

The definition of states (achievement level) is controlled by the value of quantified population distribution information, called in [60] evolutionary factor  $f$  that is calculated at each iteration. Consider the mean distance of each particle  $i$  ( $1 < i < N$ ) to all the other particles as  $d_i$ , and calculate:

$$f = \left| \frac{d_{gbest} - d_{min}}{d_{max} - d_{min}} \right| \quad (15)$$

where:  $gbest$  is the global best particle and

$$d_i = \frac{1}{N-1} \sum_{j=1, j \neq i}^N \sqrt{\sum_{k=1}^D (x_i^k - x_j^k)^2}, d_{max} = \max_{1 < i < N} (d_i), d_{min} = \min_{1 < i < N} (d_i)$$

The evolutionary factor will be used to define adaptation strategy by HMM in the next paragraph, adopted from [60] because it reflects the relative position of gBest to the other particles. Its value is relatively large within the jumping out and exploration state (because particles are more distant). Then, it becomes smaller in the exploration and smaller in the convergence state (particles are close to each other around gbest).

The relation between states or strategies and values of the factor  $f$  is established by a defuzzification technique (see figure 12)([60]). According to it, PSO parameters are adjusted. It has a major role in the convergence across iterations.

On the other hand, defuzzification used in APSO takes the only value of the factor  $f$  in consideration without any use of any past information about previous executed iterations. The adopted method consist of replacing the defuzzification in APSO by HMM classification, which uses in addition to the value of  $f$ , all past executions of iterations to classify and define particle state and then the corresponding strategy. This will be described in the following paragraph.

#### d. Model of state classification

In this generic model, we use the probabilistic machine learning method by Hidden Markov Chain (HMM). That is, HMM is used to have stochastic state control of PSO at each iteration. The main idea is to assign state selection (defined in the previous paragraph) inside the particle swarm optimization to HMM. This process is performed by the Viterbi algorithm that gives the most probable path of states in each PSO iteration. Also, HMM parameters are calculated and updated at each iteration according to the change in particle environment.

Therefore, We define the Hidden Markov Model by a triple  $\lambda = (\Pi, A, B)$ , all processes are defined on a probability space.

- $\Pi = (\pi_i)$  The vector of the initial probability distribution over states;
- $A=(a_{ij})$  The state transition matrix,  $P(S_t = i | S_{t-1} = j), i, j \in [1, N]$
- $B = (b_{jk})$  The emission matrix also called the confusion matrix,  $P(Y_t = k | S_t = j), j \in [0, N], k \in [0, M]$ .

The set of  $N$  states  $\{q_t\}_{t \in \mathbb{N}}$  takes values from the set  $S = \{S_i\}_{i \in [1,4]}$ , which references respectively: exploration, exploitation, convergence, and jumping out. The change of state is reflected by the PSO state sequence  $Q = q_1 q_2 \dots q_T$  for example :  $(q_1 = S_2) \Rightarrow (q_2 = S_1) \Rightarrow (q_3 = S_2) \Rightarrow \dots$ , as deduced by [60]), corresponding to the Markov Chain in figure 11.

Furthermore, we define corresponding initial transition probabilities,  $P(S_t = i | S_{t-1} = j), i, j \in [1,4]$ . This probability controls all behavior of transition between states of

APSO resolution. The initial state probability corresponds to deterministic start in exploration state:

$$\Pi = (\pi_i) = [1 \ 0 \ 0 \ 0] \quad (16)$$

The observed parameter of this hidden chain is the evolutionary factor  $f$  (defined in [60]) of the APSO as depicted in figure 12.

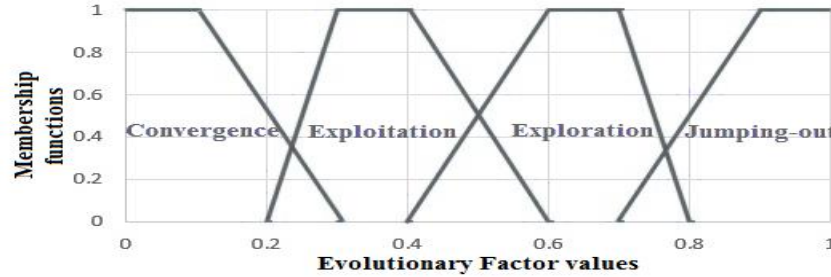


Figure 12 Fuzzy functions of evolutionary factor by particles state [60]

Observations will be described by the  $f$  value membership to subintervals of  $[0,1]$  ( $[0,0.2]$ ,  $[0.2,0.3]$ ,  $[0.3,0.4]$ ,  $[0.4,0.6]$ ,  $[0.6,0.7]$ ,  $[0.7,0.8]$ ,  $[0.8,1]$ ). We divide  $[0,1]$  to seven subintervals, so the set observation  $Y = \{y_i\}_{i \in [1, \dots, 7]}$  will be the number of the subinterval which belong  $f$ . Let  $sub: [0,1] \rightarrow \{1, 2, \dots, 7\}$  the function that returns the corresponding interval of  $f$ ; it corresponds also to the observation.

$$sub(f) = \delta_{[0,0.2]}(f) + 2\delta_{[0.2,0.3]}(f) + 3\delta_{[0.3,0.4]}(f) + 4\delta_{[0.4,0.6]}(f) + 5\delta_{[0.6,0.7]}(f) + 6\delta_{[0.7,0.8]}(f) + 7\delta_{[0.8,1]}(f) \quad (17)$$

$$(with \ \delta_{[a,b]}(x) = \begin{cases} 1, & x \in [a, b] \\ 0 & otherwise \end{cases} \quad a, b \in \mathbb{N}, x \in \mathbb{R})$$

Emission probabilities are deduced from the defuzzification process (figure 12) of [60] as follows:

$$P = \begin{bmatrix} 0 & 0 & 0 & 0.5 & 0.25 & 0.25 & 0 \\ 0 & 0.25 & 0.25 & 0.5 & 0 & 0 & 0 \\ 2/3 & 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/3 & 2/3 \end{bmatrix} \quad (18)$$

$P$  represents in another way the same probabilities of the defuzzification technique. More detailed analysis of the relationship between strategies and the diversity of particles can be found in [60]).

After initializing HMM parameter, Baum-welch algorithm (algorithm 3) is used at each iteration to estimate and update HMM emission and transition matrix; this allows HMM to be more adaptive and accurate in the classification step.



**Algorithm 3: Baum-welch algorithm ([79])**

**Data:**  $\Pi, O, A, B, S, Y$   
 Initialisation of forward[4,T] and backward[4,T] matrix  
**repeat**  
**for**  $i = 1$  to 4 **do**  
   forward[ $S_i, 1$ ]  $\rightarrow \pi_i \times b_{i,1}$ ;  
**end**  
**for**  $t = 2$  to T **do**  
   **for**  $i = 1$  to 4 **do**  
     forward[ $S_i, t$ ]  $\rightarrow \sum_{k=1, \dots, 4} \text{forward}[i, t-1] \times a_{k,i} \times b_{i,y_t}$   
   **end**  
**end**  
   forward[ $Y_t, T$ ]  $\rightarrow \sum_{k=1, \dots, 4} \text{forward}[k, T]$   
   backward[ $Y_t, T$ ]  $\rightarrow \sum_{k=1, \dots, 4} \text{backward}[k, T]$   
**for**  $t = 1$  to T **do**  
   **for**  $i = 1$  to 4 **do**  
     backward[ $S_t, t$ ]  $\rightarrow \sum_{k=1, \dots, 4} \text{backward}[i, t-1] \times a_{k,i} \times b_{i,y_t}$   
   **end**  
**end**  
**for**  $t = 1$  to T **do**  
   **for**  $i = 1$  to 4 **do**  
     **for**  $j = 1$  to 4 **do**  
        $\xi_{i,j}(t) = \frac{\text{forward}[i,t] \times \text{backward}[i,t] \times b_{i,y_t}}{P(O/(A,B,\pi))}$   
     **end**  
   **end**  
 $\lambda_i(t) = \frac{\text{forward}[i,t] \times \text{backward}[s,t]}{P(O/(A,B,\pi))}$   
**end**  
   **for**  $i = 1$  to 4 **do**  
     **for**  $j = 1$  to 4 **do**  
        $\pi_i = \lambda_1(t), a_{ij} = \frac{\sum_{t=1, \dots, T-1} \xi_{i,j}(t)}{\lambda_i(t)}, b_{ik} = \frac{\sum_{t=1, \dots, T} \xi_{i,j}(t)}{\sum_{t=1, \dots, T} \lambda_i(t)}$   
     **end**  
   **end**  
**until** no increase of  $P(O/\lambda)$  or no more iterations are possible  
**Result:**  $(\pi, A, B)$

Then, the Viterbi Algorithm is used with the estimated parameters to find the most probable sequence associated with hidden states given a sequence of observed states. The Viterbi algorithm does not only accept the most likely state for a given time instant but also takes a decision based on the whole observation sequence. The algorithm will find the corresponding Q (state sequence  $Q = q_1 q_2 \dots q_T$ ) for a given observation sequence ( $O = o_1 o_2 \dots o_T$ ) by means of induction (t the iteration number). It is about to find the highest probability paths for states ([79]). Viterbi algorithm is given in Algorithm 4 below.

**Algorithm 4 : Viterbi algorithm [79]**

```

Data:  $\Pi, O, A, B, S, Y, T$ 
Initialisation : matrix of Viterbi[4,T]
for  $i = 1$  to 4 do
    Viterbi[ $i, 1$ ]  $\rightarrow \pi_i \times b_{i,1}$ ;
    State[ $i, 1$ ]  $\rightarrow X_1$ ;
end
for  $t = 2$  to T do
    for  $i = 1$  to 4 do
        Viterbi[ $i, t$ ]  $\rightarrow \max_{s'=1,\dots,4} \text{Viterbi}[k, t-1] \times a_{k,i} \times b_{i,y_t}$ 
        State[ $i, t$ ]  $\rightarrow \text{argmax}_{s'=1,\dots,4} \text{Viterbi}[k, t-1] \times a_{k,i}$ ;
    end
end
 $Y_T \rightarrow \text{argmax}_{k=1,\dots,4} \text{Viterbi}[k, T]$  ;
 $X_T \rightarrow X_{Y_T}$ 
for  $t = T, T-1, \dots, 2$  do
     $Y_{t-1} = \text{State}[Y_t, t]$ 
end
Result: The state sequence  $Q_z = (S_{Y_1}, \dots, S_{Y_T})$ 

```

*e. Online parameters estimation*

Additional online learning for HMM parameters can be integrated into the proposed model by online Expectation-Maximization algorithm instead of the batch learning given by Baum-welch algorithm. Also, HMM parameters are calculated and updated at each iteration according to the change in particles environment. The online Expectation-Maximization algorithm brings continuous parameters update for the proposed model.

Considering Online HMM training, this was defined by [93] and [94]. An HMM that use online learning can independently learn from a new block of data at a time. So, HMM parameters should be efficiently updated from new data without requiring access to all training data. In addition, parameters are re-estimated online upon observing each new sub-sequence [95]. The online EM algorithm for HMM allows continuous adaptation of HMM parameters along a potentially infinite observation stream.

An online EM learning is first performed at each iteration to calculate and update HMM parameters that are re-estimated upon observing each new sub-sequence.

Particles positions and velocities vary over iterations, also impacting the evolutionary factor. Then, the classification environment for HMM changes during operations. Online learning of new data sequences allows adapting HMM parameters as new data becomes available as shown in figure 13 where  $(D_i)_{i \in [0,n]}$  are data observations and  $(\lambda_i)_{i \in [0,n]}$  parameters values updates.

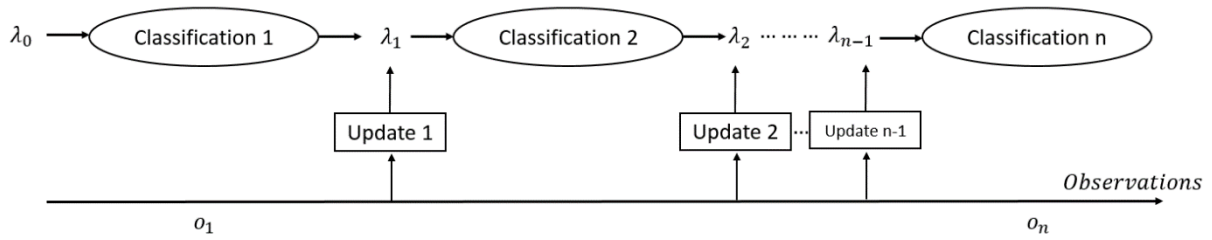


Figure 13 HMM classifications

At each iteration  $t$ , a new classifier is performed with new updated parameters. We choose online learning EM algorithm [95] instead of Batch learning (classical Baum-Welch algorithm [96]), because this last one needs to run one all observation sequence where this is not our case.

The online Expectation-Maximization algorithm (algorithm 5) used for HMM parameters learning is described as follow:

**Algorithm 5 :** Expectations-Maximization algorithm

**Input :** observation sequence  $O = (o_1 o_2 \dots o_T)$ , states  $S$ , initial parameters set  $\lambda_0$   
**Output :** Estimated parameters  $\lambda$   
**For**  $t = 1 \dots N_{step}$  **do**  
**E-step** - find conditionally optimal hidden trace :  
 $S_i = \text{argmax}_s P(O|S, \lambda_{i-1})$   
 Compute likelihood:  $L^{i-1}(\lambda) = P(O|S, \lambda_{i-1})$   
**If**  $i < N_{step}$  and likelihood not yet converged:  
**M-step** - find conditionally optimal parameter set  $\lambda$ :  
 $\lambda_i = \text{argmax}_\lambda P(O|S_i, \lambda)$   
**Return** optimal parameter set  $\lambda$

This Algorithm is based on an online version of the Expectation-Maximization algorithm: the algorithm involves a stochastic version of the E-step and M-step to include the information through the newly existing observation. This online version of EM does not require the whole data set to be made available at each iteration. Regarding simulations that apply tested observations on experimental data [93], the performance of this algorithm is better than the batch algorithm.

### 4.3.2 Parameter control of Homogeneous PSO

By integrating the identification of the suitable achievement level of the search process (hidden state), PSO parameter adaptation will be more suitable if it is adjusted according to the classified PSO hidden state. The application of this model needs to analyze first the relationship between different parameter sets estimating the transition probability with the selection of the optimum parameter set in each state. This defined model of hidden PSO states and the method of classification and computing the related transition probabilities will be investigated for application in PSO for different parameters in the case of homogeneous PSO structure in this chapter.

We implement the generic model defined in the previous paragraph in the case of Homogeneous PSO, also referred to as mono swarm. The majority of PSO variants [38] use homogeneous swarms; every one of the particles assumes to use exactly the same behavior applying the same velocity and position update rule at the same iteration. The improvement of PSO will concentrate on parameter control of PSO that all particles use of the same parameter at each iteration. So that the implementation will concentrate on the definition of the relationship between the PSO parameter set to be tuned and the classified PSO hidden state. An empirical study can be the best choice to identify this relationship.

*a. Control of acceleration coefficients*

**a.1 Coefficients update by state**

The HMM classification in the last paragraph will define, at each iteration, one of the four swarm states: exploration, exploitation, convergence, and jumping out. In contrast with APSO, HMM perform a stochastic state classification that takes into account all state history change across iterations (state sequence  $\underline{Q}$ ). Therefore, the HMM classification will be more accurate than the fuzzy logic used by APSO [97]. Furthermore, According to the classified state, PSO parameters are controlled and updated.

Acceleration coefficients update by state (Algorithm 6) is done by a benchmark between increasing or decreasing  $c_1$  and  $c_2$  according to the following notion as also given by [60].  $c_1$  is the cognitive component that measured the degree of self-confidence of a particle and measures the degree at which it trusts its performance.  $c_2$  is the social component that relies on the capability of the swarm to find better candidate solutions.

The increase of  $c_1$  promotes exploration of local regions and maintaining the diversity of the swarm.  $c_2$  increases contrariwise helps the swarm to converge towards the current global best region. So, decreasing  $c_1$  or  $c_2$  gives its opposite influence. From this fact, we can conclude the algorithm 6, which is inspired by the APSO approach of adapting  $c_1$  and  $c_2$  factors to the population state/strategy.

**Algorithm 6:** Update by state of acceleration coefficients[60]

```

Data: Position and acceleration factors
Initialization: positions and acceleration factors  $c_1$  and  $c_2$ ;
if state = exploration then Increasing  $c_1$  and Decreasing  $c_2$  ;
else if state = exploitation then
    Increasing  $c_1$  and Slightly Decreasing  $c_2$ 
else if state = jumping out then
    Increasing Slightly  $c_1$  and Increasing  $c_2$ 
else if state = convergence then
    Decreasing  $c_1$  and Increasing  $c_2$ 
end if
Return  $c_1$  and  $c_2$ 

```

The idea behind exploration state is that increasing  $c_1$  and decreasing  $c_2$  can help particles explore individually and enhance their own historical best positions, rather

than crowd around the current best particle that may be associated with a local optimum.

In exploitation state, the particles are making use of local information and grouping toward possible local optimal niches indicated by the historical best position of each particle.

In the convergence state, the swarm seems to find the globally optimal region, and, therefore, the influence of  $c_2$  should be emphasized to lead other particles to the estimated globally optimal region. On the other hand, the value of  $c_1$  should be decreased to let the swarm converge fast.

In jumping out phase, a large  $c_2$  together with a relatively small  $c_1$  may help to jump out of local optimum toward a better optimum.

### **a.2 HMM adaptation of acceleration factors (HMM-APSO)**

In this paragraph only acceleration coefficients are controlled by HMM state, other parameters are static and given by the user at initialization except for the inertia weight, it is done as follows to balance the global and local search capabilities:

$$w(f) = \frac{1}{1 + 1.5e^{-2.6f}} \in [0.4, 0.9] \forall f \in [0, 1] \quad (19)$$

By the HMM-APSO (HMM Adaptive PSO), HMM adaptively controls the variation of both  $(c_1, c_2)$ . The HMM-APSO has been introduced in [97]. In this algorithm, we delegate choosing states of PSO iterations to HMM classification. Transitions between states is explicated with probability transitions with observations through the algorithm simulation. The Baum welch algorithm re-estimates HMM parameters for best classification of states. The Viterbi Algorithm is then used for state classification of APSO iteration. It is designed to find the most probable sequence of hidden states given the sequence of observed states. An elitist learning strategy is performed when the evolutionary state is classified as a convergence state to perturb the particle that guides the swarm and to enhance the local convergence [60]. According to this classified state, the control of the variation of both  $(c_1, c_2)$  is adapted. The algorithm of HMM-APSO is given below:

**Algorithm 7: HMM-APSO**

```

Data: The objective function (F)
Initialization: iteration number  $t=0$ , positions  $X$  velocities  $V$ ,
acceleration factors  $(c_1, c_2)$ , HMM parameters  $(\Pi, A, B, S, Y)$ , observation
sequence  $O$ , state sequence  $Q$ , population size  $N(t)$ ;
while (number of iterations  $t \leq t_{\max}$  not met) do
 $f \leftarrow \text{equation.15}(X)$ ; (Calculate the evolutionary factor)
 $w \leftarrow \text{equation.19}(f)$ ; (Update the inertia weight)
 $O[t] \leftarrow \text{sub}(f)$  (Update observation sequence)
 $O \leftarrow O \cup O[t]$ 
 $(A, B, \Pi) \leftarrow \text{Baum-Welch}(\Pi, O, A, B, S, Y)$ ; (Update HMM parameters)
 $(Q[1], \dots, Q[t]) \leftarrow \text{viterbi}(\Pi, O, A, B, S, Y)$ ; (Classification of PSO state by HMM
classifier)
 $(c_1, c_2) \leftarrow \text{algorithm.6}(Q[t])$ ; (Update  $c_1$  and  $c_2$  values)
For  $i = 1$  to  $(N(t))$  do
  Update velocities and positions:
    -  $X_i(t) \leftarrow \text{equation.1}(X_i(t-1), V_i(t-1))$ ;
    -  $V_i(t) \leftarrow \text{equation.2}(X_i(t-1), V_i(t-1))$ ;
  compute  $f(X_i)$ ;
  if ( $f(X_i) \leq f_{\text{best}}$ ) then
     $f_{\text{best}} \leftarrow f(X_i)$ ;
     $p_{\text{best}} \leftarrow X_i$ ;
  end
  if ( $f(p_{\text{best}}) \leq f_{g\text{best}}$ ) then
     $f_{g\text{best}} \leftarrow f_{\text{best}}$ ;
     $g_{\text{best}} \leftarrow X_{\text{best}}$ ;
  end
  if state = convergence then
    Elitist learning ([60]);
  end
end
 $t \leftarrow t + 1$ ;
end
Return  $p_{\text{best}}$  and  $f_{\text{best}}$  (the best particle in the population and its
corresponding fitness)

```

HMM-APSO performances will be investigated by experimentations to prove the HMM classification priority in parameters control.

In the next section, we will introduce another parameter to be controlled, the population size control by HMM state classification.

### *b. Control of Population Size*

As discussed above, HMM-APSO is influenced by population variation. So, if HMM control parameters such as  $c_1$  and  $c_2$  across iterations, it can also be used to control the number of particles. More precisely the manner how population changes from iteration to another is based on the state that HMM classification defines.

#### **b.1 Population update strategy by state**

As deduced from the first section, smaller population size will increase the probability of being trapped in a local optimum. However, added particles result in the rise of

computing cost. Therefore, better optimization capability is not necessarily gained with population increase. More profit on optimization performances can be acquired with a dynamic change and continuous variation like in [17] and [44]. Therefore, our primary purpose is to adapt a dynamic variation of population change according to the classified HMM state.

Population size variation is done between two value  $N_{min}$  and  $N_{max}$  in  $T$  total number of iterations. Population size is changed according to a changing function  $N(t)$ , where  $t$  is the iteration number. We define four functions for varying population size (see F).

- Linear increase function :

$$N_a(t) = \frac{(N_{max} - N_{min})(t - T)}{T} + N_{max} \quad (20)$$

- Linear decrease function :

$$N_b(t) = \frac{(N_{max} - N_{min})(T - t)}{T} + N_{min} \quad (21)$$

- Sine function :

$$N_c(t) = (N_{max} - N_{min}) \cdot \frac{\sin\left(\frac{t}{A}\right)}{2} + \frac{N_{max} + N_{min}}{2} + N_{min} \quad (22)$$

- Sine Attenuation function :

$$N_d(t) = N_b(t) \cdot \frac{\sin\left(\frac{t}{A}\right)}{3} + \frac{2N_b(t)}{3} \quad (23)$$

These functions are inspired from [17];  $A$  is the amplitude of the sin function.

After state classification by HMM, a population size adaptation is executed according to the given state at each iteration. Then, to each state, a varying population function is associated to adapt the variation to the state of particles according to the algorithm 8 below to resume the use of size function by state:

**Algorithm 8:** Population size control

```

Data: state, iteration number t, last population size N(t-1);
Initialization: population size ;
if state=exploration then population size  $\leftarrow N_c(t)$  (Equation (22));
else if state = exploitation then
    N(t)  $\leftarrow N_d(t)$  (Equation (23) )
else if state = jumping out then
    N(t)  $\leftarrow N_a(t)$  function (Equation (20) )
end
else if state = convergence then
    N(t)  $\leftarrow N_b(t)$  (Equation (21) )
end
Return population size N(t)

```

The following specifications designated in Algorithm 8 by state describes PSO size adaptation:

- In the exploration state: The Sine function  $N_c$  is adopted to fluctuate the swarm periodically; this helps to re-initialize the particles again, which improves the swarm diversity for exploration.
- In the exploitation state: The Sine Attenuation function  $N_d$  is used to also fluctuate the swarm periodically but with reducing the number of worst particles; this helps to gain more in exploiting capacity.
- In the jumping out state: The Linear increase function  $N_a$  is adopted to increase jump capabilities with a random increase of population to be more distributed in the search space.
- In the convergence state: The Linear decrease function  $N_a$  is adopted to increase convergence capabilities. Worst particles are continuously eliminated without any generation to focus the search aptitude on converging.

## b.2 Generation and elimination of particles

As a result of applying the population size control, particles are even eliminated or generated according to the value of comparing  $N(t-1)$  against  $N(t)$ . Elimination and generation of particles is not new; it has been presented in [43] where it addresses some rules about how to eliminate and how to generate particles inside tribes of particles. In our approach, we do not have any notion of separation of particles and we use a global topology. Unfortunately, the swarm changes its state according to HMM. Then, a strategy for elimination and generation must be done according to this state. In [43], the worst particles are eliminated in favor of good tribes. This rule is adopted in any case of elimination in this approach. We present how particle generation is done in algorithm 9:

### Algorithm 9: Generation and elimination strategy

```

Data: Population  $(X,V)$ , iteration number  $t$ ;
Initialization: population and velocities
if  $(N(t) > N(t-1))$  (generation)
if state = convergence or state = exploitation then;
    for  $i=N(t-1)$  to  $N(t)$ 
         $X_i \leftarrow x_{gBest}$  (Copy the best particle's position).
         $V_i \leftarrow random$  (random velocity).
    end
else if state = exploration or state = jumping out then
    for  $i=N(t-1)$  to  $N(t)$ 
         $X_i \leftarrow random$  (random position).
         $V_i \leftarrow random$  (random velocity).
    end
end
else if (elimination)
    population  $\leftarrow$  Quick sort (population) [98]
    population  $\leftarrow$  the best  $N(t)$  particles.
end
Return updated population  $(X,V)$ 

```



Particles are ordered according to their fitness value from the worst to the best particle using Quick sort algorithm [98]. The population is ordered in a pile of particles and the operation of decreasing is done from the worst side of the pile. So, in case of population decrease, always the worst particles are eliminated.

For population increase, we choose between two types of strategies of generation: random or duplication are defined. A random one is the easy way as in [43]: according to a uniform distribution in the search space with help to explore more regions. Contrariwise, the duplication method contributes to gain more in exploitation because it increases the density of exploiting particles in a specific region.

### **b.3 HMM control of Population size (HMM-PPSO)**

To do that, as given by the generic model, transitions between states are represented by transitions probabilities and the Viterbi Algorithm is used for state classification of APSO iteration. We update the population according to the classified state. The Baum-welch algorithm updates transition probabilities. We note HMM-PPSO (HMM Population control for PSO) as the same HMM control strategy applied only for population variation without  $c_1$  and  $c_2$  adaptation. It has been taken as a static value during the run. In addition to HMM classification, the same elitist learning strategy as in the previous paragraph is performed when the evolutionary state is classified as a convergence state [60]. According to the classified HMM state, the control of the variation of the population is adapted to meet the best population size for the search stage through iterations. The inertia weight is also updated according to equation 19. The complete HMM-PPSO is depicted below (Algorithm 10):

**Algorithm 10: HMM-PPSO**

```

Data: The objective function (F)
Initialization: iteration number  $t=0$ , positions  $X$ , velocities  $V$ , acceleration
factors  $(c_1, c_2)$ , HMM parameters  $(\Pi, A, B, S, Y)$ , observation sequence  $O$ ,
state sequence  $Q$ , population size  $N(t)$ ;
while (number of iterations  $t \leq t_{\max}$  not met) do
     $w \leftarrow$  equation.19 ( $f$ ) ; (Update the inertia weight)
     $O[t] \leftarrow \text{sub}(f)$  (Update observation sequence)  $O \leftarrow O \cup O[t]$ 
     $(A, B, \Pi) \leftarrow$  Baum-Welch( $\Pi, O, A, B, \Pi, S, Y$ ) ; (Update HMM parameters)
     $(Q[1], \dots, Q[t]) \leftarrow$  viterbi( $\Pi, O, A, B, S, Y$ ) ; (Classification of PSO state
    by HMM classifier)
     $(N(t)) \leftarrow$  algorithm.8( $Q[t], t, N(t-1)$ ) ; (Calculate population size)
     $(X, V) \leftarrow$  algorithm.9( $N(t), X, V, t$ ) ; (Update population)
    For  $i = 1$  to  $(N(t))$  do
        Update velocities and positions:
            -  $X_i(t) \leftarrow$  equation.1( $X_i(t-1), V_i(t-1)$ ) ;
            -  $V_i(t) \leftarrow$  equation.2( $X_i(t-1), V_i(t-1)$ ) ;
        compute  $f(X_i)$  ;
        if ( $f(X_i) \leq f_{\text{best}}$ ) then
             $f_{\text{best}} \leftarrow f(X_i)$  ;
             $p_{\text{best}} \leftarrow X_i$  ;
        end
        if ( $f(p_{\text{best}}) \leq f_{\text{gbest}}$ ) then
             $f_{\text{gbest}} \leftarrow f_{\text{best}}$  ;
             $g_{\text{best}} \leftarrow X_{\text{best}}$  ;
        end
        if state = convergence then
            Elitist learning [60];
        end
    end
     $t \leftarrow t + 1$  ;
end
Return  $p_{\text{best}}$  and  $f_{\text{best}}$  (the best particle in the population and its
corresponding fitness)

```

*c. Control of inertia weight*

The balance between global and local search throughout a run is crucial for the success of an optimization algorithm. Inertia weight is a crucial parameter influencing the PSO search process. In consequence, according to the state, the inertia weight is controlled. In this paragraph, a humble attempt to determine a generalized framework for the

setting of the inertia weight adaptation and control is depicted and named HMM-wPSO .

### c.1 Inertia weight control by state

The following specifications by state describe PSO inertia weight adaptation according to the previous HMM classification:

- In the exploration state: Particles are exploring the search space; random values of inertia weight are attributed at each iteration to guarantee more diversification in search space sweep. The value of the inertia weight is given by the formula :

$$w = w_{min} + (w_{max} - w_{min}) * rand() \quad (24)$$

Where :

$rand()$  : function that returns random values in the interval  $[0,1]$ .

- In the exploitation state: in this state, inertia weight will vary according to the distance between particles to guide the exploitation phase. The same formula defined in APSO is used just for exploitation:

$$\omega(f) = \frac{1}{1 + 1.5e^{-2.6f}} \in [0.4, 0.9] \forall f \in [0, 1] \quad (25)$$

- In the jumping out state: a maximum value of  $w$  is used to increase jump capabilities in the search space:

$$w = w_{max} \quad (26)$$

- In convergence state: To increase convergence capabilities, only local search capabilities are enabled. Then, inertia weight is set to the minimum value :

$$w = w_{min} \quad (27)$$

Elastic learning as in [60] is also used in the convergence state.

Algorithm 11 summarizes the adaptation approach of  $w$ .

#### Algorithm 11: Adaptive inertia weight control

```

Data: Position and inertia weight
Initialization: positions and weight  $w$ 
if state = exploration then
     $w = w_{min} + (w_{max} - w_{min}) * rand();$ 
else if state = exploitation then
     $w = \omega(f) = \frac{1}{1 + 1.5e^{-2.6f}} \in [0.4, 0.9] \forall f \in [0, 1]$ 
else if state = jumping out then
     $w = w_{max}$ 
else if state = convergence then
     $w = w_{min}$ 
end if
Return  $w$ 

```

### c.2 HMM control of inertia (HMM-wPSO)

Our HMM-wPSO algorithm of HMM control of inertia weight will be given in algorithm 12:

**Algorithm 12: HMM-wPSO**

```

Data: The objective function (F)
Initialization: iteration number  $t=0$ , positions  $X$ , velocities  $V$ , inertia weight  $w$  and HMM parameters  $(\Pi, A, B, S, Y)$ , observation sequence  $O$ , state sequence  $Q$ , population size  $N$ ;
while (number of iterations  $t \leq t_{\max}$  not met) do
   $f \leftarrow \text{equation.15}(X)$ ; (Calculate the evolutionary factor)
   $w \leftarrow \text{equation.19}(f)$ ; (Update the inertia weight)
   $O[t] \leftarrow \text{sub}(f)$  (Update observation sequence)
   $O \leftarrow O \cup O[t]$ 
   $(A, B, \Pi) \leftarrow \text{Baum-Welch}(\Pi, O, A, B, S, Y)$ ; (Update HMM parameters)
   $(Q[1], \dots, Q[t]) \leftarrow \text{viterbi}(\Pi, O, A, B, S, Y)$ ; (Classification of PSO state by HMM classifier)
   $w \leftarrow \text{algorithm.11}(Q[t])$ ; (Update  $w$  value)
  For  $i = 1$  to  $N$  do
    Update velocities and positions:
      -  $X_i(t) \leftarrow \text{equation.1}(X_i(t-1), V_i(t-1))$ ;
      -  $V_i(t) \leftarrow \text{equation.2}(X_i(t-1), V_i(t-1))$ ;
    compute  $f(X_i)$ ;
    if ( $f(X_i) \leq f_{\text{best}}$ ) then
       $f_{\text{best}} \leftarrow f(X_i)$ ;
       $p_{\text{best}} \leftarrow X_i$ ;
    end
    if ( $f(p_{\text{best}}) \leq f_{g\text{best}}$ ) then
       $f_{g\text{best}} \leftarrow f_{\text{best}}$ ;
       $g_{\text{best}} \leftarrow p_{\text{best}}$ ;
    end
    if state = convergence then
      Elitist learning [60];
    end
  end
   $t \leftarrow t + 1$ ;
end
Return  $p_{\text{best}}$  and  $f_{\text{best}}$  (the best particle in the population and its corresponding fitness)

```

HMM-wPSO gives an online adaptation of inertia  $w$  in this paragraph with a machine learning technique; that is the HMM classification.

#### *d. Empirical evaluation*

In this part, tests and validations of the proposed adaptive PSO algorithms based on HMM for adaptation and parameters control are performed. Experimentations are done using several benchmark functions by comparing our generic model of PSO to each parameter adaptation with other PSO's variants from the literature.

##### **d.1 Parameters setting**

For each of the benchmark functions shown in Table 1, we perform thirty executions, and compare each function, the best and the average value.

The used experimentation machine has i5 processor third generation of 2.5 GHz, with 4 Gb of RAM and 128 Gb of storage.

Table 1 Description of Benchmark functions

Benchmark functions	Name	Type
$f_1 = \sum_{i=1}^D [(10^6)^{i-1/D-1} x_i^2]$	Elliptic	Unimodal
$f_2 = \sum_{i=1}^D ( x_i + 0.5 )^2$	Step	Unimodal
$f_3 = \sum_{i=1}^D x_i^2$	Sphere	Unimodal
$f_4 = 10^6 x_1^2 + \sum_{i=2}^D x_i^2$	Tablet	Unimodal
$f_5 = \sum_{i=1}^D (\sum_{i=1}^D x_i)^2$	Quadric	Unimodal
$f_6 = \sum_{i=1}^D [x_i^2 - 10 \cos(2\pi x_i) + 10]$	Rastrigrin	Multimodal
$f_7 = -20 \exp(-0.2 \sqrt{\frac{1}{D} x_i^2})$	Ackley	Multimodal
$f_8 = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \pi \cos(x_i / \sqrt{i}) + 1$	Griewang	Multimodal
$f_9 = \sum_{i=1}^D x_i \sin(\sqrt{ x_i })$	Schwefel	Multimodal
$f_{10} = -\frac{1 + \cos(12\sqrt{x_1^2 + x_2^2})}{1/2(x_1^2 + x_2^2) + 2}$	Drop wave	Multimodal

These benchmark functions in table 1 contain both unimodal and multimodal functions.

The examination of the proposed approach on homogeneous PSO will be carried out in two phases:

- In the first one, we compare our approach with a number of improved variants of PSO chosen for the homogenous case of PSO variants. Table 2 below shows these chosen variants related to each approach. That is, SAPSO proposes a self-adapting approach while LinWPSO consists of varying inertia weight. In YSPSO, another parameter has been used which is the constriction factor instead of the inertia weight. APSO is an Adaptive PSO with a fuzzy classification of states. CLSPSO consists of using another search strategy to enhance particles cooperation in PSO. SimPSO consists of using a hybrid approach instead of adapting the parameters. RandWPSO is a random inertia weight PSO. LinWPSO consists of Linear decreasing weights PSO. RankPSO is

an Adaptive Particle rank, and ChaoPSO is a Chaotic Inertia Weight PSO. DPPSO is a Dynamic population size based PSO.

*Table 2 PSO variants from literature*

Algorithm	Name	Parameters Setting	Reference
APSO	Adaptive PSO	$c_1 = c_2 = 2, \omega = 0.9$	[60]
YPSO	PSO with compressibility factor	$c_1 = c_2 = 2, \omega = 0.9$	[99]
SELPSO	Natural selection based PSO	$c_1 = c_2 = 2, \omega = 0.9$	[100]
DPPSO	Dynamic population size based PSO	$c_1 = c_2 = 2, \omega = 0.9$	[17]
LinWPSO	Linear decreasing weights PSO	$\omega_{min} = 0.0001,$ $\omega_{max} = 0.1$	[101]
CLPSO	Cooperative line search PSO	$c_1 = c_2 = 2, \omega = 0.9$	[102]
SAPSO	Self-adaptive PSO	$c_1 = c_2 = 2, \omega_{min} = 0.01,$ $\omega_{max} = 0.9$	[100]
SecPSO	Swarm-core evolutionary PSO	$c_1 = c_2 = 2, \omega = 0.9$	[103]
AsyLnCPSO	Asynchronous PSO	$c_{1min} = c_{2min} = 0.01$ , $c_{1max} = c_{2max} = 2$	[100]
RandWPSO	Random inertia weight PSO	$c_1 = c_2 = 2, \omega_{min} = 0.01,$ $\omega_{max} = 0.9$	[101]
SecVibratPSO	Order oscillating PSO	$c_1 = c_2 = 2, \omega = 0.9$	[104]
SimuAPSO	PSO with Simulated Annealing	$\lambda = 0.0001, c_1 = c_2 = 2,$ $\omega = 0.9$	[103]

- In the second phase, we compare different variant of our approach to analyze the effect of each parameter adaptation using our generic adaptation model. In HMM-APSO, HMM is used just for adapting the acceleration factors as in [97]. In HMM-PPSO, HMM is executed for the control of the population size. In HMM-Wpso, the generic adaptive model is applied to the inertia weight adaptation.

Concerning the first compared approaches, we have implemented them using Matlab, while for the literature variants we have executed their code which is available online as benchmark functions.

Tests are performed 31 times with the same value of the parameters (if the parameter value is fixed in the corresponding variant). The used swarm population size is 30 with a dimension of 30. Population changes is done between 20 and 50 in the case of

population size variation as in HMM-PPSO. Each run contains 1000 generation of the optimization process.

To improve our HMM-based approaches, we compared results obtained for the benchmark test functions with best known PSO variants from literature. Performance is qualified following the main measured observations: Comparison on the solution accuracy, Comparison on the convergence speed and statistical tests.

The use of parametric two-sided tests named t-test (parametric) is performed on obtained results. Using t-test as an inferential statistical evaluation in the experiments [105]. Results variance is estimated using the sample execution test sets with Student's t-Test that measures if the difference between two means is statistically significant. Given as:

$$S^2 = \frac{\sum_{i=1}^N (r_i - \bar{r})^2}{N - 1} \quad (28)$$

Where N number of executions and  $r_i$  the result execution i.

We address the two hypothesizes as:

- Hypothesis H0 is that the compared approach is similar to the other PSO variants
- Hypothesis H1 is that the compared approach is different from the other PSO variants.

We perform this statistical test to investigate the given hypotheses. Tests are done with a significance level of 0.05 between the HMM-based approaches and different compared approaches of PSO variants. (More detailed explanation of this approach is given in [60]. For very little P-values ( $P\text{-value} < e-06$ ), it takes 0 as value. Statistical tests are executed with the statistical toolbox of Matlab. Tables will display the P-values of each compared variant with additional Rows: 1 (Better), 0 (Same) and -1 (Worse), to give the number of functions that is compared to the proposed approach, performs significantly (respectively) better than, almost the same as, and significantly worse than the compared algorithm.

#### **d.2 Examination of HMM-wPSO with other PSO variants**

We perform several runs on the HMM-wPSO and others PSO variants earlier described.

- solution accuracy

Best and the average value resulted from experimentations are given in table 3 below:

**Table 3 Results comparisons with other variants of PSO**

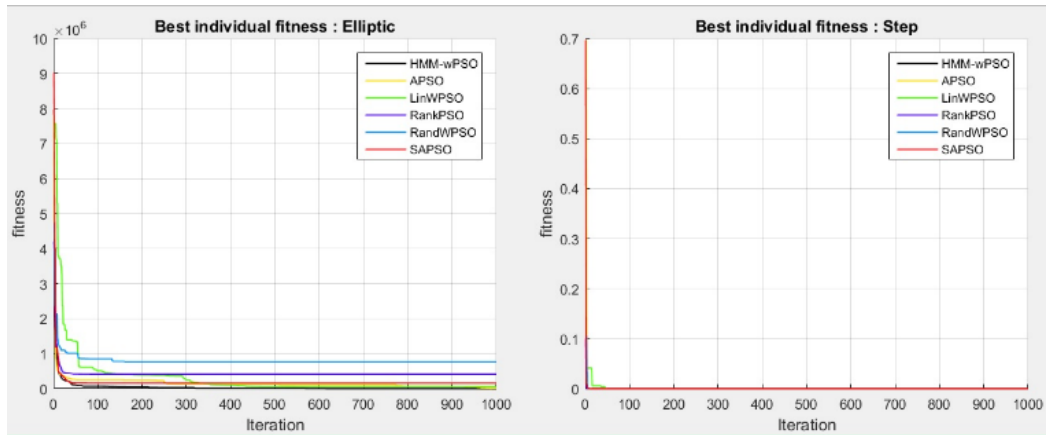
Functions		HMM-wPSO	PSO	APSO	ChaoPSO	RandWPSO	RankPSO	SAPSO	LinWPSO
$f_1$	Best	<b>35.0045</b>	17556.61	3314.52	744382.6	291814.4	85957.88	36451.55	10694.93
	Mean	<b>271.66</b>	71920.9	12697.63	1741984	576666	192511	188337	51398
$f_2$	Best	<b>0</b>	0	0	4.55e-27	0	0	0	0
	Mean	<b>0</b>	0	0	1.56e-06	1.22e-07	0	0	0
$f_3$	Best	<b>0.0177</b>	15.3432	0.83768	53.9977	29.3347	14.6594	21.771	17.0416
	Mean	<b>0.0569</b>	26.1594	2.2923	81.7409	41.9482	32.5917	29.0454	21.8367
$f_4$	Best	<b>0.0327</b>	36.3069	1.0421	117.1186	70.319	28.4467	34.7361	26.7697
	Mean	<b>0.161</b>	48.9845	2.682	221.7902	132.559	68.6518	56.6107	42.4692
$f_5$	Best	<b>13835</b>	1459678	281462	132e+06	113e+06	115e+04	828e+04	173e+04
	Mean	<b>56158</b>	103e+05	915e+03	575e+06	265e+06	364e+05	225e+05	102e+05
$f_6$	Best	<b>7.1343</b>	136.7593	28.955	282.7734	253.1783	140.6755	169.8982	120.695
	Mean	<b>12.3368</b>	190.6136	62.0158	333.6893	290.8753	228.0515	207.5413	189.3113
$f_7$	Best	<b>0.022324</b>	3.4004	1.7784	5.3846	4.7509	4.1292	4.341	4.1626
	Mean	<b>0.37133</b>	4.3781	2.453	6.4555	5.219	5.221	5.1512	4.6914
$f_8$	Best	<b>6.51e-05</b>	0.075818	0.016021	0.41017	0.14908	0.1714	0.064283	0.043456
	Mean	<b>0.0147</b>	0.13982	0.030788	0.5108	0.3844	0.20155	0.17696	0.11427
$f_9$	Best	<b>14.5948</b>	17.6234	52.4179	51.5312	72.3369	33.3361	21.5786	11.9212
	Mean	<b>25.8097</b>	26.7725	71.5969	155.0976	138.019	47.0875	36.5762	22.14
$f_{10}$	Best	<b>-1</b>	-1	-1	-0.99946	-0.99999	-1	-1	-1
	Mean	<b>-0.98725</b>	-0.99362	-0.96812	-0.94191	-0.96172	-0.96812	-0.98087	<b>-1</b>

The proposed approach gives, in most cases, better results than the majority of state of the art. The solution accuracy is enhanced for both unimodal (Elliptic, Step, Sphere, Tablet, Quadric) and multimodal functions (Rastrigrin, Ackley, Griewang, Schwefel, Drop wave). For Elliptic function, the HMM-wPSO gives more accuracy in the order of  $10^3$  compared to the other inertia weight variation strategies of PSO variants. For Step function, this function is simple; the HMM-wPSO has obtained the best solution 0 as almost all the other variants. For Sphere function, the HMM-wPSO gives more accuracy in the order of  $10^2$  compared to the other inertia weight variation strategies of PSO variants. For Tablet function, the HMM-wPSO gives more accuracy in the order of  $10^2$  compared to the other inertia weight variation strategies of PSO variants. For Quadric function, the HMM-wPSO gives more accuracy in the order of  $10^3$  compared to the other similar PSO variants. For Ackley function, the HMM-wPSO gives more accuracy in the order of 10 compared to the other inertia weight variation strategies of PSO variants. For Rastrigrin function, the HMM-wPSO gives also enhanced accuracy results in order of 10 compared to the other inertia weight variation strategies of PSO variants. For Schwefel function, the HMM-wPSO has a little improvement of accuracy compared to the other chosen PSO variants. For Drop wave function, the HMM-wPSO provides similar solution results compared to the other inertia weight variation strategies of PSO variants. Indeed, results presented in bold, it exceeds the majority of other PSO variants. HMM-wPSO has improved good performances in terms of solution accuracy.



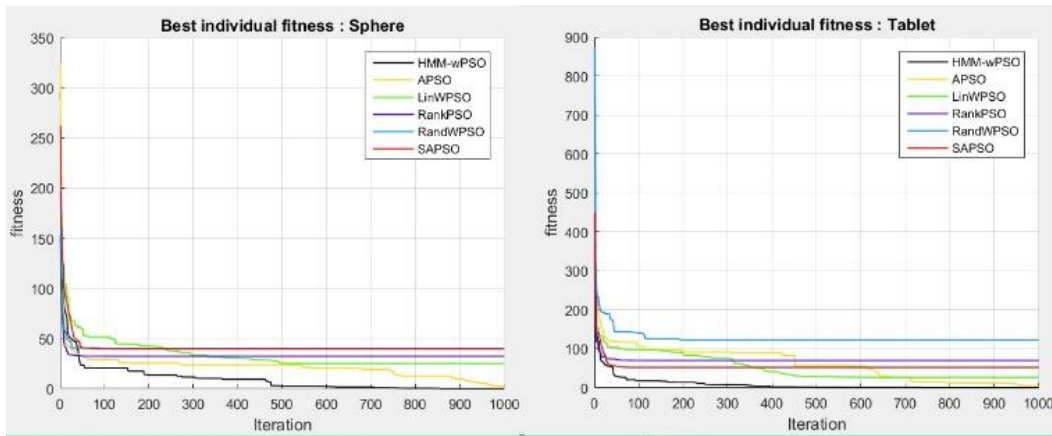
- convergence speed

Convergence speed of HMM-wPSO is shown in the following figures (Figure 13):



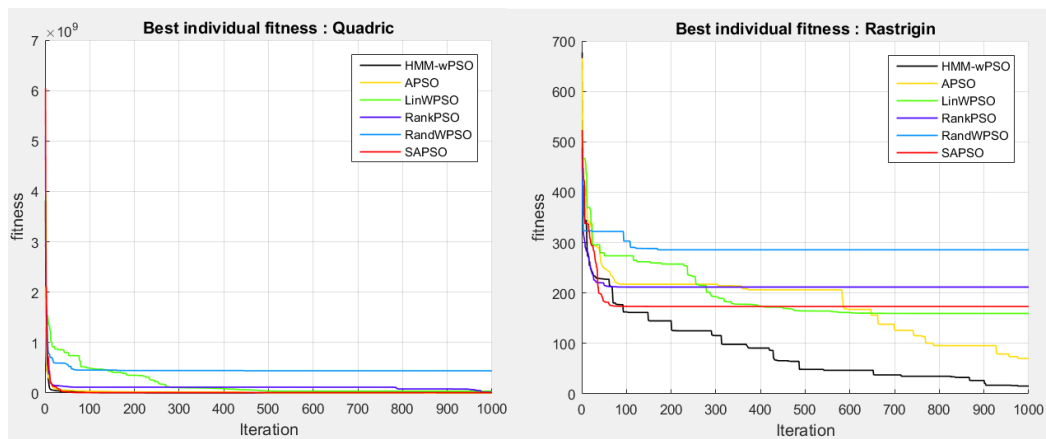
(a)

(b)



(c)

(d)



(e)

(f)

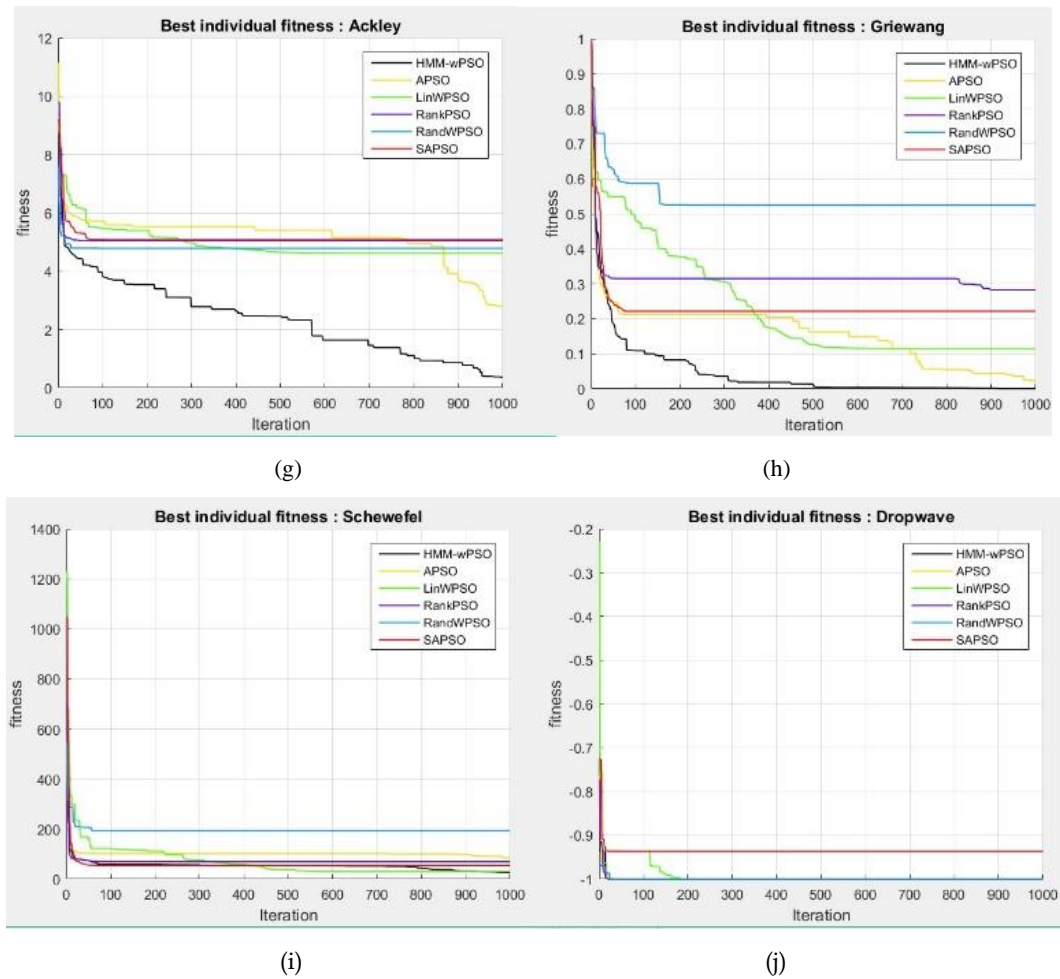


Figure 14 Comparison on convergence speed on fitness functions (a)  $f_1$ . (b)  $f_2$ . (c)  $f_3$ . (d)  $f_4$ . (e)  $f_5$ . (f)  $f_6$ . (g)  $f_7$ . (h)  $f_8$ . (i)  $f_9$ . (j)  $f_{10}$

For unimodal functions, HMM-wPSO quickly gives the best solution even before 50 iterations in Elliptic, step and Quadric functions. And before 400 in sphere and tablet.

For other more complex multimodal functions, the line showing the convergence of HMM-wPSO appears largely under all other PSO variants, which shows the superiority against the others. This is easily distinguished in Rastrigin, Ackley and Griewang. HMM-wPSO quickly gives the best solution even before 20 iterations in Dropwave, and slightly better convergence in Schwefel.

In general, the black line of figure 14 has more convergence speed than other lines. For all functions, the HMM-wPSO speeds up the optimization across iterations. Then, for the convergence speed, HMM-wPSO shows its supremacy. We can consider that the adaptiveness of inertia weight with HMM control gives a huge positive impact on the convergence speed better than other inertia weight control PSO variants from literature.

- statistical tests

To prove the performances of our approach, we use a parametric two-sided test named t-test (parametric), it is performed on obtained results. Tests are done with a

significance level of 0.05 between the HMM-wPSO and different compared inertia weight approaches of PSO variants.

*Table 4 T-test comparison*

Functions	PSO	APSO	ChaoPSO	RandWPSO	RankPSO	SAPSO	LinWPSO
$f_1$	0	0	0	0	0	0	0
$f_2$	0.5560	0.1769	0	0.0734	0.1769	0.6278	0.1510
$f_3$	0	0	0	0	0	0	0
$f_4$	0	0.0093	0	0	0	0	0
$f_5$	0	0	0.0013	0	0.0103	0	0.0158
$f_6$	0	0	0	0	0	0	0
$f_7$	0.8127	0	0	0	0	0.0389	0.3040
$f_8$	0	0	0	0	0	0	0
$f_9$	1	1	0.3275	0.3305	1	1	1
$f_{10}$	0	0	0	0	0	0	0
<b>+1 (better)</b>	17	18	19	18	18	17	18
<b>0 (same)</b>	3	2	1	2	2	3	2
<b>-1 (worse)</b>	0	0	0	0	0	0	0

We display in table 4 of P-values on every function of statistical tests with a significance level of 0.05. Rows “1 (Better),” “0 (Same),” and “-1 (Worse)” give the number of functions that the HMM-wPSO performs significantly better than, almost the same as, and significantly worse than other algorithms.

Executing statistical inferred t-test on the thirty executions, clearly, the HMM-wPSO outperforms the other algorithms and gives a competitive upgrade in PSO performances. In general, in approximatively 90% of results, HMM-wPSO is better and in 10% is similar in the case of simple functions like the Step function. So, inertia adaptation using HMM is confirmed to be a good approach.

The approach of adapting inertia weight based on HMM gives much better statistical results than most of PSO related variants. The reason may lie in the fact that HMM can give a more robust and authentic estimation of the four states at each iteration by benefiting from the complete historical information based on the observed evolutionary factor through iterations. Also, it combines different dynamic variation strategies according to each search state.

### d.3 Examination of HMM-PPSO with other PSO variants

In order to further evaluate our approach HMM-PPSO, we compare with some related well-known variants of PSO in the literature as defined in the parameters setting paragraph.

- solution accuracy

Table 5 below shows the comparison of results between these algorithms.

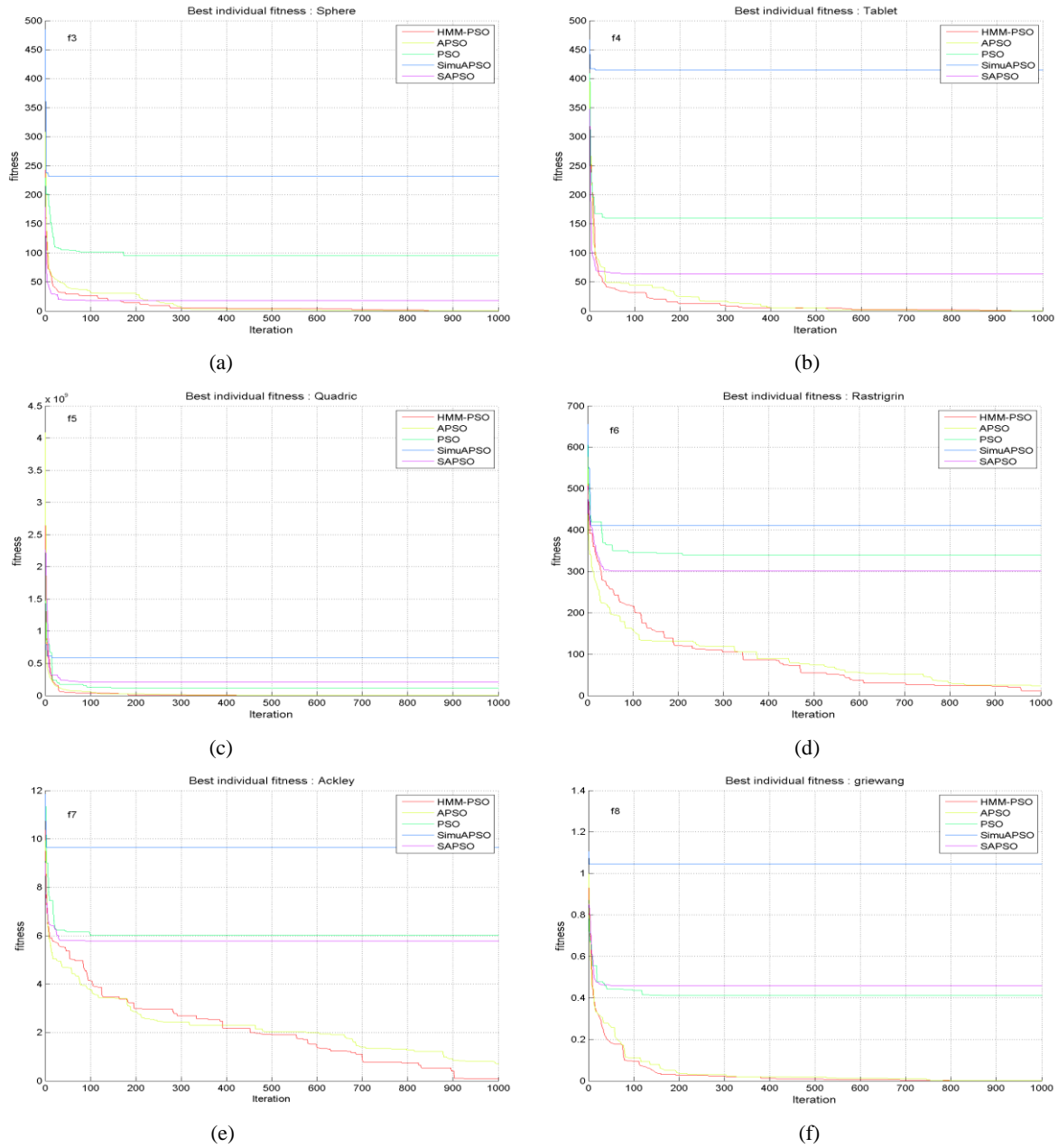
*Table 5 Comparison of results*

Functions		CLS-PSO	YPSO	PSO	SAPSO	SimPSO	DPPSO	APSO	HMM-PPSO
$f_1$	Best	85567	646.982	17556.61	3899.11	76023	331.4682	377.5448	<b>28.5207</b>
	Mean	627396	2676.93	71920.96	15669.79	295411	877.7602	1038.920	<b>132.2178</b>
$f_2$	Best	0	0	0	0	8.78e-05	1.04e-11	0	<b>0</b>
	Mean	1.1e-05	0	0	0	0.052643	7.12e-07	0	<b>0</b>
$f_3$	Best	71.6543	7.3068	15.3432	13.159	108.7304	3.297	0.48545	<b>0.010322</b>
	Mean	225.9075	14.448	26.1594	29.648	186.9263	5.3647	2.4973	<b>0.054323</b>
$f_4$	Best	202.1904	22.0672	36.3069	19.0642	283.6052	13.4607	1.1408	<b>0.007489</b>
	Mean	402.8212	41.8765	48.9845	84.6	494.3694	24.2636	4.2255	<b>0.047697</b>
$f_5$	Best	205e+6	1705011	1459678	280e+5	4510e+5	160e+05	87133.58	<b>458.8865</b>
	Mean	707e+6	6950767	103e+05	940e+5	146e+7	484e+05	893462.6	<b>7585.629</b>
$f_6$	Best	371.2168	109.964	136.7593	184.9184	328.9158	182.9232	37.3897	<b>7.2419</b>
	Mean	535.107	182.184	190.6136	246.1648	473.3492	234.3271	57.3572	<b>15.3853</b>
$f_7$	Best	7.3411	3.659	3.4004	4.2541	7.0977	2.3791	1.4562	<b>0.016588</b>
	Mean	9.4979	4.4661	4.3781	5.5108	9.0712	2.8154	2.2541	<b>0.4568</b>
$f_8$	Best	0.38083	0.07117	0.075818	0.11752	0.50343	0.026	0.00595	<b>4.51e-05</b>
	Mean	0.75682	0.16698	0.13982	0.34062	0.91885	0.10926	0.026352	<b>0.016116</b>
$f_9$	Best	158.36	15.5126	17.6234	38.2425	162.8896	26.1634	33.239	<b>6.5624</b>
	Mean	405.2457	24.7067	26.7725	66.7705	456.2188	46.7503	56.3267	<b>16.5693</b>
$f_{10}$	Best	-1	-1	-1	-1	-0.93624	-0.99988	-1	<b>-1</b>
	Mean	-0.94403	-0.98972	-0.99362	-0.96092	-0.79288	-0.99474	-0.9856	<b>-0.97326</b>

However, those results improve the HMM classification methodology with adaptive size. The solution accuracy is enhanced for both unimodal (Elliptic, Step, Sphere, Tablet, Quadric) and multimodal functions (Rastrigrin, Ackley, Griewang, Schwefel, Drop wave). For unimodal functions, the HMM-PPSO gives more accuracy in a varying order that can be remarkable especially in the Quadric function, where HMM-PPSO gives a better solution accuracy at an order of  $10^3$ . The results of the Step function are similar due to its simplicity to find the best solution. On the other hand, the multimodal functions show more enhanced solution accuracy in the order of  $10^2$  for the Griewang function, and relatively better solution in Rastrigrin, Ackley, and Schwefel. For the Drop wave, results are similar. Therefore, HMM coupled with size adaptation gives more improved state adaptation to the PSO algorithm.

- convergence speed

To illustrate the difference between these algorithms in terms of convergence, we display in Figure 15 below the evolution of the results during the optimization process of each one of these algorithms by each fitness function.



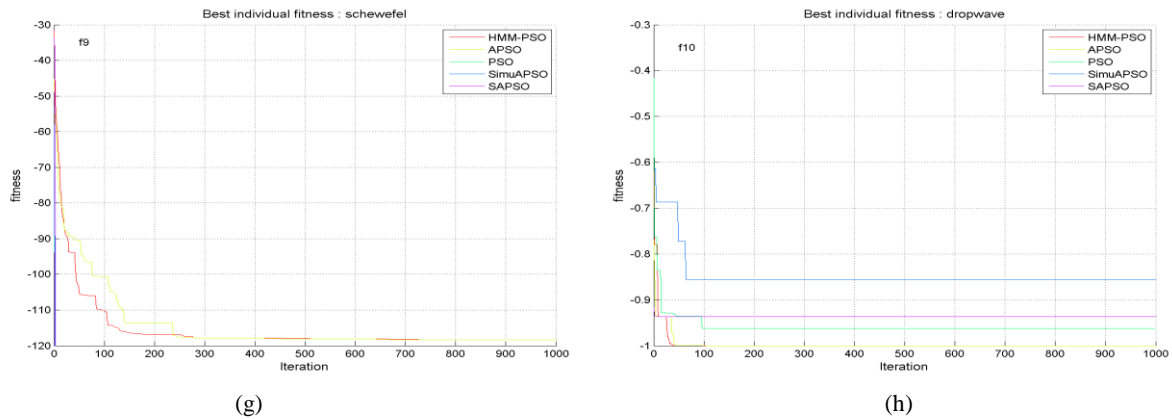


Figure 15 Performance comparison with different functions (a)  $f_3$ . (b)  $f_4$ . (c)  $f_5$ . (d)  $f_6$ . (e)  $f_7$ . (f)  $f_8$ . (g)  $f_9$ . (h)  $f_{10}$

When we interest on the convergence rate for unimodal and multimodal functions, HMM-PPSO gives the best solution. However, the solution is not obtainable is a relatively good speed. It converges before 400 iterations in Elliptic, Step and Griewang. It converges before 200 in Quadric and Dropwave functions. As regards the Rastrigin, Schwefel and Ackley, the convergence line of HMM-PPSO shows that the HMM-PPSO remains seeking and improving the solution almost in all iterations.

Thus, it can be noticed that the HMM adaptation improve the convergence visibly, so HMM-PPSO has a faster convergence rate than other shown PSO variants.

- statistical tests

For further comparison of the algorithms, we have considered the parametric two-sided test, which is the t-test (parametric). A significance level of 0.05 is considered between the HMM-PPSO and different compared PSO variants. (The sample is superior to thirty [106]).

Table 6 Statistical tests

Functions	HMM-APSO	HMM-PPSO	APSO	DPPSO	CLSPSO	YSPSO	SAPSO	LinWPSO	SimPSO
$f_1$	0.2975	0.9019	0	0	0	0.0072	0	0	0
$f_2$	0.1700	0.9635	0	0	0	0	0	0	0
$f_3$	0.7713	0.1855	0.7713	0.0468	0.0024	0.3575	0.0043	0.0095	0
$f_4$	0.6962	0.8364	0	0	0	0	0	0	0
$f_5$	0	0.7960	0	0	0	0	0	0	0
$f_6$	0.309	0.7545	0.0193	0	0	0	0	0	0
$f_7$	0.9211	0.3420	0	0	0	0	0	0	0
$f_8$	0	0.2063	0	0	0	0	0	0	0
$f_9$	0	0.8163	0.1027	0.011	0	0	0	0	0
$f_{10}$	0.6246	0.5986	0	0	0	0	0	0	0
<b>+1 (better)</b>	3	0	8	10	9	9	9	9	10
<b>0 (same)</b>	7	10	2	0	1	1	1	1	0
<b>-1 (worse)</b>	1	0	0	0	0	0	0	0	0

Compared with the other algorithms, when performing the statistical inferred t-test on the thirty executions, visibly the HMM-PPSO outperforms the other algorithms and gives a good upgrade in PSO performances. In overall, and at approximately 80% of the results, HMM-PPSO is better in 20% of results and is similar in the case of some simple functions like the Step function. So, population adaptation using HMM is also confirmed to be a good approach. It can be seen that HMM-PPSO outperforms the other algorithms in terms of the t-test test results.

#### d.4 Examination of HMM-APSO with other PSO variants

To improve our HMM-APSO approach, we compared results obtained for the benchmark test functions with best known PSO of the literature.

- solution accuracy

In order to further evaluate our approach, we compare HMM-APSO with well-known variants of PSO in the literature. For every benchmark function, ten executions are performed for all variants of PSO. The table below shows the results of the simulations.

Table 7 Results comparisons with the variants of PSO

Functions		APSO	SimuA-PSO	Sec-PSO	Rand WPSO	YSPS O	SeI PS O	SecVi bratPS O	SAPSO	LinW PSO	AsyLn CPSO	HMM-APSO
$f_1$	Best	49	115719	4689	9843	1420	16382	275	5095	7067	3765	<b>44</b>
	Mean	150	288102	10930	33384	2726	27998	32132	14850	20280	11045	<b>172</b>
$f_2$	Best	0	5.8e-06	1.9e-31	2.1e-12	0	8.6e-10	7.3e-10	0	0	0	<b>0</b>
	Mean	0	0.04	9.8e-12	3.6e-05	0	1.8e-05	0.03	0	0	3.1e-30	<b>0</b>
$f_3$	Best	0.01	93.19	20.58	37.7	6.77	36.01	0.8	18.20	17.66	22.89	<b>4.6e-3</b>
	Mean	0.05	188.56	38.79	64.09	13.77	63.59	71.05	31.96	40.53	34.58	<b>0.04</b>
$f_4$	Best	0.02	251.36	71.4	110.48	23.72	30.47	17.42	37.21	51.24	21.73	<b>0.02</b>
	Mean	0.05	396.21	110.48	246.78	42.31	77.51	199.04	84.79	95.74	44.32	<b>0.07</b>
$f_5$	Best	16435	587e+6	421e+5	102e+6	127e+4	839e+5	107e+6	459e+5	165e+5	131e+5	<b>7644</b>
	Mean	67851	210e+7	978e+5	434e+6	843e+4	210e+6	562e+6	166e+6	155e+6	384e+5	<b>49205</b>
$f_6$	Best	8.24	358.22	208.54	293.88	142.44	285.16	221.93	165.66	170.64	193.83	<b>4.31</b>
	Mean	16.14	462.02	262.89	322.35	175.84	315.60	344.76	273.31	261.77	291.54	<b>12.41</b>
$f_7$	Best	0.03	6.9436	4.8963	5.403	3.1785	5.665	1.2753	3.8279	4.7261	5.8372	<b>0.03</b>
	Mean	0.31	8.6336	5.2716	6.5613	4.2219	6.1951	4.4528	5.2531	5.6829	6.8189	<b>0.33</b>
$f_8$	Best	7.50e-5	0.52	0.15	0.25	0.05	0.27	0.05	0.13	0.14	0.08	<b>1.1e-5</b>
	Mean	0.01	0.87	0.25	0.45	0.12	0.41	0.42	0.25	0.31	0.23	<b>0.07</b>
$f_9$	Best	-118.35	-7.2+47	-4+158	-	-3+34	-1e+30	-	-1+231	-1e+22	-3e+47	<b>-118.35</b>
	Mean	-118.34	-1e+47	-9e+15	-	-3e+33	-	-	-1e+230	-1e+21	-3e+46	<b>-118.34</b>
$f_{10}$	Best	-1	-0.92	-1	-0.94	-1	-0.99	-0.93	-1	-1	-1	<b>-1</b>
	Mean	-1	-0.74	-0.96	-0.93	-0.98	-0.95	-0.82	-0.97	-0.96	-0.98	<b>-1</b>

The Results obtained from the mean of all executions show that for almost all the benchmark functions, HMM-APSO gives the best results when comparing to the other PSO variants from the literature. The solution accuracy is improved for both unimodal (Elliptic, Step, Sphere, Tablet, Quadric) and multimodal functions (Rastrigrin, Ackley, Griewang, Schwefel, Drop wave). For the Step, Sphere and Drop wave functions HMM-APSO is slightly the best and outperforms with a little difference. The more noticeable improvements of solution accuracy can be found in an order attaining more than  $10^2$  of best accuracy; it is obtained in Quadric and Sphere functions. For the other function, the results from HMM-APSO are better than that the compared PSO variants, and slightly better than the APSO. In general, HMM-APSO gives good accuracy.

- convergence speed

HMM-APSO is compared to other PSO variants in terms of convergence speed. Figure 16 shows the drawn convergences lines across iterations:



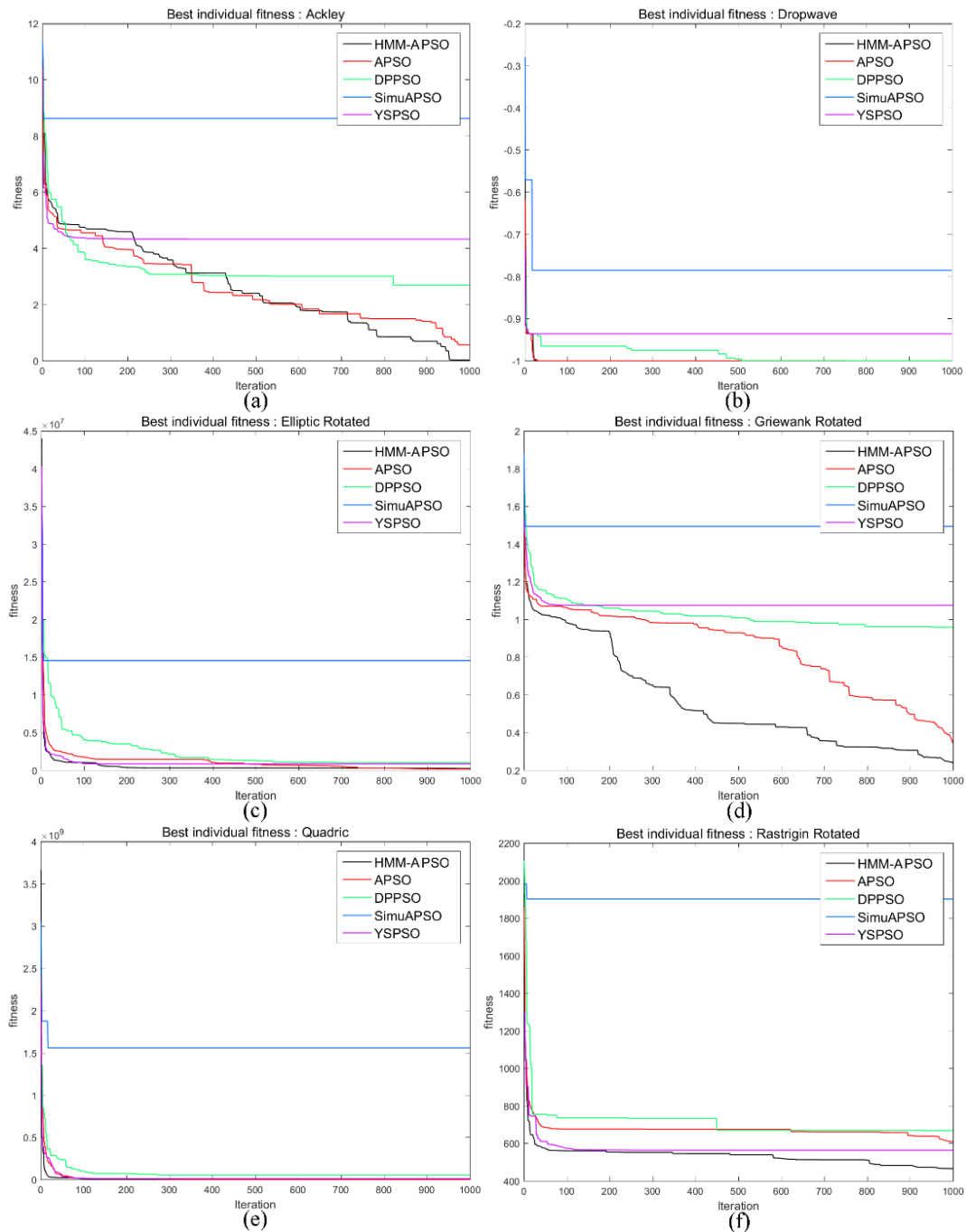


Figure 16 Comparison on Benchmark functions. (a)  $f_7$ . (b)  $f_{10}$ . (c)  $f_1$ . (d)  $f_8$ . (e)  $f_5$ . (f)  $f_6$ .

To illustrate the difference between the chosen PSO variants and the proposed approach in terms of convergence, we display in figure 16 the evolution of the results during the optimization process of each one of these algorithms by each fitness function. Considering the convergence rate, we can notice that the HMM adaptation of acceleration coefficients improve the convergence visibly. HMM-APSO quickly gives the best solution even before 50 iterations in Drop wave and Quadric functions. And before 400 in Elliptic function. The convergence is improved in the Ackley and Rastrigin functions where their convergence lines continue to show an improved solution during approximately all the run time. Thus, we can notice from Figure 16, that HMM-APSO has a faster convergence rate than other PSO variants

- Statistical tests

The statistical tests comparison is drawn in the following table 8:

*Table 8 Statistical tests*

Functions	APSO	DPPSO	CLPSO	YPSO	SAPSO	LinWPSO	SimPSO
$f_1$	0	0	0	0.0072	0	0	0
$f_2$	0	0	0	0	0	0	0
$f_3$	0.7713	0.0468	0.0024	0.3575	0.0043	0.0095	0
$f_4$	0	0	0	0	0	0	0
$f_5$	0	0	0	0	0	0	0
$f_6$	0.0193	0	0	0	0	0	0
$f_7$	0	0	0	0	0	0	0
$f_8$	0	0	0	0	0	0	0
$f_9$	0.1027	0.011	0	0	0	0	0
$f_{10}$	0	0	0	0	0	0	0
+1 (better)	18	20	19	19	19	19	20
0 (same)	2	0	1	1	1	1	0
-1 (worse)	0	0	0	0	0	0	0

For further comparison of the PSO variants against the HMM-APSO approach, we have considered a statistical test that is the t-test with a significance level of 0.05 as shown in the parameters setting of the paragraph above. Visibly, the HMM-APSO overcomes the other algorithms and leads to a competitive enhancing in PSO performances. In the totality, in just around 90% of the results, HMM-APSO is better and in 10% is similar in the case of simple functions like the Step function. So, acceleration coefficients adaptation using HMM is statistically giving better results than the compared state of arts.

#### d.5 Comparison of the HMM-based approaches

In this part, we compare all HMM-based approaches for different PSO parameter control and adaptation namely: HMM-wPSO, HMM-APSO, and HMM-PPSO. The online version of the generic model is tested for adapting acceleration coefficient called Online-HMM-APSO. Another variant called HMM-PSO is added to this comparison that merges the three approaches in one. HMM-PSO is using the generic adaptation model to control: inertia weight, acceleration coefficients, and population size. The empirical study of HMM derivate approaches will be like the previous paragraph according to the three aspects: solution accuracy, Comparison on the convergence speed and statistical tests.

- Comparison on solution accuracy

Table 9 below shows the solution accuracy comparison of the proposed HMM-based approaches of PSO related to each parameter control:

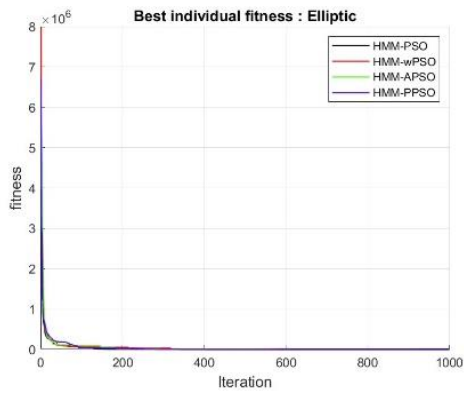
*Table 9 Results comparison*

Function		HMM-APSO	Online-HMM-APSO	HMM-PPSO	HMM-wPSO	HMM-PSO
$f_1$	Best	89	44	28.5207	35.0045	16.647
	Mean	145	172	132.2178	271.66	116.24
$f_2$	Best	0	0	0	0	0
	Mean	0	0	0	0	0
$f_3$	Best	0.02	4.6e-3	0.010322	0.0177	0.013
	Mean	0.09	0.04	0.054323	0.0569	0.0484
$f_4$	Best	0.03	0.02	0.007489	0.0327	0.0096
	Mean	0.26	0.07	0.047697	0.161	0.0554
$f_5$	Best	6622	7644	458.8865	13835	425.11
	Mean	51085	49205	7585.629	56158	5215.029
$f_6$	Best	4.89	4.31	7.2419	7.1343	6.2499
	Mean	12.29	12.41	15.3853	12.3368	15.3697
$f_7$	Best	0.05	0.03	0.016588	0.022324	0.051531
	Mean	0.29	0.33	0.4568	0.37133	0.46979
$f_8$	Best	1.32e-4	1.1e-5	4.51e-05	6.51e-05	4.20e-05
	Mean	5.24e-3	0.07	0.016116	0.0147	0.00712
$f_9$	Best	-118.35	-118.35	6.5624	14.5948	6.645
	Mean	-117.97	-118.34	16.5693	25.8097	14.8617
$f_{10}$	Best	-1	-1	-1	-1	-1
	Mean	-0.97	-1	-0.97326	-0.98725	-0.9851

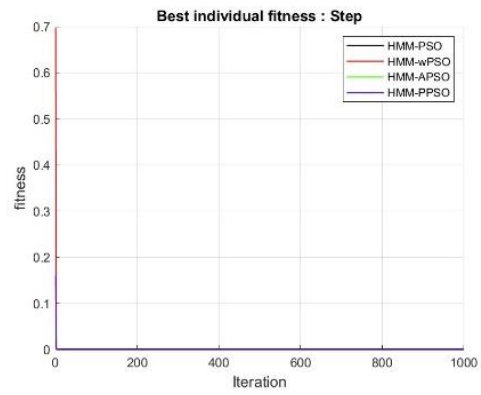
Concerning the solution accuracy, it has been more enhanced when controlling all parameters in HMM-PSO. Thus, the HMM-PSO has improved visibly the solution accuracy for the Elliptic, Tablet, Quadric and Schwefel functions. The improvement is at the order of 10 at maximum. For the other functions, the results are the same or slightly different. Results are normally the case of simple functions as Step and Dropwave. The online learning version gives approximatively the same results as the batch learning. The one difference is in the CPU time, which is a little improved. When classing the results of HMM approaches applied to the PSO parameters control, the HMM-PSO which control all parameters is the best, then the HMM-PPSO which controls the population size, then the HMM-APSO which controls the acceleration coefficients, then finally with little difference the HMM-wPSO, which adapts the inertia weight.

- Comparison on convergence speed

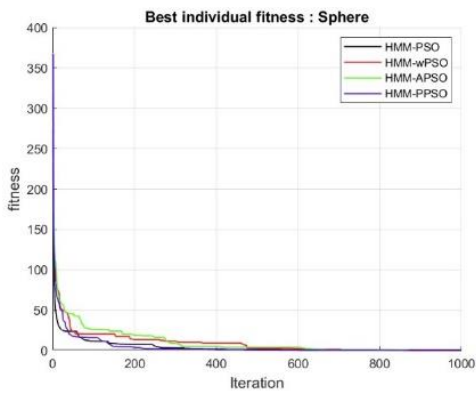
We compare HMM proposed approaches based on convergence speed.



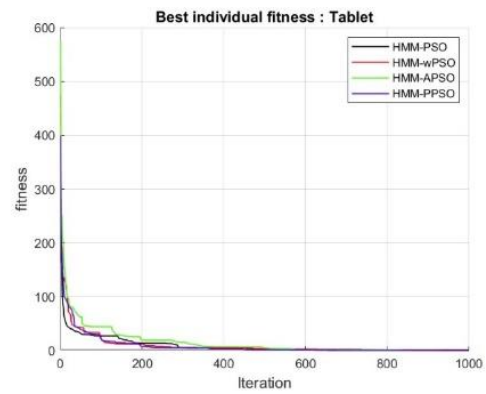
(a)



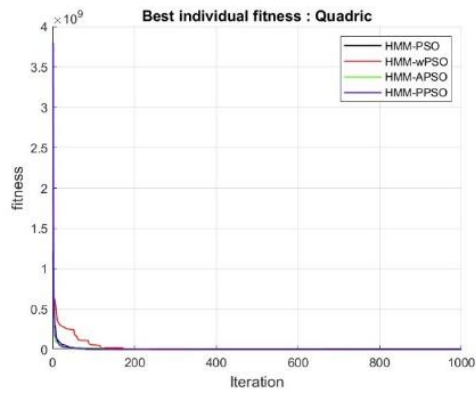
(b)



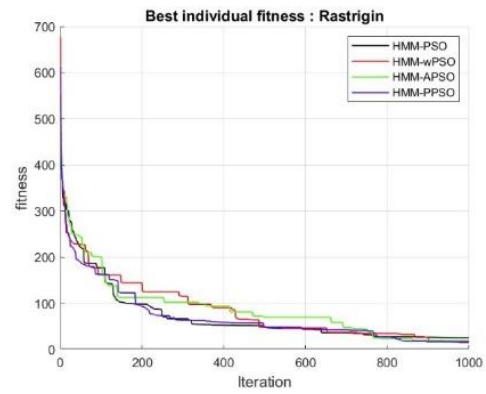
(c)



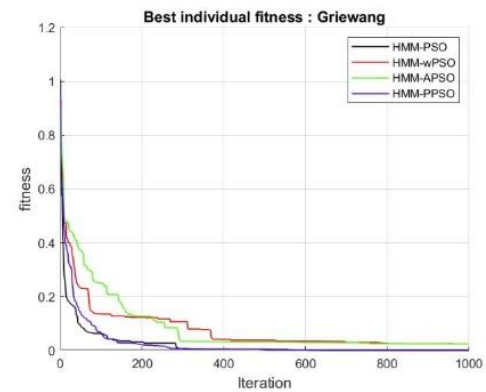
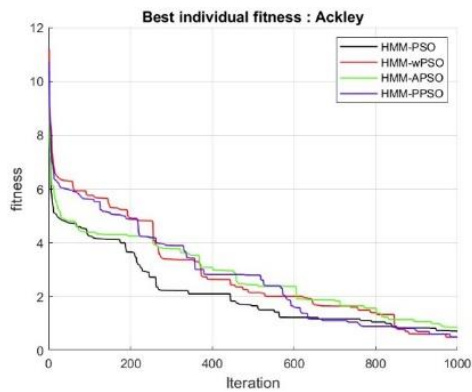
(d)



(e)



(f)



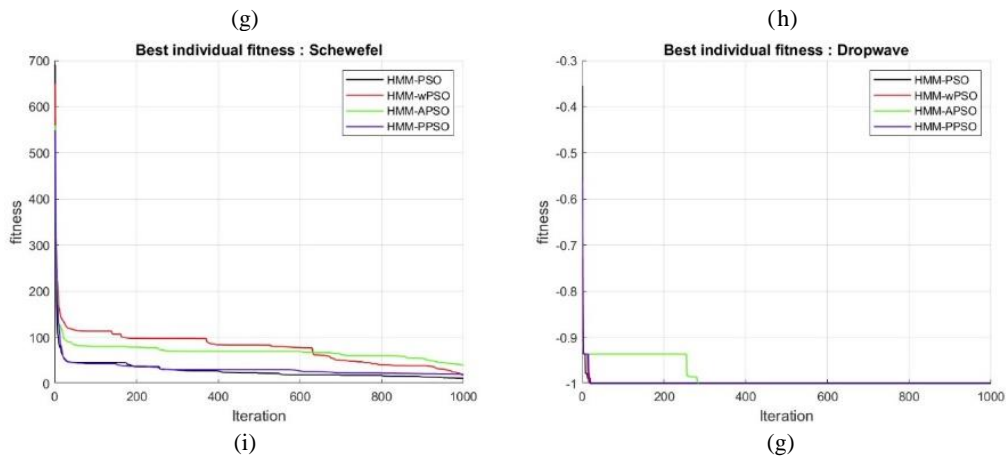


Figure 17 HMM based approaches comparison of convergence speed (a)  $f_1$ . (b)  $f_2$ . (c)  $f_3$ . (d)  $f_4$ . (e)  $f_5$ . (f)  $f_6$ . (g)  $f_7$ . (h)  $f_8$ . (i)  $f_9$ . (j)  $f_{10}$

We can observe that the convergence speed in figure 17 is almost the same for all HMM-based approaches, with a little improvement for the HMM-PSO (line in black). When controlling one of the PSO parameters adequately and adaptively, the convergence speed will be enhanced due to the sensitivity of PSO convergence to each of its parameters. HMM-PSO gives best results normally due to the adaptively of all chosen PSO parameters.

- Statistical tests

The hypothesis  $H_0$  is that the HMM-PSO is similar to other HMM approaches and we perform a statistical test to investigate this assertion.

Table 10 Statistical tests

Function	HMM-wPSO	HMM-PPSO	HMM-APSO
$f_1$	1.2697e-04	0.9184	3.5433e-08
$f_2$	1	1	1
$f_3$	0.4995	0.5243	0.6900
$f_4$	3.2375e-04	0.8690	0.0720
$f_5$	0	0.9230	0
$f_6$	0.1338	0.6056	0.1906
$f_7$	0.3172	0.8988	0.1017
$f_8$	0.0122	1.7391e-04	0.0175
$f_9$	2.12e-04	0.1045	4.0143e-05
$f_{10}$	0.8323	0.8261	0.9454
+1 (better)	5	1	3
0 (same)	5	9	6
-1 (worse)	0	0	1

According to statistical tests, and in general, the HMM-PSO is better with a little relative enhancement regarding other HMM variants. HMM approaches work better when adapting all parameters together. When classifying the PSO based HMM approaches according to their statistical test, the better one will be the HMM-PSO that controls all parameters, then the HMM-wPSO with control the inertia weight, then the HMM-APSO which controls the acceleration coefficients, and lastly with a little difference the HMM-PPSO that controls the population size. However, any parameter adaptation with HMM gives a good result, which is almost similar to the approach of enhancing all parameters together. This outcomes from the influence of each of the three chosen parameters control (population size, inertia weight and acceleration coefficients) to modify and improve PSO search capabilities when it is well adapted with the PSO state.

## 5. CONCLUSION

In this chapter, we have integrated the Hidden Markov Model in PSO to learn and predict the most probable state to control PSO parameters at each iteration. This looks advantageous from the view that HMM is a robust stochastic classification tool that takes into account past information about the population to control and adapt the PSO parameters. The proposed meta-model based HMM has been used in this chapter in both online and offline parameters setting of the PSO algorithm.

In the offline adaptation or also called PSO tuning, we have proposed a novel approach which uses hidden Markov model to estimate the most likely state of the parameter values of any metaheuristic algorithm as the PSO algorithm. The proposed tuned PSO using HMM is a problem dependent. Its application will be done in chapter 5 for the airline scheduling problem (crew scheduling).

On the other hand, a meta-model of online parameters setting of PSO has been proposed in this chapter for homogeneous PSO. The adaptation scheme based HMM has been applied to the three chosen key parameters of PSO that are: the acceleration coefficients, the inertia weight, and the population size. Then, all those parameters have been controlled in one algorithm called HMM-PSO. The integrated Hidden Markov Model in PSO is used to learn and predict the most probable state to control each of the PSO parameters. This approach looks advantageous from the view that HMM is a robust stochastic classification tool that takes into account past information about the population to control and adapt the PSO parameters.

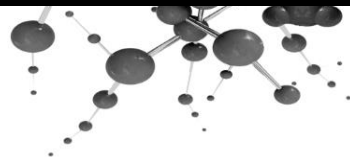
All PSO based HMM proposed variants have been evaluated and compared to the state art of the literature. The results were promising, and the proposed approach gives better results than the state of the art of other PSO variants regarding both solution accuracy and convergence speed. Those results were also demonstrated by advanced statistical tests which show the priority of HMM control in PSO. The results show also that PSO is sensitive to its parameters, and gains more enhancements when they are controlled adequately as in this approach.

The CPU time is influenced negatively in this proposed approach despite the gain in PSO convergence. This is normally due to the additive computation time used by the machine learning integration in PSO.

In this chapter, the PSO is assumed to be homogeneous, and all particles parameters are controlled in the same way at each iteration. The next chapter will give to particles different search behaviors in the same iteration such as the self-adaptation and the multi-swarm design.

Future research should attempt to apply our approach to other metaheuristics based population to benefit from the advantages of HMM state classification in parameters control. Also, the proposed approach needs to be applied to real complex optimization problems. An extension of this method to heterogeneous PSO is provided in the next chapter.

---



## CHAPTER II: ONLINE PARAMETERS CONTROL OF HETEROGENEOUS PARTICLE SWARM OPTIMIZATION

---

Parameters control of PSO can also be done in the context of a heterogeneous swarm, where the particles can adopt multiple behaviors during the same run of the PSO. Regarding heterogamous particle swarm optimization (PSO), we will make use of the earlier approach presented in the first chapter, to design a multi-agent behavior of particles as a means to enhance the diversity of the algorithm or to achieve a trade-off between exploration and exploitation. A commonly used heterogeneous form of PSO is based on the idea of considering multi-swarms (multi-populations). It consists of dividing the whole search space into local subspaces, each of which might cover one or a small number of local optima, and then separately searches within these subspaces. Another way to define multi-agent in PSO is to assign different roles to particles. Thus, different particles can play different roles, and each one of these particles can play different roles during the search processes. A challenging task within this PSO variant is how each particle has to decide which role it will assume. In this chapter, we investigate the integration of the generic model to a different type of heterogeneities, which can be attributed to particle swarms.



## 1. IMPROVED PSO VARIANTS FROM LITERATURE

In heterogeneous PSO [107], particles are allowed to take on different search behaviors. So, their parameters are not necessary properties of the swarm as a whole in the standard PSO, instantiating them at the individual level, therefore producing heterogeneity. We call a particle swarm heterogeneous if it has at least two particles that differ in any of the attributes mentioned above. Hence, at the individual level, we refer to a particular instantiation of these components, a particle's configuration. According to the nature of the distinctions between particles' configurations, several sorts of heterogeneity can be identified from the literature.

We can see that this variant of PSO enhancement method has been presented in the literature as a specific and separate algorithm known by multi-swarm optimization [108]. In the proposed variant, the authors have been inspired by the quantum model of atoms to define the quantum swarm. Also, another grouping approach has been suggested by [109].

Furthermore, the population has been divided into two swarms in order to introduce the divide and conquer concept using genetic operators. Another automation approach which can be used inside PSO is cellular automata (CA). CA can be considered as an arrangement of FSM. It can be used for instance to split the population of particles into different groups across cells of cellular automata. Reference [110] has integrated it in the velocity update to modify the trajectories of particles.

Also, [43] uses the notion of tribes to adapt PSO parameters including also the adaptation of tribes size by adding and removing particles.

Furthermore, [43] proposed the TRIBES, an adaptive variant of PSO which consists of changing the particles' behaviors as well as the topology of the swarm depending on the performance of the algorithm.

Moreover, [111] defined a cooperative approach to PSO based on dividing the population into four sub-swarms according to the four states. Furthermore, van den Bergh and Engelbrecht [30] used the concept of cooperative learning which consists of using multiple swarms to optimize the various components of the solution vector cooperatively. This idea is similar to the multi-agent approach which consists of dividing the particles into agents. Also, this type of control is related to the concept of cooperative swarms, which have been introduced by [30]. This principle has been achieved in their paper by using multiple swarms to optimize different components of the solution vector cooperatively. This idea is similar to the multi-agent approach which consists of dividing the particles into agents.

This issue can also be treated by clustering approaches as proposed by [112]. Their approach consists of assigning particles to different promising sub-regions basing on a hierarchical clustering method, where, at each iteration, particles are grouped into classes (named clusters), and the particles of each class have a specific role in the swarm (as in the multi-agent systems).

We can see from the literature that many papers have inspired from some approaches used in multi-agent systems to define the automated cooperative approach. An example of using the multi-agent concept in PSO can be found in [113], also in [49]. That is, incremental social learning which is often used to improve the scalability of systems composed of multiple learning agents has been used to improve the performance of PSO. Furthermore, [114] proposed a multi-agent approach which combines simulated annealing (SA) and PSO. We can remark that their idea is related to the generic notion of hyper-heuristics which consists of finding the most suitable configuration of heuristic algorithms. Reference [115] has cited the may feature obtained by using agents in configuring metaheuristics which are distributed execution, remote execution, cooperation, and autonomy. The using of multi-agent concepts can be useful to self-organize particles in PSO using simple rules as defined by [116]. Their main idea was to define six states, which are cohesion, alignment, separation, seeking, clearance, and avoidance.

In terms of the multi-swarm design of PSO, [117] provided a multi-swarm and multi-best for the particle swarm optimization algorithm. They randomly split particles into multi populations. This algorithm updates velocities and positions of particles using multi-gbest and multi-pbest rather than single gbest and pbest. [118] proposed a novel variant known as Center PSO; it makes use of an extra particle as a center particle that controls the search direction of the entire swarm. Also, [119] built a Multi-swarm cooperative particle swarm optimizer based on a master-slave model; the slave swarms perform as a single PSO while the master swarm iterates depending on its knowledge as well the knowledge of the slave swarms.

This issue (the interaction between swarm intelligence and multi-agent systems) has been given much attention in the last few years in particular by the popularization of the swarm robotic field. In particular, [120] affirmed the concept of swarm appears nowadays closely associated with intelligent systems in order to carry out useful tasks. The author also analyzed qualitatively the impact of automation concepts to define intelligent swarms. Moreover, [121] have outlined the main characteristics of swarm robotics and analyzed the collective behavior of individuals in some fields. They affirmed that finite state machines are one of the most used adequate approaches to model this behavior. Another commonly used approach for this purpose is reinforcement learning.

## **2. PARAMETER CONTROL OF HETEROGENEOUS PSO**

Concerning heterogamous particle swarm optimization (PSO), attempts have been made to formalize the design of such multi-agent behavior of particles as a means to enhance the diversity of the algorithm or to achieve a trade-off between exploration and exploitation. A commonly used heterogeneous form of PSO is based on the idea of considering multi-swarms (multi-populations). It consists of dividing the whole search space into local subspaces, each of which might cover one or a small number

of local optima, and then separately searches within these subspaces. Another way to define multi-agent in PSO is to assign different roles to particles. Thus, different particles can play different roles, and each one of these particles can play different roles during the search processes. A challenging task within this PSO variant is how each particle has to decide which role it will assume.

In this chapter, we investigate the integration of the generic model to different types of heterogeneity that can be attributed to particle swarms, namely: the self PSO and the multi-swarm PSO with different cooperation rules. After controlling the particles configuration in the entire swarm level, we intend in this paragraph to control particles configuration on an individual level as well as in the case of multi swarms level.

## 2.1 SELF-STATE IDENTIFICATION FOR PSO

We apply the generic model to be associated with each particle movement. A Finite State Machine (FSM) is applied to model the decision making of an agent with the aim of guiding particles to move toward different promising sub-regions. To do that, swarm behavior can be represented as a finite state machine based on very simple agents and simple interaction rules. That is, a behavior specification defines a set of finite state machines, called options and a set of predefined behavior routines, called basic behaviors.

We integrate the HMM model described earlier as a probabilistic FSM [122] to learn and predict the most probable states of the probabilistic FSM in order to control particles behavior of PSO. This process is performed through the Viterbi algorithm that gives the most likely path of states for each particle in each PSO iteration. Each particle is viewed as automata having four finite states which are exploration, exploitation, convergence, and jumping-out.

HMM generic model is applied to address the self-state selection as the most common type of probabilistic FSM. Indeed, HMM has the ability to learn states of our automata from hidden observation based on the maximum like likelihood estimation [123]. This learning feature of HMM is used to control the particles individually across PSO iterations.

We associate to each one particle a Markov chain experiencing several finite states controlled by an HMM model. HMM associated with a particle can be viewed as a small machine learning guided by inputs and provides many outcomes that are state classification across iterations. We inspire from the generic model to adapt HMM model to fit each particle.

The HMM classification model comprises to determine one of four evolutionary states: exploration, exploitation, convergence, and jumping out to each particle at each iteration in order to give automatic intelligent control of the inertia weight. Our contribution to these works is to use HMM for each particle to recognize the appropriate state at each iteration. Therefore, we can produce the Markov Chain as identified.

We define each particle as a machine having different finite states. A particle of PSO can have many alternative traces for a given input because of the random behavior of the PSO algorithm. FSM associated with a particle is a little machine that feeds with inputs and provides many outcomes across iterations. We can say that the particle is associated with a random process.

During iterations, a particle is a probabilistic FSM related to a state  $\{x_i\}_{i \in N}$  that generates outcome or also called observation  $\{y_i\}_{i \in N}$ .

This definition yields to have at each iteration several groups of particles, each one plays a defined role according to its identified state machine. So, we can have for instance 40 particles, in which 20 particles explore throughout the search space, 10 others are exploiting, 7 are converging, and 3 are jumping out. Thus, particles are divided into sub-swarms with different states. The change of state or role of particles during iterations is governed by their associated probabilistic FSMs which is defined by the following formalism for each individual particle:

- Outcomes  $\{y_i\}, i \in N$
- State  $\{S_i\} i \in N$
- $A = (a_{ij})$  The state transition matrix:  $P(x_t = i \mid x_{t-1} = j) \quad i, j \in N, t : \text{iteration number.}$
- $B = (b_{jk})$  The emission probabilities of outcomes:  $P(S_t = k \mid S_t = j) \quad k, j \in N, t \text{ is the iteration number.}$

Our approach consists of finding the most suitable current state by finding the most probable trace for the given input of states across iterations. This problem constitutes the same HMM generic model to be resolved at a particle level.

**Algorithm 13:** self-adaptive inertia weight control by particle

```

Data: Position and inertia weight
Initialization: positions and weight  $w$ 
if state = exploration then
     $w_i = w_{min} + (w_{max} - w_{min}) * rand();$ 
else if state = exploitation then
     $w_i = \omega(l) = \frac{1}{1 + 1.5e^{-2.6l}} \in [0.4, 0.9] \forall l \in [0, 1]$ 
else if state = jumping out then
     $w_i = w_{max}$ 
else if state = convergence then
     $w_i = w_{min}$ 
end if
Return  $w$ 

```

According to the classified state, each particle has its associated state and can adapt individually its parameters. In our case, we apply the FSM state to adapt the inertia weight called selfHMM-wPSO and the acceleration factors called selfHMM-APSO. Algorithms for adaptation are the same as defined for the whole swarm but applied by a particle. The pseudocode is given below:

**Algorithm 14: SelfHMM-APSO algorithm**

```

Data: The objective function
Initialization: positions, velocities of particles, accelerations factors
and HMM parameters; Set t value to 0;
while (number of iterations  $t \leq t_{max}$  not met) do
    Update HMM parameters by EM process (Algorithm 5) ;
    Classification of PSO state by HMM classifier (Algorithm 4) ;
    Update  $\mathbf{c}_1$  ,  $\mathbf{c}_2$  and w values according to the corresponding state ;
    for  $i = 1$  to number of particles do
        compute f ;
        Update velocities and positions according to Eqs. (1) and (2) ;
        if (  $f \leq f_{best}$  ) then
             $f_{best} \leftarrow f$  ;  $p_{best} \leftarrow X$  ;
        end
        if (  $f(p_{best}) \leq f(g_{best})$  ) then
             $f(g_{best}) \leftarrow f(p_{best})$  ;
             $g_{best} \leftarrow X_{best}$  ;
        end
        if state = convergence then
            Elitist learning
        end
    end
     $t \leftarrow t + 1$ 
end
Result: pbest and fbest (the best particle and the best fitness)

```

**Algorithm 15: SelfHMM-wPSO**

```

Data: The objective function (F)
Initialization: iteration  $t=0$ , positions  $\mathbf{X}$ , velocities  $\mathbf{V}$ , inertia weight w,
HMM parameters ( $\mathbf{\Pi}, A, B, S, Y$ ), observations  $\mathbf{O}$ , states  $\mathbf{Q}$ , swarm size N;
while (number of iterations  $t \leq t_{max}$  not met) do
For  $i = 1$  to N do
     $O[t] \leftarrow I_i$  (Update observation sequence)
    ( $A, B, \mathbf{\Pi}$ )  $\leftarrow$  Baum-Welch( $\mathbf{\Pi}, O, A, B, \mathbf{\Pi}, S, Y$ ); (update HMM probabilities)
    ( $Q_i[1], \dots, Q_i[t]$ )  $\leftarrow$  viterbi( $\mathbf{\Pi}, O, A, B, \mathbf{\Pi}, S, Y$ ); (Classification of state)
     $w_i \leftarrow$  algorithm.13( $Q_i[t]$ ); (Update w value)
    -  $X_i(t) \leftarrow$  equation.1( $X_i(t-1), V_i(t-1)$ ); (Update positions)
    -  $V_i(t) \leftarrow$  equation.2( $X_i(t-1), V_i(t-1)$ ); (Update velocities)
    compute  $f(X_i)$  ;
    if (  $f(X_i) \leq f_{best}$  ) then
        -  $f_{best} \leftarrow f(X_i)$  ;
        -  $p_{best} \leftarrow X_i$  ; end
    if (  $f(p_{best}) \leq f_{gbest}$  ) then
        -  $f_{gbest} \leftarrow f_{best}$  ;
        -  $g_{best} \leftarrow X_{best}$  ; end
    if state = convergence then
        Elitist learning [60]; end
end
 $t \leftarrow t + 1$  ;
end
Return pbest and fbest (the best particle and the best fitness)

```

## 2.2 COOPERATIVE MULTI-SWARM

In this section, the machine learning algorithm hidden Markov model (HMM) is applied at an individual level (each particle) to model how the decision making of particles to choose the adequate sub-swarm to which it will belong. That is, HMM is used to learn and predict the most likely swarm, corresponding to each particle in order to control particles behavior of PSO. Hence, for each sub swarm, an associated role is given: exploration, exploitation, convergence and jumping out. Then, in a collective level of the swarm, a cooperative design is made to guide the search and move toward different promising sub-regions.

Each sub-swarm will adapt its own configuration of the parameters of its particles. Cooperation rules will be defined to ensure the information exchange between subs warms during the search process.

### 2.2.1 Sub-swarms constitution

In this approach, the swarm is divided to a sub-swarms in the objective to achieve a good trade-off between the population diversity and the convergence speed, and especially good management of the exploration and exploitation of the search process during execution in order to attain the best possible solution in the minimum number of iterations. Inspired from the definition of [60] of the evolutionary states for PSO, each sub swarm will group particles of a specific four evolutionary state that are: Exploration, exploitation, convergence, and jumping-out.

Then, each particle is viewed as a Markov chain having a state  $\{s_i\}_{i \in [1,4]}$ . During iterations, a particle can have a specific state  $i$  that represents its membership to a specific swarm  $i$ . Also, a particle can change the state from iteration to another and change consequently its corresponding sub swarm. So, a movement between sub swarms is indicated by the rows in Figure 18.

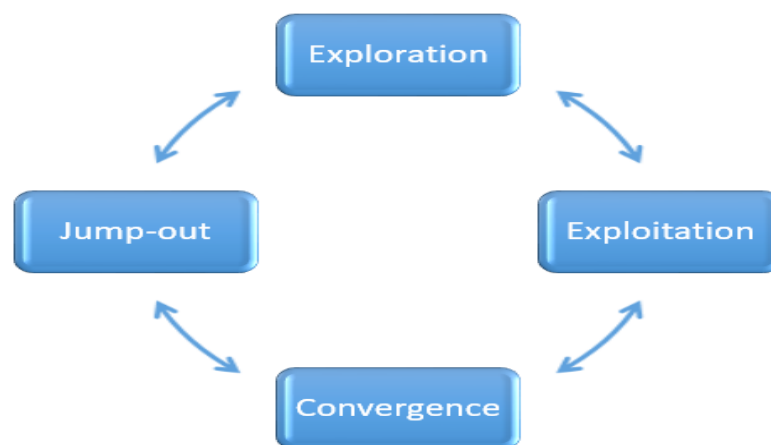


Figure 18 Sub-swarms and possible particle movements

To model the associated swarm of a particle, an associated markov chain with state  $\{S_i\}_{i \in [1,4]}$  is defined to each particle. However, the particle state cannot be perceived directly but only by observing some key parameters across iteration. Hence, a hidden markov chain is defined for each particle as the generic model.

A real-time state estimation procedure is performed to identify each adequate particle swarm: exploration, exploitation, convergence, and jumping out. It qualifies an automatic control of the sub-swarms.

## 2.2.2 Multi-swarms cooperation

To make use of the multi-swarm design given in the previous paragraphs, it is mandatory to set a cooperation model to make use of the search capabilities given by each sub-swarm. Two cooperative designs are chosen: A Master/Slave scheme and adaptive cooperation scheme.

### a. Master/Slave scheme

A master/slave cooperation model is chosen in this approach like in [119], where the slave swarms perform as a single PSO while the master swarm iterates depending on its knowledge as well as the knowledge of the slave swarms. In our case, the master swarm is the swarm associated with the convergence state. Then, the slave swarms will be those associated with exploration, exploitation and jumping-out states.

Each slave swarm with some  $n$  particles adapts itself according to its own evolutionary attached state separately. So, a slave swarm can be viewed as an independent swarm not connected to the other slaves. For the master swarm, the particles improve themselves not simply depending on the social knowledge of the master swarm but as well as that of the slave swarms. This notion is made by additional integrating a new dimension on the velocity of the particles in its velocity update. The equations for the velocity update of the master swarm will be:

$$V_i^c(t+1) = w V_i^c(t) + c_1 r_1 (pbest - X_i^c(t)) + c_2 r_2 (gbest^c - X_i^c(t)) + c_3 r_3 (gbest^s - X_i^c(t)) \quad (29)$$

Where C represents the convergence sub-swarm,  $c_3$  is called migration coefficient,  $r_3$  an uniform random sequence in the range  $[0, 1]$ ,  $gbest^c$  is the global best of the convergence swarm and  $gbest^s$  is the global best of the other slave sub swarms, in particular: exploration, exploitation and jumping-out.



Figure 19 Sub-swarms and the master/slave interactions

Figure 19 represents a communication scheme between sub-swarms. Then, the global algorithm of this approach is described in algorithm 16.

#### Algorithm 16: MsHMM-PSO

```

Data: The objective function (f)
Initialization: positions X, velocities of particles V, accelerations
factors of all four swarms; Set t value to 0;
while (number of iterations  $t \leq t_{\max}$  not met) do
for i = 1 to the number of particles do
    Decoding specific particle state (viterbi) ;
    Associate particle i to its decoded sub-swarm;
    Update w according to Equation (19) ;
    Update  $C_1$  and  $C_2$  values according to the corresponding state
    (algorithm 6) ;
    if convergence swarm
        Update velocities according to convergence Equation
    else Update position according to convergence Equation
    end
    Update positions according to position Equation
    compute  $f(x_i)$  ;
    For each sub-swarm i:
        if (  $f(X_i) \leq f_{\text{best}}$  ) then
             $f_{\text{best}} \leftarrow f(X_i)$  ;
             $p_{\text{best}} \leftarrow X_i$  ;
        end
        if (  $f(p_{\text{best}}) \leq f_{\text{gbest}}$  ) then
             $f_{\text{gbest}} \leftarrow f_{\text{best}}$  ;
             $g_{\text{best}} \leftarrow X_{\text{best}}$  ;
        end
        if sub-swarm = convergence then
            Elitist learning [60];
        End
    end
end
t  $\leftarrow$  t + 1 ;
end
Return  $p_{\text{best}}$  and  $f_{\text{best}}$  ;
Result: The solution based on the best particle in the population and
corresponding fitness Value

```

#### b. Adaptive cooperation

In this approach, the whole swarm is divided into four sub swarms that can conduct a heterogeneous search, but also can exchange information between each other, which can be relevant to explore a much larger space wherever the optimal model composition values may locate. In the presented multi-swarm cooperation variant, the population contains four sub-swarms, each with some predefined role. The sub-swarms maintain particular liaisons, which improves the search capacity. The cooperation mechanism chosen to be integrated with our generic model has been firstly introduced in [124]. As shown in Figure 20, this used mechanism is applied to maintain synchronization and cooperation between the four sub-swarms as identified by the HMM classification model in order to get the best update of the fitness values.



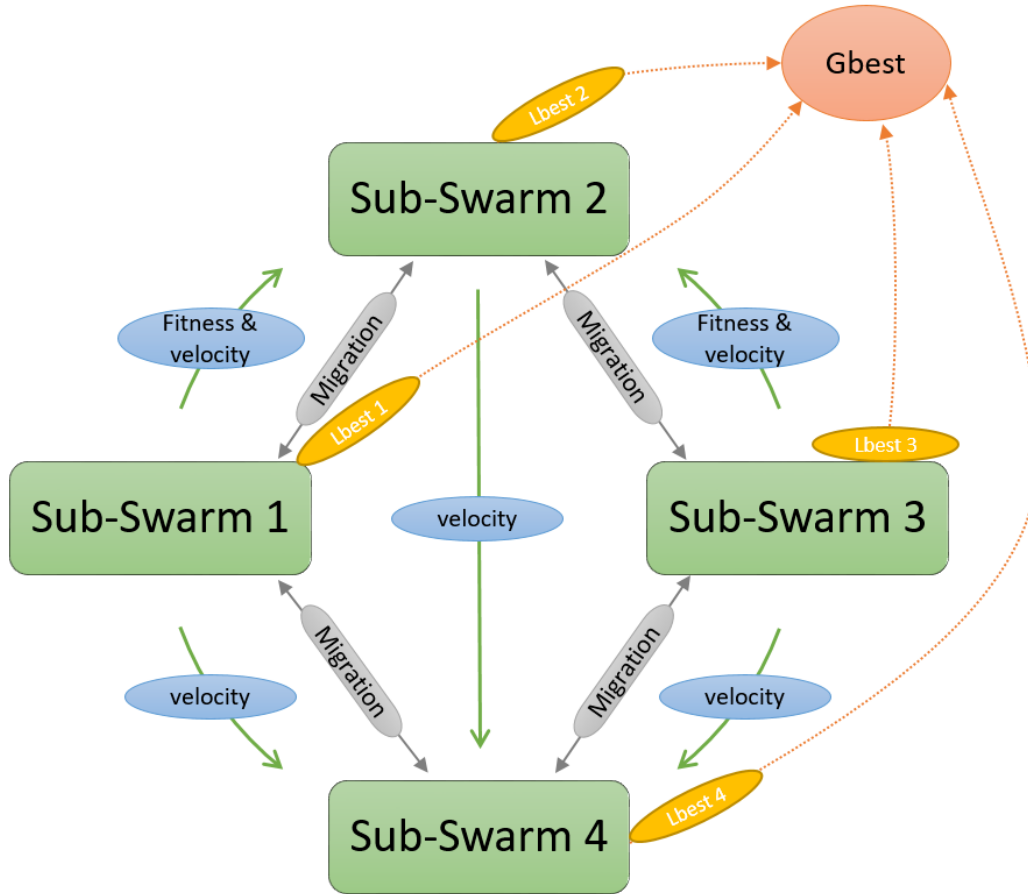


Figure 20 Sub-swarms and its cooperation diagram

In this general concept, each sub-swarm perform execution of a single PSO, comprising the update of position and velocity as stated by its own equations, shown in detail below. Once all the sub-swarms are positioned with the new generation, the global best is attained from the best local individual of each sub-swarm. The sub-swarm 1 and 2 are considered respectively as exploration and exploitation sub-swarms, and its positions and velocities are updated according to the original PSO equations (2) and (3) with the best parameters (inertia weight and accelerations) of the corresponding state. Additionally, the particle in the sub-swarm 3, identified with a convergence state, is updated depending on the fitness values and velocities of the particles in the sub-swarms 1 and 2 (Exploration and exploitation). The velocity of the particle in sub-swarm 4, identified with a jumping out state, is updated only with a combination of the velocities of particles of other sub-swarms. Other three control factors that are provided by the HMM generic model are applied to update the position.

The update equations of the particles in the sub-swarms S1 and S2 ( exploration and exploitation) are defined as follows:

$$V_i^{(1)/(2)}(t+1) = wV_i^{(1)/(2)}(t) + c_1r_1(pbest - X_i^{(1)/(2)}(t)) + c_2r_2(gbest^{(1)/(2)} - X_i^{(1)/(2)}(t)) \quad (30)$$

In the convergence sub-swarm S3, the particles can adapt the flight directions by learning from better particles in the two sub-swarm S1 and S2. Its velocity is updated as follows:

$$V_i^{(3)}(t+1) = w \left( \frac{\gamma}{\gamma_1} V_i^{(1)}(t+1) + \frac{\gamma}{\gamma_2} V_i^{(2)}(t+1) + V_i^{(3)}(t) \right) + c_1 r_1 (pbest - X_i^{(3)}(t)) + c_2 r_2 (gbest^{(3)} - X_i^{(3)}(t)) \quad (31)$$

where  $\gamma_1$  and  $\gamma_2$  are the fitnesses of current iteration in the sub-swarms 1 and 2 respectively.  $\gamma$  is the sum fitness of  $\gamma_1$  and  $\gamma_2$ . This represents a fitness monitoring methodology where the fitness values of the better swarm has more impact on the current particle.

The update equation of the velocity in sub-swarm 4 is described as follows:

$$V_i^{(4)}(t+1) = V_i^{(1)}(t+1) + V_i^{(2)}(t+1) - V_i^{(3)}(t+1) \quad (32)$$

The update in the jumping out swarm is designed to look for new areas based on the differences among the sub-swarms. The update equation can generate more likely solutions and explore new spaces. The position in sub-swarm 4 is updated as follows:

$$X_i^{(4)}(t+1) = \alpha_1 X_i^{(4)}(t) + \alpha_2 pbest_i^{(4)} + \alpha_3 gbest + V_i^{(4)}(t+1) \quad (33)$$

$\alpha_1, \alpha_2, \alpha_3 \in [0,1]$  are named the impact factors, which manage the impact of the historical information. In our approach, we calculate  $\alpha_1, \alpha_2, \alpha_3$  from the likelihood given by HMM of the corresponding state: exploration, exploitation, and convergence respectively. The likelihood of a state is calculated from the historical achievement of the swarm and gives a quantified probability to each state.

The control mechanism of inertia weight and acceleration factors are done as the same as in the homogeneous PSO approach, described as follows:

**Algorithm 17:** Adaptive acceleration update for swarms [60]

**Data:** Position and accelerations factors

**Initialization:** positions and accelerations factors  $\mathbf{c}_1$  and  $\mathbf{c}_2$  ;

**if** sub-swarm = exploration then Increasing  $\mathbf{c}_1$  and Decreasing  $\mathbf{c}_2$  ;

**else if** sub-swarm = exploitation then

Increasing  $\mathbf{c}_1$  and Slightly Decreasing  $\mathbf{c}_2$

**else if** sub-swarm = convergence then

Decreasing  $\mathbf{c}_1$  and Increasing  $\mathbf{c}_2$

end

**Return**  $\mathbf{c}_1$  and  $\mathbf{c}_2$

**Result:** Updated acceleration factors

**Algorithm 18:** Adaptive inertia weight control by the swarm

```

Data: Position and inertia weight
Initialization: positions and weight  $w$ 
if sub-swarm = exploration then
     $w_i = w_{min} + (w_{max} - w_{min}) * rand()$ ;
else if sub-swarm = exploitation then
     $w_i = \omega(l) = \frac{1}{1 + 1.5e^{-2.6l}} \in [0.4, 0.9] \forall l \in [0, 1]$ 
else if sub-swarm = convergence then
     $w_i = w_{min}$ 
end if
Return  $w$ 

```

The algorithm of the adaptive multi-swarm approach is described in Algorithm 19 below:

**Algorithm 19:** MsAHMM-PSO

```

Data: The objective function (f)
Initialization: positions, velocities of particles, accelerations factors
of all four swarms; Set t value to 0;
while (number of iterations  $t \leq t_{max}$  not met) do
for  $i = 1$  to the number of particles do
    Decoding specific particle state (viterbi) ;
    Associate particle  $i$  to its decoded sub-swarm;
    Update  $w$  according to Equation (19) ;
    Update  $c_1$  and  $c_2$  values according to the corresponding state
(algorithm 17) ;
    if swarm 1 ou 2
        Update velocities according to Equation of swarm 2
        Update positions according to Equation of swarm 2
    Else if swarm 3
        Update velocities according to Equation of swarm 3
        Update positions according to Equation of swarm 3
    Else if swarm 4
        Update velocities according to Equation of swarm 4
        Update positions according to Equation of swarm 4
    end
    Update positions according to Equation of positions
    compute  $f(X_i)$  ;
    For each sub-swarm  $i$ :
        if (  $f(x_i) \leq f_{best}$  ) then
             $f_{best} \leftarrow f(X_i)$  ;
             $p_{best} \leftarrow X_i$  ;
        end
        if (  $f(p_{best}) \leq f_{gbest}$  ) then
             $f_{gbest} \leftarrow f_{best}$  ;
             $g_{best} \leftarrow X_{best}$  ;
        end
    end
end
 $t \leftarrow t + 1$  ;
end
Return  $p_{best}$  and  $f_{best}$  ;
Result: The solution based on the best particle in the population and
corresponding fitness Value

```

## 2.3 EXPERIMENTATION

The three variants of heterogeneous PSO based HMM presented in the last paragraph will be evaluated by experimentation, and the corresponding results are given. Simulations are done on various benchmark functions: unimodal and multi-modal. Then, results are compared with other related variants from state of the art of PSO. The parameter setting is the same as the earlier one presented in the homogeneous PSO section as well as the presentation of the PSO related variants.

### 2.3.1 Performance evaluation

Firstly, we address the obtained results of all executions to compare the solution accuracy of our heterogeneous PSOs version based HMM. The best and the average values resulted from experimentations are given in Table 11 below:

*Table 11 Results comparison*

Functions		APSO	PSO	SimuA -PSO	Rand WPSO	YSPSO	SelfPSO	SAPSO	LinW PSO	MsPSO	SelfH MM- PSO	MsH MM- PSO	MsAH MM- PSO
$f_1$	Best	49	13566	115719	460.49	1420	16382	5095	9843	7067	14	6.03	2
	Mean	150	24618	288102	5677.4 9	2726	27998	14850	33384	20280	27	38	11
$f_2$	Best	0	5.1e-09	5.8e-06	0.00	0	8.6e-10	0	2.1e-12	0	0	0	0
	Mean	0	6.5e-05	0.04	0.0025	0	1.8e-05	0	3.6e-05	0	0	0	0
$f_3$	Best	0.01	26.55	93.19	0.00	6.77	36.01	18.20	37.7	17.66	5.74-6	0.0041	8.11-7
	Mean	0.05	50.15	188.56	0.05	13.77	63.59	31.96	64.09	40.53	5.79-5	0.0068	1.34-6
$f_4$	Best	0.02	94.97	251.36	0.14	23.72	30.47	37.21	110.48	51.24	5.74-6	0.0078	3.62-8
	Mean	0.05	135.82	396.21	0.94	42.31	77.51	84.79	246.78	95.74	5.01-5	0.0133	2.01-6
$f_5$	Best	16435	629e+5	587e+6	166679 .13	127e+4	839e+5	459e+5	102e+6	165e+5	223	2697	144
	Mean	67851	196e+6	210e+7	934189	843e+4	210e+6	166e+6	434e+6	155e+6	941	6684	423
$f_6$	Best	8.24	266.20	358.22	0.29	142.44	285.16	165.66	293.88	170.64	7.53	6.89	0.52
	Mean	16.14	307.24	462.02	56.81	175.84	315.60	273.31	322.35	261.77	8.55	14.70	2.87
$f_7$	Best	0.03	4.79	6.9436	0.03	3.1785	5.665	3.8279	5.403	4.7261	0.001	0.001	0.0003
	Mean	0.31	5.61	8.6336	0.22	4.2219	6.1951	5.2531	6.5613	5.6829	0.004	0.016	0.0008
$f_8$	Best	7.50e-5	0.17	0.52	0.00	0.05	0.27	0.13	0.25	0.14	1.6-7	1.5e-4	2.8-8
	Mean	0.01	0.39	0.87	0.04	0.12	0.41	0.25	0.45	0.31	0.003	2.1e-4	0.0006
$f_9$	Best	-118.35	-3e+28	-7.2+47	0.00	-3+34	-1e+308	-1+231	-	-1e+220	-118.35	-118.35	-118.35
	Mean	-118.34	-3e+28	-1e+47	3.83	-3e+33	-	-1e+230	-	-1e+219	-118.34	-118.34	-118.34
$f_{10}$	Best	-1	-1	-0.92	-1	-1	-0.99	-1	-0.94	-1	-1	-1	-1
	Mean	-1	-0.95	-0.74	-0.99	-0.98	-0.95	-0.97	-0.93	-0.96	-1	-1	-1

Table 11 shows the results obtained from the mean of all executions for the benchmark functions of heterogeneous PSO based HMM approaches which are: SelfHMM-PSO, MsHMM-PSO, and MsAHMM-PSO. When comparing HMM approaches against the states of the art of PSO variants, the corresponding results show a good solution accuracy in overall. The solution accuracy is enhanced for both unimodal (Elliptic, Step, Sphere, Tablet, Quadric) and multimodal functions (Rastrigrin, Ackley, Griewang, Schwefel, Drop wave). For Step and Ackley functions results are similar due to the functions simplicity. For other functions, the performances of the solution accuracy are largely increased. As for the Sphere and Tablet functions where the order of improvement attain more that  $10^5$  in the solution accuracy. When comparing between HMM heterogeneous approaches, MsAHMM-PSO is giving the best of results in the accuracy.

### 2.3.2 Convergence speed

In terms of convergence speed, comparisons are illustrated in figure 21.

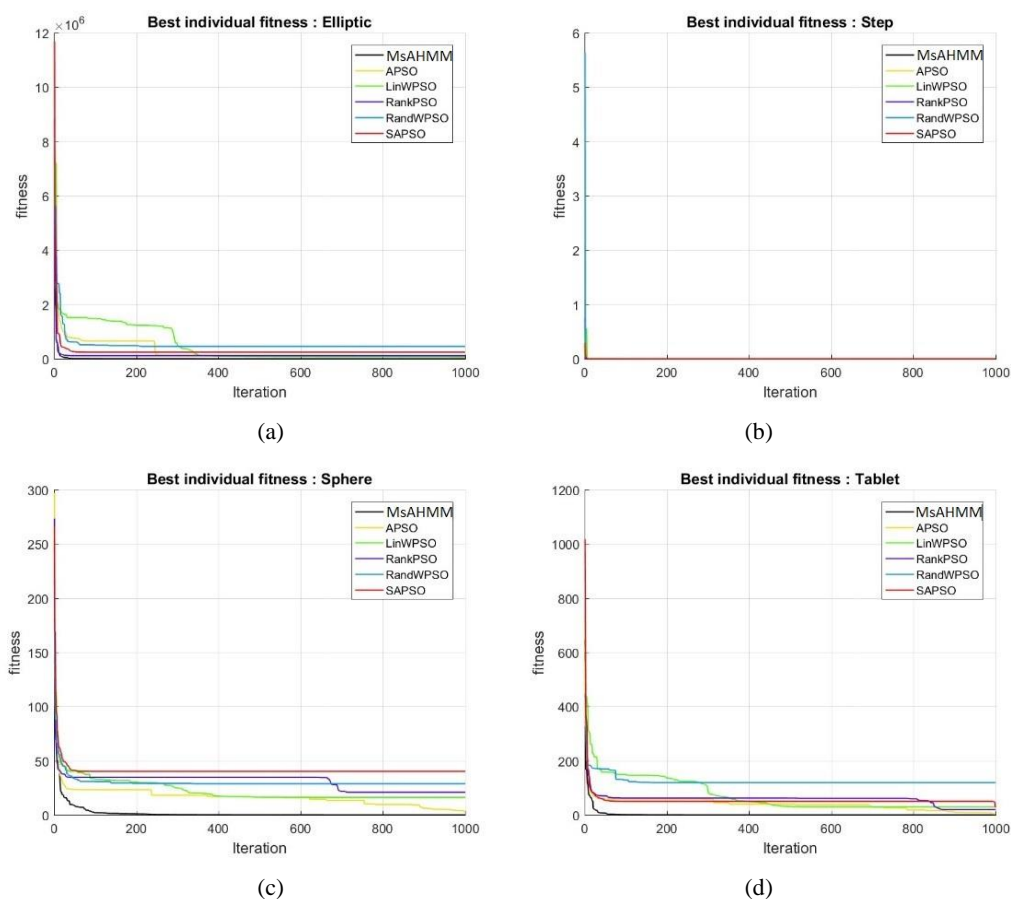


Figure 21 MsAHMM-PSO Convergence speed comparison

As shown in figure 21, the black line that gives the executions of MsAHMM-PSO results is under all other lines. Subsequently, MsAHMM-PSO provides a quicker convergence when compared to all diverse used PSO variants from the literature.

Given the exposed effective results of our approach, we can most certainly recognize that the multi-swarm cooperation based on hidden markov model supplies noticeably more significant performances for the PSO algorithm regarding the solution accuracy and the convergence speed.

### 2.3.3 Statistical tests

A more advanced comparison is performed using a statistical test, in particular parametric two-sided tests named t-test. The test is executed with a significance level of 0.05 between the MsAHMM-PSO and other PSO variants. The guidance of how statistical tests are performed, is described in the experimentation paragraph of the previous chapter.

*Table 12 T-test comparison*

Function	PSO	APSO	RandWPSO	RankPSO	SAPSO	LinWPSO	MsPSO	SelfHMM-PSO	MsHMM-PSO
$f_1$	0	0.0078	2.67e-06	0.0065	0	0	0.001	7.1e-07	0.0076
$f_2$	1	1	0.187	1	1	1	0.308	0.198	0.113
$f_3$	0	0.0015	0	0	0	0	0.0001	0.00051	0.061
$f_4$	0	0.0002	0	0	0	0	0.0164	0.0096	0.0043
$f_5$	0	0.0008	3.07e-06	0.0017	0.002	0.0012	0	0	0.0013
$f_6$	0	0.006	0	0.0092	0	0	0.0647	0.0043	0.0022
$f_7$	0	0	0	0	0	0	0.0084	0.0409	0.0054
$f_8$	0	0.007	0	0	0	0	0.194	0.204	0.108
$f_9$	0	0	0	0	0	0	0.219	0.669	0
$f_{10}$	0.0018	0.0023	0.872	0.41	0.388	0.0376	0.079	0.0074	0.022
<b>+1 (better)</b>	9	9	8	8	8	8	5	4	4
<b>0 (same)</b>	1	1	2	2	2	2	5	6	6
<b>-1 (worse)</b>	0	0	0	0	0	0	0	0	0

Given the revealed effective results of our approach, we could undoubtedly notice that the MsAHMM-PSO based cooperation scheme between swarms provides visibly higher performances to the PSO algorithm in terms of statistical tests. When compared to state of the art, MsAHMM-PSO gives more than 80% of best results, and less than 20% can be similar. Alternatively, when compared to other heterogeneous PSO approaches based HMM, namely MsPSO, SelfHMM-PSO and MsHMM-PSO, the MsAHMM-PSO is approximatively 50% better and 50% similar results. Indeed, the

application of HMM to improve the different type of heterogeneous PSOs has given good results.

### 3. CONCLUSION

HMM has been used in this chapter to control parameters of different heterogeneous cases of PSO. This constitutes a machine learning technique for online control of PSO search. It looks valuable from the view that HMM is a robust stochastic classification tool that takes into account past information about the population to control and adapt the algorithm in a heterogeneous way. Our multi-swarm approach is powered by an attached hidden Markov chain to each element of the swarm that provides swarm control of particle during the search process. According to each swarm, acceleration coefficients and inertia weight are updated. Then, the cooperation between swarms boosts the search more as in the master/slave cooperation scheme. The proposed heterogeneous PSO variants give better results than the state of the art of other PSO variants regarding both solution accuracy and convergence speed. Besides, statistical tests show good results compared with state of the art. We can deduct from the obtained results that associating a multi-swarm based machine learning with a cooperation strategy enhances PSO performances significantly.

The CPU time consumption has increased clearly when experimenting the proposed approach even more than the previous approach of Chapter 1. The associated machine learning has more computational effort in the multi-swarm design.

Future research should attempt the use of parallel computing that can significantly improve the time of simulation calculation due to machine learning CPU time computing. Also, the proposed approach needs to be evaluated on real complex optimization problem in different fields of application.

The next chapter will attempt to associate a reinforcement learning strategy to the multi-swarm design of PSO based HMM proposed in this chapter.

---



# CHAPTER III:

## MULTI-AGENT REINFORCEMENT LEARNING BY HMM-BASED PSO

---

After the online parameters control of PSO based multi-swarm given in the last chapter, this chapter will use of the good results and performances of a multi-agent system of the particle swarm optimization in the framework of reinforcement learning. Machine learning algorithms are usually classified into three main taxonomies: supervised learning, unsupervised learning and reinforcement learning (RL). RL is a machine learning approach that is based mostly on trial and error. In RL, an agent explores its nearby environment by making use of actions and obtaining rewards for these actions. The main target of the agent is to maximize its utility function that is depending on rewards. The intent behind this chapter is to present basic background information regarding RL as a first step to recognize the problem of RL in multi-agent systems to be used as a resolution methodology of the Markov decision process model. The Markov Decision Processes give a mathematical platform for decision making. They are usually intended to solve real-world problems related to planning and control as they are remarkably able to taking the basis of purposeful activity in a multitude of situations. For those factors, they have created the principle on which essential research in the subject of learning, planning, and optimization has been developed. As a consequence, several diverse methods have been produced for their solution. The principle of an MDP includes an immense range of models. However, this kind of generality shows up at a price. The definition gives too minor structure to produce successful MDP solution techniques. Instead of traditional dynamic programming resolution techniques, MDP can use reinforcement learning to build its solution. In this chapter, we start by Markov decision processes (MDP), a popular framework for designing agents that interact with their environment by performing actions and obtaining feedback signals revealing how good the actions are, learning how to solve a specific task by using these repeated interactions. Several solution methods are listed, along with diverse action selection strategies, which can be made use of by the agents within the learning process. Subsequently, we evaluate its weaknesses and present methods to fix them by Reinforcement learning techniques to cover an interesting class of real-world situations. In this way, we derive from the HMM-PSO algorithm a new reinforcement learning process to solve a range of decision problems typically known as sequential decision problems under uncertainty.



## 1. INTELLIGENT AGENTS

Agents formalism is becoming applied in a large range of applications currently. There is however no one unique definition of the terminology agent. For example, one standard definition is provided by Russell and Norvig in [125] they identify an agent as “anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators”.

Within many various definitions mentioned in the literature all over the years, we can differentiate the one offered in [126]. This is a rather general definition, but it still provides the specifications that an agent requires to meet in the context of this thesis; consequently, this is the one considered in this chapter.

A definition of an agent is a computer system, located in an environment, that is qualified of variable autonomous action in this environment to be able to satisfy its determined objectives [126].

By this definition, it is practical to realize that no particular environment is identified, and neither is the conception of objectives and also how the agents may attain their objectives. Intelligent agents are explained in [127] as agents that have to perform robustly in quickly varying, unpredictable, or open environments, where there is a high chance that actions will likely fail. As mentioned in [128] there are three features that an intelligent agent has to get in order to satisfy its assigned objectives:

- **Reactivity:** intelligent agents have the capacity to experience their environment, and react regularly to alterations that happen in it
- **pro-activeness:** intelligent agents can express goal-directed behavior by taking action;
- **social ability:** intelligent agents are able to interact with other agents.

If it is conceivable to assure that a particular environment is fixed, a goal-directed agent is specified simply to perform in this environment. However, in the real world, the environment is not stationary. Circumstances are continuously varying, information is often not detailed. For this reason, the risk of failure has to be considered. A *reactive* system is one that maintains continuous communication and interaction with its actual environment and replies to variations that arise in it (on time to be a beneficial response).

Our agents need to be more reactive; additionally, to operate in the direction of long-term goals. Hence, it is significant to obtain the best balance between reactivity and proactivity. Nevertheless, a certain amount of goals may only be accomplished considering the cooperation of other agents, and here social capacity is regarded. Agents will need to have the ability to have interaction with other agents located in the actual environment as a way to satisfy their associated objectives.

With the intention to learn from experience, the agents may perform by using supervised learning. Through the supervised learning, the agent is assigned instances of state-action pairs, and also knowledge regarding if the action was, in fact, either correct or incorrect. The goal of supervised learning is to generate a general policy

from the simulation instances, which is usually adequate to handle hidden cases. Therefore, supervised learning will involve a ‘teacher’ that can bring properly tagged instances. In contrary, reinforcement learning could be applied to problems where the knowledge area is possibly unavailable or expensive to collect [129]. It does not need past knowledge of right and wrong decisions; as a result, an agent has to actively explore its own environment to be able to monitor the impact of its own actions, wherever for each applied action, the agent obtains a mathematical signal implying in what way this action is appropriate or not. This trial-and-error interaction along with the environment is much more suited for the types of problems we are dealing with in this thesis; the upcoming section provides more complete clarification regarding Markov decision processes models and an offered Reinforcement Learning solution in the last section.

Agent-based systems concept is significantly applied for conceptualizing, building, and developing software systems [130]. Agents are autonomous computer programs which can be made to make use of intelligence to automatically carry out complex functions. Many of these systems are referred as Multi-agent systems (MASs). A MAS is a system that comprises several agents that have interaction together in an environment. This section introduces agent-related principles that allow the comprehension of MASs.

## **1.1 Agent Systems**

Generally, there are two categories of agent systems: centralized single agent, and multi-agent systems, as described in [131]. Regarding the centralized agent system, a single agent is in charge of making all of the decisions in the system, in contrast to the other agents that react only as slaves. This section examines both the basics of single-agent systems and multi-agent systems.

### **1.1.1 Single-agent Systems**

When it comes to single-agent systems, the environment’s interaction patterns are dependent on a single agent (central decision). Whenever exists additional agents inside the environment, they are regarded as slaves (agents who do not possess goals and pursue the guidelines of a central process).

### **1.1.2 Multi-agent Systems**

In a Multi-Agent System (MAS), numerous agents have interaction with each other in an environment. The elements of the environment (just like cells or obstacles in the maze problem) are regarded as passive agents since they do not require to learn goals, even while the actors (as an example the robot in a maze problem) who has got goals inside the environment are regarded as active agents [132].

Usually, the major difference among single-agent systems and multi-agent systems is the fact the environment’s dynamics are driven by some more than one agent in multi-

agent systems, while a central agent can determine the environment's dynamics in a single agent system [133].

In a common multi-agent of the PSO algorithm, there is an agent per particle. The composition of the agents (decision systems, learning algorithms or interaction capacities) can be either heterogeneous or homogeneous.

## 1.2 Multi-agent Systems Classifications

Based on the character of the multi-agent design, multi-agent systems are often referred by either cooperative or competitive systems. The objective of agents in cooperative multi-agent framework is to maximize an ensemble of utility functions when the objective of competitive agents is to maximize their particular utility [134].

On the other hand, receiving distinct (individual) utility functions would not necessarily claim competition, specially if these functions do not have any interdependencies. Once we consider maximizing one utility, it shows up at the charge of maximizing some other, and then we experience sensible competition.

This chapter is primarily concerned with cooperative multi-agent learning, instead of competitive learning.

## 1.3 Multi-agent Learning

The capacity to learn is a primary characteristic of intelligent agents. Various research analyses in artificial intelligence include the learning conducted by one agent [135]. In this kind of learning, an agent learns to conduct correctly in an anonymous dynamic environment [136]. Various methods have been offered for single-agent learning; nevertheless, in a multi-agent, setting up the learning environment is made up of many learners. An effective learner needs to consider measuring the existence of diverse learners in the same environment and the means they impact the dynamics of the environment [136].

### 1.3.1 Cooperative Multi-agent Learning Methods

Cooperative multi-agent models are systems in which agents make use of their particular interaction to cooperatively resolve affected tasks as well as maximizing some utility functions [137].

In multi-agent systems, agents may cooperatively learn a means to solve huge tasks that are very difficult for a single agent to learn in an affordable period of time. Hoen et al. [134] suggested that cooperative multi-agent learning methods can be split up into two main classes. Primary, team learning where a single learner is liable for learning the behaviors of the whole team. Secondary, concurrent learning where a learner for every team member is in charge of learning its behavior.

The literature of team learning has centered on the heterogeneity of the team individuals, while the study in team learning has centered on the interactions among concurrent learners [134].

### 1.3.2 Team Learning

Team learning is a technique to model the learning in a centralized single-agent system. In team learning, a single learner does apply a single-agent learning algorithm to learn a group of behaviors for an ensemble of agents. This strategy is an ordinary way to cooperative multi-agent learning; however, the possibility that a single agent learns the behaviors of all agents, produces the state space to be highly large. For instance, a team of  $n$  agents in an environment of  $m$  states and  $p$  actions per state provides a state space of as multiple as  $m^n$  states and an action space as multiple as  $p^n$ . [134] categorized team learning right into three partitions: homogeneous, heterogeneous, and hybrid learnings.

#### *a. Homogeneous Team Learning*

In homogeneous learning, a particular single learner finds out a single behavior which is taken from all other team individuals. This is available if all agents possess the precise same behavior. The state space of homogeneous learners is often small considering that all agents have the same behavior [138].

Homogeneous team learning could be utilized in problems where one agent does better [139], including problems having a large number of agents (swarm robotics). Additionally, problems that do not involve decomposition are appropriate for homogeneous team learning [140], just like the standard hunter prey problem or also the standard particle swarm optimization.

#### *b. Heterogeneous Team Learning*

A heterogeneous learner learns a particular behavior for every member of the team. Often the state space of the learner is significant and expands in complexity with the rise of the number of agents.

Heterogeneous team learning could be used in problems in which task expertise is needed, including robotic soccer [139] or to naturally decomposable problems [140], including air-traffic control systems and distributed decision.

#### *c. Concurrent Learning*

In concurrent learning, each agent is a learner and learns concurrently with the different learners through the multi-agent system. The primary goal, in this case, is to process the joint state space into  $m$  individual spaces. The issue of this kind of methodology is that the presence of multiple learners simultaneously may impact the stationary feature of the agents' environment [134]. In [134], the authors suggested that there are three most important tendencies in concurrent learning study. First, the study

that usually analyses the distribution of reward signal between team mates. Then, an investigation that analyses the cooperative adaptation of agents. Finally, the study on ways that each team member may model one another.

The next section will provide a well-known model of the agent learning formalism that is the Markov decision process.

## 2. MARKOV DECISION PROCESS

### 2.1 DEFINITION OF MDP

MDPs are one instrument of artificial intelligence (AI) that can be utilized to obtain optimal action policies within a stochastic area. It gives a mathematical framework for modeling decision making in conditions where outcomes are partially random and in part within the control of the decision maker. They came from the analysis of stochastic optimal control in the 1950s and have remained as of major importance in that area. Their particular principle has been expanded to develop across the last years to suit a wider range of problems and has produced a variety of common algorithmic concepts and theoretical research. In recent times, MDPs are applied in a multitude of areas, just like automated control, robotics, planning, economics and also manufacturing.

An MDP is a tuple  $\langle S, A, P, R \rangle$  where  $S$  is a finite set of states,  $A$  is a finite set of actions,  $P$  is a Markov transition model that represents the probability  $P(s'|s, a)$  of reaching a state  $s'$  when executing an action  $a$  in state  $s$ , and  $R : S \times A \rightarrow R$  is a reward function that provides the reward  $R(s, a)$  acquired after taking action  $a$  in state  $s$ . An agent's policy is defined as a mapping  $\pi : S \rightarrow A$ . The objective is to find an optimal policy  $\pi^*$  that maximizes the expected discounted future reward for each state  $s$ . We adopt that the MDP has an infinite horizon, and that the future rewards are discounted exponentially with some discount factor  $\gamma \in [0, 1)$ . The optimal action value function provides the expected discounted future reward for any state  $s$  when performing an action  $a$  and then following the optimal policy.

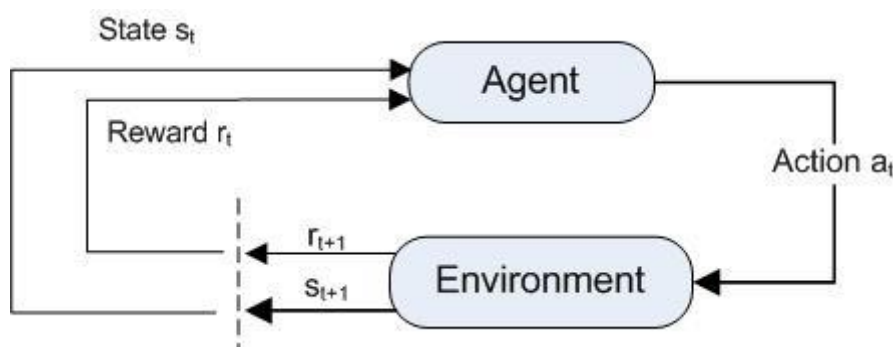


Figure 22 Markov Decision Process

A formalized description of an MDP (figure 22) is the tuple  $(S, A, T, R)$ , where:

$S$  is a finite set of states of the world.

- $A$  is a finite set of actions.
- $T: S \times A \rightarrow P$  is the transition function of states and  $P$  represents the transition matrix. To each action and state of the world, there is a probabilistic distribution over states of the world that they can be reached after executing the actions. The function  $T_t(s, a, s')$  is defined as the probability  $p_{t,s,s'}$  of reaching state  $s'$  starting in state  $s$  and given the action  $a$  and the time  $t$ .
- $R: S \times A \rightarrow \mathcal{R}$  is a reward function. Each action in each state of the world is assigned a real number. The function  $R_t(s, a)$  is defined as the reward of executing action  $a$  in state  $s$  at time  $t$ .

To solve the MDP problem, two main algorithms could be used to compute an optimal policy: value iteration [141] and policy iteration [142].

For the agent to be capable of maximizing the reward from its interaction with the environment, it has to be in a position to measure the value of a state and apply a mapping from states to probabilities of choosing each possible action at each time stage. This mapping is known as the agent's policy and is referenced by  $\pi$ , where  $\pi_t(s, a)$  is the probability that action  $a$  will be picked at time  $t$  if we are in state  $s$  at time  $t$ . The approximated value of a state is described in terms of future rewards that may be expected. Obviously, the rewards the agent can anticipate to acquire in the future rely upon what actions it will make. Consequently, its value function is identified with respect to a specified policy. As a result, the value of state  $s$  within policy  $\pi$ , denoted  $V^\pi(s)$ , is the expected return when starting in  $s$  and following  $\pi$  afterwards and could be defined formally as:

$$V^\pi = E_\pi\{R_t | s_t = s\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+a} | s_t = s\} \quad (33)$$

where  $E_\pi\{\}$  denotes the expected value provided that the agent uses policy  $\pi$  and  $\gamma$  is the discount rate which establishes exactly how much we appeal future reward contrasted to immediate reward.

Once we resolve an MDP, we are searching to attain the optimal policy, which is usually identified as the policy with an expected return higher than or equal to all other policies for any of the states. The optimal policy is denoted as  $\pi^*$ , and there can be even more than a single optimal policy. The MDP structure is abstract, adaptable, and supplies the methods required for the solution of various critical real-life problems. The overall flexibility of the framework enables it not only to be implemented for very diverse problems but even in various possibilities. For example, the time steps can involve arbitrary consecutive stages of decision making and acting. The actions can easily be any decisions we want, which makes the state consist of whatever it could be valuable in producing them.

## 2.2 FINITE-HORIZON MDP VS INFINITE-HORIZON MDP

Finite-horizon MDPs are a category of MDPs in which the performance (maximizing discounted reward) is needed to be maximized during a finite time horizon [143]. This

model is often applied when the agent lifetime is known earlier. Commonly, any discrete-time Markov decision process which usually has a finite number of decision stages is regarded as a finite-horizon process. Equation 34 explains that the agent maximizes its anticipated reward for the upcoming  $N$  steps.

As opposed to finite-horizon MDPs, infinite-horizon MDPs are appropriate for continuous tasks just where the behavior proceeds indefinitely, and the reward can be experienced at any time. The time horizon of infinite-horizon MDPs flows to infinity. Equation 34 displays that the agent maximizes its own expected reward with no mark of the endpoint ( $N$  step infinity).

$$V^\pi(s) = \left\{ \sum_{t=0..N} r_{(s_t, a_t)} \right\} \quad (34)$$

Finite-horizon and infinite-horizon problems, and their relations to discounted rewards can be reviewed in details in [144]. These principles were created from the MDP search area.

### 2.3 BELLMAN'S EQUATION

Bellman's equation of Dynamic programming [141] is applied in sequential decision problems to compute the maximum expected sum of discounted rewards for every state (equation 35). In MDP, the Bellman's formula is utilized to determine the utility of every state within a provided policy. An MDP of  $n$  states possesses a system of  $n$  simultaneous Bellman's equations, one particular for every single state.

$$U^\pi(s) = R(s) + \gamma \sum_{s'} T(s, a, s') U^\pi(s') \quad (35)$$

The Bellman's equation (equation 35) is linear while the Bellman's equation for optimal computing policies called Bellman optimality equation is a non-linear equation. The max function joined with the Bellman optimality equation renders the function a non-linear one. The max function formulates the process of choosing the action with the best possible return:

$$U^*(s) = R(s) + \gamma \sum_{s'} T(s, a, s') U^*(s') \quad (36)$$

Multiple varieties of this function are used in RL to formulate the utility functions of several RL algorithms.

### 2.4 SOLUTION METHODS

In this section, the main solution methods are presented to resolve a RL problem. The classic approaches can be categorized into two principal groups:

- Model-based approaches: make use of an explicit model of the environment to obtain the optimal policy.

- Model-free approaches: obtain the optimal policy not needing to explicitly offer the model.

In the next sub-section, Dynamic Programming is introduced, which needs a total and exact model of the environment and consequently is supposed to be a class of model-based approaches. The model-free RL-algorithms are a considerably more general framework for resolving MDP where transitions probabilities are not required to resolve the problem; reinforcement learning algorithms including Q-Learning will be provided in the next section.

### 2.4.1 Policy Iteration

Policy Iteration is a dynamic programming algorithm which usually manipulates the policy exclusively once used to calculate the optimal policy. It begins by evaluating an arbitrary policy, and after, makes use of the value function of that policy to get enhanced policies. This is carried out by taking into consideration a deviation from the recent policy in state  $s$  that identify if the policy is to adjust to deterministically select an action  $a$  distinct from the last one regarding the  $\pi(s)$ . If the value of this latest policy is higher than the existing policy  $\pi$ , then, simply, the policy is effectively improved. This procedure of producing a new policy that enhances an original policy, by producing it greedy or almost greedy with respect to the value function is known as policy improvement. As soon a policy  $\pi$  has been improved utilizing  $V^\pi$  to produce an enhanced policy  $\pi'$ , we can calculate its value function  $V^{\pi'}$  and maximize it once again to produce an even best policy  $\pi''$  wherever every policy is assured to be a strict progress above the past one. Since a finite MDP has just a finite number of policies, this process needs to converge to an optimal policy and optimal value function during a finite number of iterations. This method of interleaving policy evaluation with policy improvement is identified as policy iteration (algorithm 20) and is a principal algorithm in the review of MDPs.

#### Algorithm 20: The Classical Policy Iteration Algorithm

$\pi$  any policy

**While**  $\pi \neq \pi'$

$$\pi := \pi'$$

**For** all  $s \in \mathcal{S}$

    Compute  $V_\pi(s)$  by solving a system of  $|\mathcal{S}|$  unknowns

**For** all  $s \in \mathcal{S}$

**If** there exists an action  $a \in \mathcal{A}$  such that:

$$R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') V_{\pi_{t-1}}(s') > V_\pi(s)$$

**Then**  $\pi'(s) := a$

**Else**  $\pi'(s) := \pi(s)$

**Return**  $\pi$



### 2.4.2 Value Iteration

Value iteration is one other dynamic programming algorithm that uses a distinct procedure to attain the optimal policy. Instead of modifying the policy directly It depends on the direct solution of the Bellman optimality equation. For the task, an iterative procedure through value functions is designed; therefore it is known as value iteration [141].

The solution is proceeding by using the state space and affecting to each state the maximum estimated value depending on the discounted value of its own neighboring states. This iterative calculation is continuing until the maximum improvement in value for all states in every sweep is smaller than some predetermined small positive number denoted as  $\varepsilon$ . The smaller the value of  $\varepsilon$  the larger the accuracy of the algorithm is. Value iteration necessitates every state to be processed just once in every sweep within the state space and in that way takes away one of the drawbacks of policy iteration that is policy evaluation, which could call for multiple sweeps throughout the state space. In fact, many stopping criteria can be regarded to stop the iterations. The more basic one enables stopping when  $V_{n+1} - V_n < \varepsilon$ ; this triggers the formal pseudocode of the algorithm follows:

#### Algorithm 21: Value Iteration Algorithm

```

Initialize  $V_0 \in V$ 
 $n \leftarrow 0$ 
  repeat
    For all  $s \in S$ 
       $V_{n+1}(s) = \max_{a \in A} \{r(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_n(s')\}$ 
       $n \leftarrow n + 1$ 
    Until  $\|V_{n+1} - V_n\| < \varepsilon$ 
  For all  $s \in S$ 
     $\pi(s) \in \operatorname{argmax}_{a \in A} \{r(s, a) + \gamma \sum_{s'} T(s, a, s') V_n(s')\}$ 
Return  $\pi, V_n$ 

```

The Value Iteration algorithm is sufficiently flexible and does not necessitate the value of the states to be calculated in any rigid order nor equally many times to be able to converge on condition that all states are processed within the sweep [145]. This provides the flexibility that the values of states can be calculated in any sequence, applying any values of other states that show up existing; the value of some state can consequently be processed many times in one sweep. This flexibility, besides its slow convergence rate, has been the motivation of some works to accelerate its computations [146]. The majority of these attempts have been centered on one of two items, either parallelization or prioritizing of calculation in an attempt to decrease unneeded computation [147].

### 2.4.3 Other Methods

A multitude of other agent-based methods are available which could usually be applied for the resolution of MDPs. These techniques generally show up within two types, Temporal Difference Learning approaches and Monte Carlo methods. MC techniques are powered by averaging sample returns for the resolution of problems. They vary from Dynamic Programming because they do not involve an entire model of the environment and they do not bootstrap; however, alternatively the estimate for each state is independent. TD Methods, nevertheless, include qualities from the two method classes since they do not necessitate a full model of the environment and additional bootstrap. TD solutions are normally executed in an online and fully incremental manner. Two popular TD techniques are Q-Learning, and Sarsa [148], our interest in this chapter is given to the Q-learning approach, which is a new derived solution approach that will be presented in the next the section 3. Some of the used variants of MDP models are presented in the following sub-section, before its application in the next chapters in various airline transport problems.

## 2.5 MDP VARIANTS

### 2.5.1 Time-Dependent Markov Decision Processes

In standard previously defined MDPs, transitions and rewards are thought to be stationary functions; they do not undergo any change during decision epochs. In the literature, some approaches like [149] define Stochastic Time-Dependent Network where stochastic transition durations are included, but transition outcomes are deterministic. A model given by [150] is one of the first models to focus on time as an independent observable state variable; it is named as Time-dependent Markov Decision Process.

Time-dependent Markov Decision Process extends the Markov decision process model where a continuous observable time dimension is contained in the state space. The added time variable allows a more real representation of large problems with time-varying transitions or rewards. So TMDP includes problems with the following properties:

- State transitions are stochastic;
- Time-dependent action durations are stochastic.
- Rewards are Time-dependent.

In the TMDP model, each transition, which arises from making an action, is decomposed into a set of possible outcomes  $\{\mu\}$ . Every single outcome identifies both a transition duration and a resulting state.

The TMDP model decomposes each transition resulting from the application of action into a set of possible outcomes  $\{\mu\}$ . Each outcome describes a resulting state and transition duration.

Formally, the TMDP is defined as in [150] by:

- S: Discrete space state.
- A: Discrete action space.
- M: Discrete set of outcomes, of the form  $\mu = (s'_\mu, T_\mu, P_\mu)$  :
- $s'_\mu \in S$ : is the resulting space
- $T_\mu \in \{ABS, REL\}$ : identifies the type of the resulting time distribution (if it is absolute or relative)
- $P_\mu(t')$  (If  $T_\mu = ABS$ ): probability density function (pdf) over absolute arrival times of  $\mu$
- $P_\mu(\delta)$  (If  $T_\mu = REL$ ): probability density function over durations of  $\mu$
- L:  $L(\mu|s, t, a)$  is the likelihood of outcome  $\mu$  given action  $a$ , state  $s$ , and time  $t$
- R:  $R(\mu, t, \delta)$  is the reward associated to outcome  $\mu$  at time  $t$  with a duration  $\delta$

Figure 23 below, it shows a simple graphic representation of TMDP evolution.

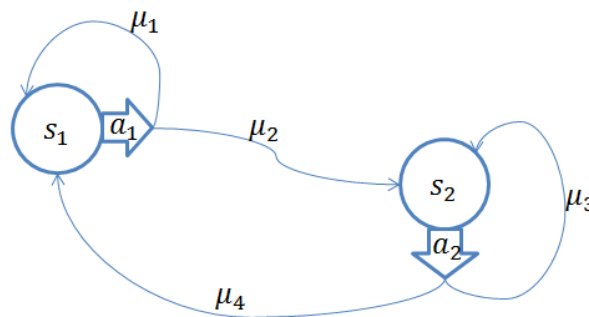


Figure 23 Elementary example of TMDP

In TDMDP and at time  $t$ , if in a state  $s_1$  agent executes an action  $a_1$ , it will generate outcome  $\mu_1$  by certain probability  $L(\mu_1|s_1, t, a_1)$  and an another outcome  $\mu_2$  by a probability  $L(\mu_2|s_2, t, a_2)$ .  $\mu_2$  represents the transition to  $s_2$  and  $P_{\mu_2}$  gives the transition absolute arrival time, while  $\mu_1$  represents the return to  $s_1$  (failure to leave  $s_1$ ) with a duration  $P_{\mu_1}$ . Implicitly, a waiting time is inserted before each action in the model.

The likelihood functions  $L$  govern possible outcomes in the model. Time distributions in a TMDP could be either “relative” (REL) or “absolute” (ABS) as shown as an example in Figure 24.



Figure 24 Representation of probability density function types

The TMDP model can be represented by the Bellman equations below:

$$V(s, t) = \max_{a \in A} Q(s, t, a) \quad (35)$$

$$Q(s, t, a) = \sum_{\mu \in M} L(\mu|s, a, t) \cdot U(\mu, t) \quad (36)$$

$$U(\mu, t) = \int_{-\infty}^{\infty} P_{\mu}(t') [R(\mu, t, t' - t) + V(s'_{\mu}, t')] dt' \quad (37)$$

(if  $T_{\mu} = ABS$ )

$$U(\mu, t) = \int_{-\infty}^{\infty} P_{\mu}(t') [R(\mu, t, t' - t) + V(s'_{\mu}, t')] dt \quad (38)$$

(if  $T_{\mu} = REL$ )

Where :

$U(\mu, t)$  : Utility associated to the outcome  $\mu$  in time  $t$

$V(s, t)$  : Time-value function of the immediate action

$Q(s, t, a)$  : Expected Q time-value through outcomes.

The resolution of this model is performed using Bellman equations 2 representing an undiscounted continuous-time MDP. At each state, the optimal time-value function is a piecewise linear function of time, which could be precisely calculated by value iteration [150]. The TMDP model is more general than semi-Markov decision processes [144] that have no notion of absolute time. With absolute time included in the state space, a comprehensive set of domain objectives can be modeled beyond the objective to minimize expected time, like for example the probability of designing a deadline. The variable time dimension may represent further quantities; it can consider planning with the non-linear utilities, or also with continuous resources.

## 2.5.2 Multi-Agent Markov Decision Processes

Multi-agent Markov decision process (MMDP) is developed by [151] to incorporate such numerous adaptive agents that interact and compute some given goals. MMDP has been applied in various domains as well as in air transportation (see [152]).

MMDP is the basis of full observability of the global state by every single agent; it is designed as a set of interacting learner agents, which are autonomous. These agents have to learn in order to cooperate and obtain their assigned goal. It can also be either centralized or decentralized in terms of decision-making main feature [153].

The multi-agent structure (see [151]) supposes having a centralized controller knowing all information regarding the system (Figure 25), including actions, the global state of the system, and rewards; thus the controller possesses the decision authority and keeps information distributed among agents.

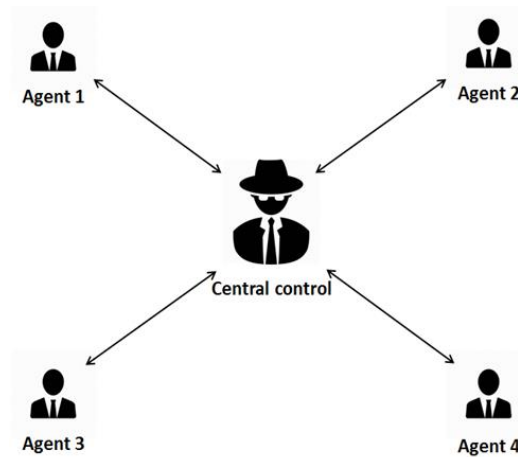


Figure 25 Centralized control in MMDP

To adapt the formalism of MDPs to cooperative multi-agent systems, Boutilier et al.[154] defined the Markov Decision Processes Multi-agent or MMDP (Multiagent Markov Decision Processes). These allow formalizing sequential decision problems in cooperative multi-agent systems. This formalism is very close to that of Markov games. However, only MMDP are modeling cooperative systems. The reward function is defined widely to all Agents, while stochastic games (Liviu Panait et al.[155]) define a function for each agent reward.

A MMDP is defined by a tuple  $\langle S, A, P, R \rangle$  as decision making conventional Markov. However, each action is described by the set of actions of individual agents, and then we talk about joint action. In addition to the tuple  $\langle S, A, P, R \rangle$ , a variable  $\alpha$  is defined. It corresponds to the number of agents in the system. So we define the MMDP by a tuple  $\langle \alpha, S, A, P, R \rangle$  such that:

- $\alpha$  : is the number of agents in the system.
- $S$  : corresponds to the set of states  $s$  in the system
- $A = A^1 \times \dots \times A^n$  : defines the set of joint actions of the agents,  $A_i$  is the set of local actions of the agent  $A_{g_i}$ .
- $P$  is a transition function; it gives the probability  $P(s, a, s')$  of the system goes into a state  $s'$  when agents run the joint action  $a \in A$  from state  $s$ .
- $R$  defines the reward function.  $R(s, a, s')$  is the reward obtained by the system when changing from one state  $s$  to a state  $s'$  by executing action  $a$ .

A MMDP can be seen as a MDP with a large state space and action. The set of agents is considered as a single agent whose goal is to compute an optimal policy for the MDP joint. A MMDP can also be considered as a stochastic game with  $n$ -player game in which the reward function is the same for all players. Formalism MMDPs corresponds to a generalization of MDP multi-agent case and specialization of stochastic  $n$ -player games.

Solving a MMDP is to calculate a joint policy  $\pi = \langle \pi_1, \dots, \pi_n \rangle$  where  $\pi_i$  is the policy of a local agent  $A_{g_i}$ . It defines a function  $\pi_i : S \rightarrow A_i$  that maps to any system having an action of the agent  $A_{g_i}$ . Such a joint policy can be calculated using a standard algorithm

as Value Iteration (This algorithm still also valid in the general case of decentralized agents, see [156]).

### 2.5.3 Time-Dependent Multi-Agent Markov Decision Processes

Multi-Agent notion can as well be combined with real-time value to include time evolution into the multi-agent system dynamics. A Time-dependent Markov Decision Process (TMDP) is provided by [150] to give this extension. This model is composed of stochastic state transitions and as well as stochastic time-dependent action durations. The actions in the TMDP model are stochastic and time-varying:

$$a(t) \sim \text{policy}(s, a(t)) \quad (39)$$

Resulting policies are actions to be performed by agents in every single time sequence. Then, the real planning window can be widespread to problems under uncertainty changing with time.

So, in this formulation as in [157], MMDPs consider an assignment centered decomposition approach, which is intermediate between the join MDP method and the method of independent agents. The centralized controller is adopted having the complete relevant information regarding the states of all agents to allocate jobs and assign jobs and resources to agents determined by a task level value functions associated with agents. After the jobs are allocated to agents, the particular lower level actions of agents are driven by the task level value functions until the primary controller reassigns jobs. Adding time dependence behavior will give a more realistic representation of the gate assignment problem, inspired by TMDP and coupled with the MMDP approach providing a new formalism of time-dependent Multi agent MDP.

Based on the two previous definitions of MMDP and TMDP, a new formalism is defined combining between those approaches. So, it is called Time-Dependent Multi-Agent Markov decision process TMMDP. This is an MMDP seen as cooperative multi-agent systems as in [154] or associated with time dependence capabilities as defined by [150]. MMDP is then extended to take a continuous observable time dimension contained in the state space. Supposing time variable is common between agents; a global time is associated to all agents.

A TMMDP is defined by:

- $n$ : Number of agents.
- $S$ : refers to the set of states  $s$
- $A = A^1 \times \dots \times A^n$ : The set of joint actions for the agents  $i$  is the set of local actions of the agent  $A_{g_i}$ .
- $M$ : Discrete set of outcomes, of the form  $\mu = (s'_\mu, T_\mu, P_\mu)$ :
- $s'_\mu \in S$ : the resulting space
- $T_\mu \in \{ABS, REL\}$ : identifies the type of the resulting time distribution (absolute or relative)

- $P_\mu(t')$  (If  $T \mu = \text{ABS}$ ): pdf (probability density function) over absolute arrival times of  $\mu$
- $P_\mu(\delta)$  (If  $T \mu = \text{REL}$ ): pdf over durations of  $\mu$
- $L$ :  $L(\mu|s, t, a)$  is the likelihood of outcome  $\mu$  given joint state  $s$ , time  $t$  and joint action  $a = (a_1, \dots, a_n)$ .
- $R$ :  $R(\mu, t, \delta)$  Reward attached to outcome  $\mu$  at time  $t$  for all agents with duration  $\delta$ .

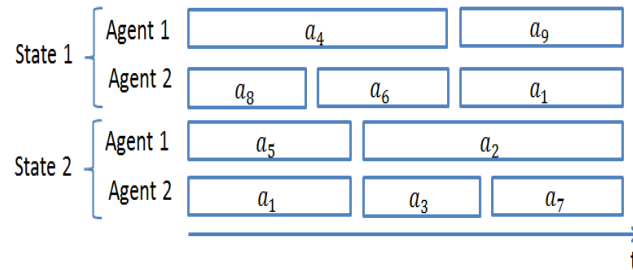


Figure 26 TMMDP policy representation

The aim of defining TMMDP formalism is to model and solve large real problems of planning under uncertainty taking into account either cooperative agent property and time evolution. Resulting policies are actions to be performed by agents in every time sequence (see Figure 26).

## 2.6 Limits of Markov Decision Processes

Various concerns limit the utilization of the MDP model and complicate its implementation. These kinds of limitations are interrelated to time and space requisites implementing the MDP model. The limitations comprise the curse of dimensionality, the memory requirement, and the stationary supposition of the problem model.

### 2.6.1 Curse of dimensionality

Obtaining an optimal policy for an MDP is polynomial in the multitude of states and actions; nevertheless, the number of states increases exponentially with the number of state variables, which in turn makes it computationally hard to solve large MDPs [158]. This issue is referred as the curse of dimensionality, which in turn is a critical concern in MDPs.

### 2.6.2 Memory Requirement

The modeling of an MDP needs the definition of the state and action spaces. Every single state/action pair involves a transition probability matrix and a reward function. The requested memory raises with the growth of the state and action spaces.

### 2.6.3 Stationary Assumption

In MDP formalism, the transition probabilities and rewards are supposed to be fixed over time; however, they would possibly not stay the same in the long run. These non-stationary features can be resolved by incorporating time into the state space or working with finite horizon MDP model. An additional solution is to incorporate some additional time depending variables in the state definition.

Multi-agent systems are usually non-stationary environments since many agents impact the environment at the same time [159]. This produces the learning policies of the agents to be non-stationary policies. Additionally, the behavior of an agent is influenced by the behaviors of the various other agents in the same environment [160].

### 2.6.4 Large Markov Decision Processes

Model-based approaches for large MDPs resolution are affected by the curse of dimensionality [141]. This is since the state space of large MDPs increases exponentially due to the number of state variables [161]. Moreover, MDP necessitates knowledge of transition probabilities of the dynamic system from a single to the subsequent state, which is in turn not practical to carry out large systems.

Different Reinforcement learning based approaches try to deal with Markov decision processes limitations when solving with dynamic programming. Several algorithms of RL have been proposed in the literature such as the well-known Q-learning algorithm. The next section will draw some RL fundamentals before addressing the proposed resolution approach in the last section of this chapter.

## 3. REINFORCEMENT LEARNING FUNDAMENTALS

An RL agent learns by trying actions and obtaining rewards for those actions. In contrast to the majority of machine learning paradigms, a RL agent attempts the actions to be able to explore the ones that generate the biggest reward.

Reinforcement learning is appropriate for online learning which includes agents that learn from interacting with their own environment. In interactive problems, it is hard and unlikely to grant the agents with all cases of possible behaviors to which they need to behave. Therefore, supervised learning, which is learning from instances provided by exterior sources, is not enough for learning from interaction [162].

### 3.1 RL BASIC MODEL

In RL, a learner interacts with its environment in the following means: the learner interprets the state of the environment and chooses an action appropriated to the state



utilizing its own decision-making function (policy). The action is therefore performed, and the agent gets a positive or a negative reward for its own action. Afterward, details regarding the reward of the state-action pair are utilized to update the agent policy (Figure 27).

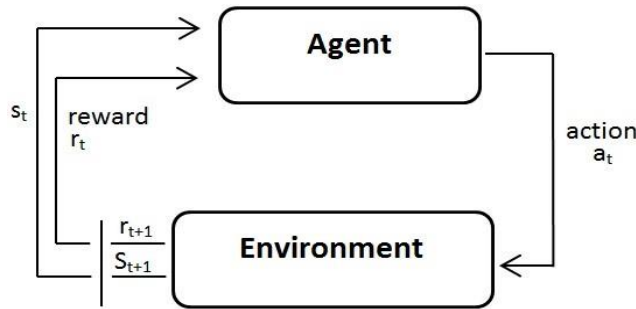


Figure 27: Reinforcement learning model [148].

The RL model is a 4-tuple associated with the basic fact that the problem model of RL is often formulated as a Markov decision process. A common RL problem comprises of [163]:

- A set of states,  $S = \{s_0, s_1, \dots, s_{n-1}\}$ .
- A set of actions,  $A = \{a_0, a_1, \dots, a_{m-1}\}$ .
- A reward model,  $R: S \times A \rightarrow R$ .
- A transition model,  $T: S \times A \times S \rightarrow [0, 1]$ .

The behavior of a reinforcement learner at any moment in time is identified by a policy that decides the optimal action to be selected at each state. A policy  $\pi$  is a mapping of environmental states into actions  $\pi: S \rightarrow A$ . A reinforcement learning agent looks for developing a policy which maximizes the sum of its rewards,  $R = r_0 + r_1 + \dots + r_n$ , for a problem that has a final state  $s_n$ , or a termination condition  $c$ .

The reward indicates that agent experiences are associated with the learning target. A reward function is a mapping of state/action pairs to numerical rewards  $S \times A \rightarrow R$ .

The value function is a utility function that presents the benefits of selecting a state overtimes to come. A value of a given state is an evaluation of the summation of rewards beginning with the given state.

It is essential to notice that the action does not commonly change the state of the environment. The objective of the agent is to maximize, at each time step  $t$ , the expected discounted gain:

$$R_t = E \left( \sum_{j=1}^{\infty} \gamma^j r_{t+j+1} \right) \quad (43)$$

Where  $\gamma \in [0, 1)$  the discount rate and the expectation are taken over the probabilistic state transitions within the actions selected by the policy of the agent.

The objective of the agent is to maximize its total of rewards ( $R_t$ ) while it is interacting with the environment.

## 3.2 POLICY TYPES

The policy that the agent uses has an essential role in the agent achievements to complete its task. A policy that is unconnected to time and does not involve the agent to hold memory is regarded as stationary policy, while non-stationary policy needs the agent to hold a memory. Policies might be categorized depending on the certainty of actions into two categories:

- Deterministic policies: The deterministic policy identifies a unique action for every state.
- Stochastic policies: a stochastic policy selects an action  $a$  from some distribution with a probability [148].

This thesis focuses on deterministic stationary policy in cooperative independent learners in Discrete-time MDPs. A discrete-time MDP has a finite number of decision stages where its overall performance is usually requested to be maximized over a finite horizon.

## 3.3 Action Selection Policies

The largely simple action selection policy is the greedy policy, which generally picks the action with the highest estimated reward to be performed. This policy is an exploitative policy. However, the objective of the action selection policies is to equalize among exploration and exploitation. The subsequent action selection policies seek to make it ensue [148]:

- $s$ -greedy: an action selection policy that picks the majority of the time the action with the top estimated reward. There is a small probability  $s$  that an action can be chosen at random.
- $s$ -soft: an action choice policy that picks the best action with probability  $1 - s$  and chooses a random action the remaining time
- Softmax: Softmax designates a weight to every single action as outlined by its action-value estimate. A random action is picked depending on its weight, which implies that the worst actions are less likely to be selected. Often, Boltzmann distribution [164, 165] is designed in Softmax. Given state  $S$ , an agent attempts an action  $a$  with a probability.

## 3.4 Exploration/Exploitation paradigm

Unlike supervised learning, a reinforcement learner has to directly explore its own environment to learn [166]. The trade-off between exploration and exploitation is a primary issue that appears in Reinforcement learning [167]. This issue is that exploration and exploitation of actions are linked. Firstly, a RL agent needs to exploit actions that have been attempted and identified to be greatly rewarded to be able to maximize its reward summation. Second, to be able to discover greatly rewarded actions, an agent needs to explore new actions.

In this section, our interest is given to the model free methods, especially the Q-learning algorithm.

### 3.5 Q-learning Algorithm

Q-learning is among the best-considered reinforcement learning algorithms that supply solutions for Markov decision processes. This algorithm features temporal differences to obtain mappings from state/action pairs to values. These values are recognized as Q-values and are determined using a utility function, named the Q-function, that returns the expected utility of choosing a given action in a provided state and pursuing a fixed policy afterward [168]. The simple fact that Q-learning does not involve a model of the environment is definitely one of its benefits.

The problem model of the Q-learning algorithm is an MDP model presented earlier which is made of a set of agents, a set of states  $S$ , and a set of actions  $A$  [169].

An agent that implements Q-learning requires a number of learning episodes to realize an optimal solution. An episode is normally a learning period that begins from a chosen state and terminates once a goal state is found. Within the episode, the agent selects an action  $a$  right from the set of actions  $A$  of its current state  $s$  depending on its selection policy. The learner later perceives the new state of the environment  $s'$ , and obtains a reward  $R(s, a)$  determined by the already executed action. The agent then upgrades its Q-table according to equation 44. This process repeats till the agent attains the target state, which will point the ending of the episode.

$$[s, a] \leftarrow Q[s, a] + \alpha(R(s, a) + \gamma \max_{a' \in A_{s'}} Q[s', a'] - Q[s, a]) \quad (44)$$

Where:  $R(s, a)$  is the reward of executing action  $a$  in the current state  $s$ ,  $a'$  is the action executed in the next state  $s'$ ,  $\alpha \in [0, 1]$  is the learning rate, and  $\gamma \in [0, 1]$  is the discount factor.

The fundamental output of the Q-learning algorithm is a policy  $\pi: S \rightarrow A$  which maximizes the sum of its discounted rewards  $R = \gamma(r_0 + r_1 + \dots + r_n)$  for an MDP that has a final state  $s_t$ , or a termination condition  $c$ . Algorithm 22 presents the pseudo-code of Q-learning.

**Algorithm 22: Q-learning**

```

Initialize matrix  $Q[S,A]$  with 0 values
observe initial state  $s$  from recorded data
repeat
 $a \leftarrow \pi(s)$  with  $\pi(s)$  being the policy to choose actions
perform action  $a$ 
observe new state  $s'$  and obtain reward  $r$  from the environment
 $Q[s, a] \leftarrow Q[s, a] + \alpha(R(s, a) + \gamma \max_{a' \in A_{s'}} Q[s', a'] - Q[s, a])$ 
 $s \leftarrow s'$ 
Until  $A_s = 0$  ( $s$  is a terminal state)

```

### 3.6 Multi-agent Reinforcement Learning

A Single agent RL approaches are commonly intended to solve stationary environments. In a multi-agent perspective, many agents influence the environment, and the actions performed by agents do not rely only on the environment but is also determined by what the numerous other agents are performing [170].

Moreover, making use of a single-agent reinforcement learning methodology to sizeable multi-agent systems is ineffective considering that the state and the action spaces of these kinds of systems are naturally so large. Single-agent method is improper for distributed problems just like air traffic transport problems [171].

In multi-agent structure, Q-learning can be implemented in an uncomplicated design to every single agent in a multi-agent system. This can be achieved by attaching a subscript to determine agents in the Q-function:

$$Q_i[s, a] \leftarrow Q_i[s, a] + \alpha \left( R(s, a_i) + \gamma \max_{a'_i \in A_{s'}} Q_i[s', a'_i] - Q_i[s, a_i] \right) \quad (46)$$

Where:  $R_i(s, a_i)$  is the reward that agent  $i$  obtains for executing an action  $a_i$  in the current state  $s$ .  $a'_i$  is the action executed by agent  $i$  in the following state  $s'$ ,  $\gamma \in [0, 1]$  is the discount factor and  $\alpha \in [0, 1]$  is the learning rate.

However, there are two limits for the aforementioned multi-agent Q-learning variant. First of all, it presumes that each agent chooses its actions individually from the other agents. Secondary, by using the same maxQ function of a single agent variant of Q-learning algorithm, it is not valid to be applied for the Value function [160].

In the past few years, various Multi-agent RL (MARL) methods had been offered to model reinforcement learning for multi-agent systems.

#### 3.6.1 Benefits of Multi-agent Reinforcement Learning

Multi-agent Reinforcement Learning methodology has many strengths over the single-agent RL methodology. The origin of the strengths of MARL is primarily due to the multiplicity of agents. The advantages contain: taking advantage of parallel

computation, sharing of knowledge among agents, robustness and appropriateness for distributed learning.

*a. Parallel Computation*

Parallel computing may easily accelerate the learning process of MARL algorithms once the agents take advantage of the decentralized composition of the task [172]. [173] suggested an approach for learning to coordinate verbal and non-verbal behaviors in interactive robots. In this procedure, a hierarchy of multi-agent reinforcement learners performs verbal and non-verbal actions in parallel. The research studies of [174] inspected the implementing parallel and distributed systems to MARL. In the proposed approach of this chapter, reinforcement learners in different units of the distributed system perform in parallel.

*b. Sharing of Knowledge*

The presence of multiple agents in the comparable multi-agent system is an opportunity for information exchange. In MARL, distinct categories of information can potentially be shared: sharing of sensory data, of episodes, and of learned policies [175]. Sharing of sensory data from one other agent is helpful if the information is relevant and enough for learning. When sharing of episodes involves sharing of the Q-values after an amount of episodes, whereas sharing of policies happens at the end of the learning process. In most cases, sharing of information boosts the learning process, if it is adequately employed.

*c. Robustness*

Multi-agent reinforcement system is innately robust considering that when a number of agents fail, the rest of the agents can carry out their assigned tasks. Reinforcement learners can easily be designed to react dynamically to undesirable situations. To illustrate, reinforcement learners that are dispersed in a distributed system can be informed once a host system is getting turned off; thus they will distribute and maintain work in another host environment [172].

### **3.6.2 Recent developments in Multi-agent Reinforcement Learning**

In this sub-section, we focus on some main studies on MARL: combinational RL algorithms and most of the widely known swarm RL algorithms.

*a. Combinational Reinforcement Learning*

Combinational RL algorithm is a kind of RL algorithm that combines much more than one RL algorithm to speed up the learning process by making benefit of the power aspects of each algorithm (Figure 28).

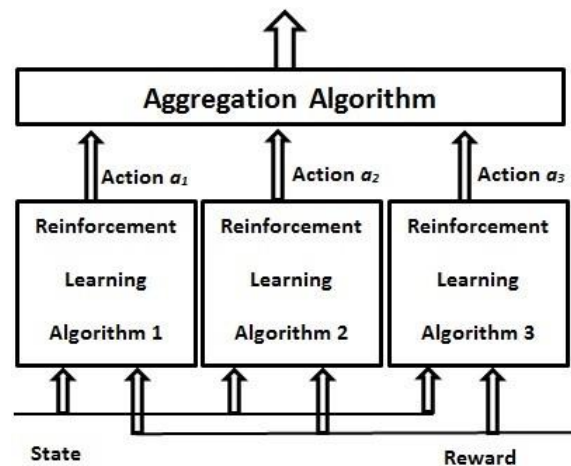


Figure 28 The AMRLS aggregation design [176].

Aggregated Multiple Reinforcement Learning System (AMRLS) was offered by [176]. It constitutes a sort of combinational algorithms. This algorithm aggregates Actor-Critic (AC),  $Q(\lambda)$ -learning and SARSA( $\lambda$ ). AMRLS involves two levels: learning and aggregation levels. The reinforcement learners can adhere to one of four performance ways when they are learning: synchronous, asynchronous, parallel or also serial execution ways. Each learner chooses an action for every state it goes to; after that, it transmits these actions towards the aggregation level. While in the aggregation level, the experienced actions for every single state are dynamically aggregated applying Majority Voting (WMV) or Weighted Borda Count (WBC) aggregation functions.

#### b. Swarm Reinforcement Learning

The swarm agent-based model is designed for use to model the learning procedure of cooperative independent learners. The studies like in [177, 178, 179] have made several works that inspected the inclusion of swarm multi-agent framework in reinforcement learning.

In [178], a MARL algorithm models the learning procedure of diverse independent learners. Through this algorithm, the learning procedure of independent cooperative reinforcement learners occurs in two levels:

- Independent learning stage: each learner works independently utilizing its own Q-learning algorithm till finishing each one learning episode.
- Q-value sharing stage: learners share their Q-values following a conventional Q-value update procedure (Q-value sharing process).

The research workers offered three interaction methods for sharing Q-values between independent learners:

- Best Q-value update procedure: The best Q-value of each state-action pair for all agents is chosen using the following update rule:

$$Q_i[s, a] \leftarrow Q_i^{best}[s, a] \quad (\forall i, s, a) \quad (45)$$

Where  $i$  is the agent identification number and  $Q_i^{best}$  best Q-value

The Q-learning algorithm that incorporates this sharing strategy is known as BEST-Q.

- Average Q-value update procedure: Each learner averages each Q-value in its Q-table with the best Q-value for each state-action pair. The update rule is:

$$Q_i[s, a] \leftarrow \frac{Q_i^{best}[s, a] + Q_i[s, a]}{2} \quad (\forall i, s, a) \quad (46)$$

Where  $i$  is the agent identification number and  $Q_i^{best}$  best Q-value

The Q-learning algorithm that incorporates this sharing strategy is known as AVE-Q (Average Q-learning).

In the next section, we are giving a new swarm reinforcement learning algorithm using the enhanced Particle Swarm Optimization from the last chapter to update and guide the Q-value optimal search.

## 4. PSO COOPERATIVE REINFORCEMENT LEARNING BASED HMM

In cooperative Q-learning, diverse independent learners find out the same task through the whole state space. The learning procedure of cooperative Q-learning often requires two stages. First, the individual learning stage, where every single learner independently utilizes its own Q-learning algorithm to progress its solution. Second, the learning through the interaction stage, where a Q-value sharing strategy is integrated. A Q-value sharing strategy enables independent learners to share their particular Q-values and make use of this information to attain new Q-tables. Sharing of Q-values among reinforcement learners speeds up the learning procedure of individual learners.

Reinforcement learning Swarm algorithms, such as [179], emerged as an appealing research sequel that handles the problem of Q-values formal update between diverse RL agents. In this section, a new swarm based cooperative learning RL is proposed based on the controlled particle swarm optimization algorithm of the second chapter.

### 4.1 Q-value HMM Sharing swarm Strategies

The PSO algorithm can incorporate the Particle Swarm Optimization (PSO) process to get a global optimal solution as provided by [178]. PSO is an optimization method that repetitively enhances a candidate solution relating to a qualitative measure [18]. PSO resolves decision problems that have diverse decision variables. As provided in the previous chapter, PSO as the swarm is a collection of particles that comprise a candidate solutions. Let  $D$  be a decision problem of  $n$  decision variables  $X = \{X_1, X_2, \dots, X_n\}$  that minimize an objective function  $f$ , and the size of the swarm as  $D$  to be composed of  $m$  particles  $\{p^1, p^2, \dots, p^m\}$ , then particle  $p^i$  of the swarm at iteration  $t$  is

$X_i(t) = (x_i^1(t), x_i^1(t), \dots, x_i^n(t))$ . The following two functions define the candidate solution of  $p^i$  at the following iteration  $t + 1$ :

$$V_i(t + 1) = w V_i(t) + c_1 r_1 (pbest_i - X_i(t)) + c_2 r_2 (gbest - V_i(t)) \quad (47)$$

$$X_i(t + 1) = X_i(t) + V_i(t + 1) \quad (48)$$

Where  $V_i(t)$  is the velocity vector of  $p^i$  at iteration  $t$ ,  $w$ ,  $c_1$ , and  $c_2$  are acceleration coefficients,  $w$  inertia weight,  $r_1$ , and  $r_2$  are random numbers between 0 to 1,  $pbest_i$  is the personal best solution of particle  $i$ , and  $gbest$  is the best solution discovered by all particles.

In PSO based reinforcement learning, the RL problem is modeled as an optimization problem in which the solution candidate and qualitative measure are respectively the Q-values and an evaluation function of the Q-values (Q-function) where the HMM model guides PSO parameters and search behavior as in the previous section. In this approach, the best Q-values of every single learner and the best global Q-values of all learners are utilized by each learner to update its Q-table.

The HMM model controls the PSO algorithm; the HMM state classification governs parameters and search behaviors.

The update of Q-value will be done as well according to the HMM classification as follows:

$$Q_i(s, a) = \delta_1 Q_i(s, a) + \delta_2 Q_{pbest_i}(s, a) + \delta_3 Q_{gbest}(s, a) \quad (49)$$

In the above function the weights  $\delta_k, k \in [0, 3]$  are named expertness and express the function of agent's relative expertness. In our approach, the formula for assigning a weight of expressiveness to agent knowledge by learner  $i$ , is using also the knowledge of all agents and controlled by the HMM classification. The weight  $\delta$  manage the impact of the historical information. In our approach, we calculated  $\delta_1$ ,  $\delta_2$  and  $\delta_3$  from the likelihood given by HMM of the corresponding state: exploration, exploitation, and convergence respectively. The likelihood of a state is calculated from the historical achievement of the swarm and gives a quantified probability to each state.

The Q-value update will also be state dependent and controlled by the search achievements.

## 4.2 HMM-QPSO ALGORITHM

According to classified state, each particle has it is associated state and can adapt its parameters individually as well as in the earlier described approach selfHMM-PSO. In our case, we apply the HMM state to adapt also the search of Q-value as well as inertia weight and acceleration factors of the PSO algorithm. Algorithms for adaptation are the same as defined for the whole swarm but applied to adapt the Q-value iteratively.

- First Stage: Q-learning

In the first stage, each learner individually incorporates its individual Q-learning algorithm to enhance its solution. The problem model of the Q-learning algorithm can



be described as Markov Decision Process (MDP), which has an agent, a set of states  $S$ , and a set of actions  $A_i$  for each state  $\in S$ .

An agent that applies Q-learning needs a number of learning episodes to find an optimal solution. An episode is a learning period that starts from a selected state and ends when a goal state is reached. During an episode, the agent chooses an action  $a$  from the set of actions  $A$  of its current state  $s$  based on its selection policy. The learner then perceives the new state of the environment  $s'$ , and receives a reward  $R(s, a)$  based on the previously implemented action. The agent then updates its Q-table based on the Q-function:

$$Q_i[s, a] \leftarrow Q_i[s, a] + \alpha \left( R(s, a_i) + \gamma \max_{a_i' \in A_{s'}} Q_i[s', a_i'] - Q_i[s, a_i] \right) \quad (50)$$

where  $R_i(s, a_i)$  is the reward that agent  $i$  receives for performing action  $a_i$  in the current state  $s$ ,  $a_i'$  is the action performed by agent  $i$  in the next state  $s'$ ,  $\alpha \in [0, 1]$  is the learning rate, and  $\gamma \in [0, 1]$  is the discount factor.

This process is repeated until the agent attains the goal state, which represents the end of the episode. The main end result of the Q-learning algorithm is a policy  $\pi : S \rightarrow A$  which maximizes the sum of its rewards  $R = r_0 + \gamma r_1 + \dots + \gamma^n r_n$  for an MDP that has a final state  $s_t$ , or a termination condition. The proposed HMM-QPSO algorithm does not require a model of the environment as [148].

The overall pseudocode of the proposed approach is bellowed:

**Algorithm 23:** HMM-QPSO algorithm

```

Data: The objective function
Initialization: positions, velocities of particles, accelerations factors
and HMM parameters, initial Q values; Set t value to 0;
while (number of iterations  $t \leq t_{max}$  not met) do
  Update HMM parameters by EM process (Algorithm 1) ;
  Classification of PSO state by HMM classifier (Algorithm 2) ; Update
c1 , c2 and w values according to the corresponding state ;
  for i = 1 to number of particles do
    compute f as Q values;
    Update  $Q_i[s, a]$  by performing Qlearning for one episode for agent i
    Update velocities and positions according to Eqs. (47) and (48) ;
    if (  $f \leq f_{best}$  ) then  $f_{best} \rightarrow f$  ;  $p_{best} \rightarrow X$  ;
    end
    if (  $f(p_{best}) \leq f(g_{best})$  ) then
       $f(g_{best}) \leftarrow f_{best}$  ;
       $g_{best} \leftarrow X_{best}$  ;
    end
    if state = convergence then
      Elitist learning
    End
  End
  Update  $Q_i[s, a]$  by equation.50
   $t \leftarrow t + 1$ 
end
Result: The best Q values for optimal action state selection

```

The next sub-section will give an experimental study to improve our method.

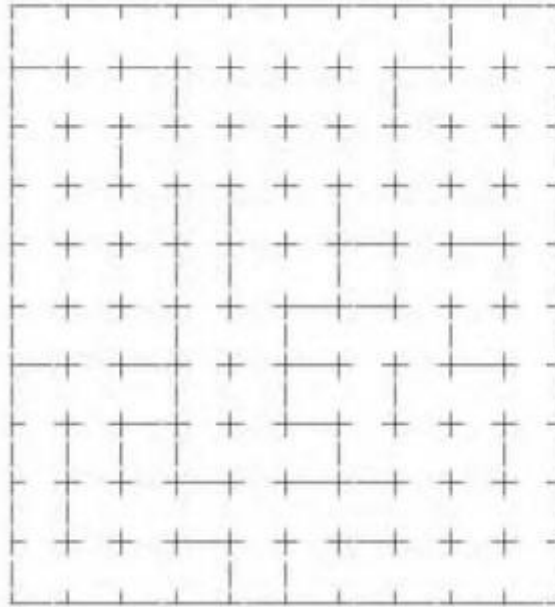
## 4.3 EXPERIMENTATION

### 4.3.1 Experience settings

This section shows the results of the  $10 \times 10$  grid (Figure 29).

The performance of the cooperative Q-learning algorithms based HMM-PSO and other Q-learning based algorithms were experimentally investigated. In the tests, an algorithm is thought to acquire convergence once the average number of steps in its policy enhances by fewer than one move through 200 successive episodes.

In this simulation, five agents learn for the exact number of learning episodes just before sharing their particular Q-values. The impact of the occurrence of information sharing influences the overall performance of cooperative learners. The agents share their very own Q-values after every single learning episode.



*Figure 29 Example of grid world for  $n=10$*

The number of learning episodes is 1000, the discount factor  $\gamma = 0.9$  and the learning rate is  $\alpha = 0.01$ , in the Q-learning algorithm (like in [180]). A learning episode terminates once the learner attains the goal cell or soon after 1000 steps with no getting to the goal cell.

### **4.3.2 Experimental Results**

The performance of the HMM-PSO based Q-learning algorithm and other Q-learning algorithms were investigated against each other following experimental models.

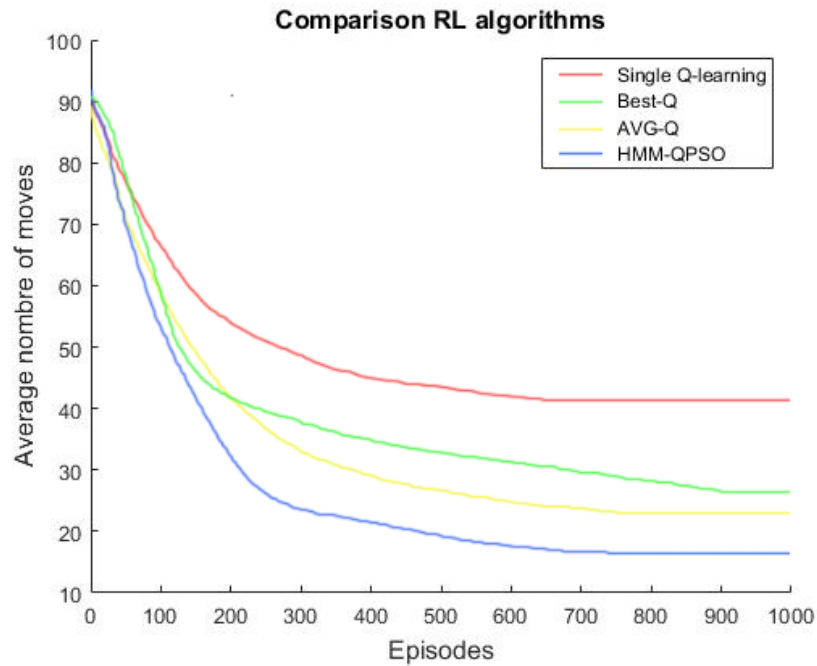


Figure 30 Comparison of Reinforcement learning algorithms

In the experiments (shown in Figure 30), an algorithm is said to have converged when the average number of moves in its policy improves by less than one move over 200 consecutive episodes. The comparison is shown in the average number of moves per one episode in a  $10 \times 10$  grid. HMM-QPSO gives the best results.


## 5. CONCLUSION

In this chapter, the multi-agent design of the particle swarm algorithm, controlled by a hidden markov model, has been used as a reinforcement learning technique for markov decision model resolution. The idea of using metaheuristics for reinforcement learning is not completely new, but the use of online controlled particle swarm by the supervised machine learning (HMM) enhance the optimal Q values search. The overall results of the HMM-QPSO suggest that each of the cooperative Q-learning algorithms performs better than single agent Q-learning when Q-value sharing performed. This is because learners with less experience incorporate knowledge of more experienced agents, giving a better average move number than single-agent learning. Also, the online controlled PSO based cooperative Q-learning gives best results; this is due to the cooperative aspect of PSO and the HMM integration to enhance the Q-value search.

This chapter presents a preliminary result of the proposed reinforcement learning method that is the HMM-QPSO. Experiments have been performed on the classical robotic problem of the optimal path in a labyrinth of  $10 \times 10$  grid. Even if the result was notably better than the other Q-learning variants, the approach needs to be compared using more competitive and complex problems. On the other hand real word

applications are required to ensure the performances of this proposed RL algorithm. Chapter 5 will present an application of the HMM-QPSO to an airline transport problem that is the maintenance routing problem; the approach is therefore investigated in an air traffic problem.

---



# CHAPTER IV: STOCHASTIC MODEL OF THE GATE ASSIGNMENT PROBLEM

---

With airports being busier and often troubled with insufficient capacity, the efficiency of the airport's resource usage becomes increasingly more imperative all across the world. The efficiency can be upgraded by taking under consideration the flight operations disruptions, which historically were not considered into planning. More efficient resource utilization will not only smooth airport operation but even require to predict unwanted disruption and manage its negative effect. The gate assignment problem is considered one of the vital airport operations, which is typically solved with no consideration of some random disruptions, such as delays that frequently arise in the flights schedule. Modeling and study of uncertainties in flight operations which may enable the possible disruption information to be evaluated in the allocation planning earlier along with the design of a new type of solution methods are revealed. It is noticed that when further information from the flight operations is integrated into the assignment planning process, the number of estimated conflicts in flight operations can be estimated. This might effects a smoother airport operation at the time of airline operations. A new approach of dealing with the gate assignment problem is presented in this chapter based on a stochastic model, that is, the Markov decision process.

The original algorithm of MDPs is considered in this chapter, providing the modeling background to solve the gate assignment problem under uncertainty. In this offered resolution methodology, we consider the stochastic aspect as a result of flight delay and with the consideration of different further constraints of aircraft size in the assignment. The main aim of this chapter is to provide an adaptive planning model of GAP; this concerns a planning/learning approach in order to create an effective approximation of state-dependent uncertainties. That will enable supplying a new model for the GAP based on MDPs. This work aims to grant to controllers at the airport a robust priory solution rather than taking the risk of online schedule modifications to manage uncertainty. The solution of this problem will be a set of optimal decisions that should be adopted in the matter of traffic disturbance. Primary experimental results on a sample of real-life data illustrate the feasibility and efficiency of our approach for managing uncertainty.

## 1. The gate assignment Problem

Considerable interest has been given in recent years to the techniques for managing and allocating airport and airline resources effectively and efficiently in a dynamic operational environment. This is due to the growth of air transport traffic (doubled since the early 1980s). The scheduling problems nowadays faced by airport and airline managers have led to complex planning problems that require new models and methods. This is firstly caused by the wide range of resource modules that apparently have to be considered like flights, terminals, crews, baggage etc, and they are highly interdependent. In the real world, we also have stochastic variations in air traffic that increase problem complexity, which is more considered in the latest researches.

### 1.1 Problematic

The major task of an airport is to ensure a smooth flow of flights traffic. Figure 31 depicts an example of gating at Mohamed V airport, where arriving aircraft are assigned to terminal gates. This is guaranteed by an optimal assignment of aircraft to their suitable gates. In fact, if not assigned yet, an aircraft will wait obligatorily on the ramp or even in the air. Such situations are undesirable more than the problem of gate capacity because of time consumption and also the limited capacity of ramps and adjacent airspace.



*Figure 31 Example of gating at Mohamed V airport*

Thus, Airport gates are considered as expensive and scarce resources in air transportation. Increasing the resource supply by involving a time-consuming and costly redesign of terminal buildings or ramps is usually not feasible in the short term. It is therefore of great importance to an airport to exploit the available gates as best as possible.

Flight gate scheduling problem relates to assigning different aircraft activities (arrival, departure, and parking) to different aircraft gates. This is, consequently, an essential concern in the daily operations of an airline. They have a major impact on maintaining the efficiency of flight schedules and passenger satisfaction. Some of the factors that

impact the assignment of gates to arriving flights include passenger walking distances, aircraft size, baggage transfer, aircraft rotation, ramp congestion, and aircraft service requirements, etc. In the real world, the deterministic solution is infeasible due to the stochastic aspect of the problem.

In real life applications, the arrival and departure times are not certain, and it is important to take the uncertainties of these input parameters into account. This is why one of the most critical challenges of gate scheduling is to build a gate scheduling that can be robust against stochastic variations of input data; in other words, schedule flexibility is necessary. The uncertain in gate scheduling can be caused by flight or gate breakdown, flight earliness or tardiness, emergency flights, severe weather conditions, errors made by staff or for several other causes. For example, delayed arrival of one aircraft may generate a series of delayed arrivals for other aircraft that have been allocated to the same gate. In the worst case, this may result in what is called "domino effect" and finally require an entire rescheduling; this type of scenario is highly undesirable.

## 1.2 Problem statement

The gate allocation problem (GAP) is the task of getting suitable positions to park aircraft at an airport. In this section, a real-world example is used to clarify the problem. Mohammed V Airport of Casablanca is used as a case study airport to demonstrate the scale of this problem, the specifications which have to consider by the currently utilized systems, and the manner in which the problem is handled.

The problem of assignment to gates is often divided into two parts. The first includes a monthly schedule. Each month, the airline operators declare a list of flights they made to plan with the assignment of flights for every day of this month. So, each flight is associated with a specific gate. Once the first assignment has been made and if a change of time has occurred, it was necessary to wait for the necessary resources (i.e., gates) to be available for the traffic. Those situations of irregularities involve extra irregular equipment or repair. For the first part of the problem, the resolution time is not a critical factor. However, the second part of the problem occurs when the planned schedules are altered because of a non-anticipated events such as the bad weather, mechanical defects, and late arrivals. Then, the controllers have only a few minutes reorganize those sort of disruptions; So that the current assignment of flights has to accommodate all arriving flights. The emergency replaces the goal that needs to be changed in real time.

The problem of assigning gates to arriving flight is a crucial decision problem in daily operations at most airports around the world. Heavy competition between airlines and the rising demands of passengers for higher comfort has produced measures of quality in decisions at an airport as significant performance indices of airport management. For this purpose, mathematical modeling of this problem and the application of advanced operations research and machine learning methods to solve the GAP problem have been examined largely in the literature. The overall characteristics of busy international airports generally require serving a large group of different airlines, a big number of daily flights, and covering several types of aircraft.



The next subsections will provide a description of an airport, its likely configurations, and its assigned missions.

### 1.2.1 Airports Description

We will present and clarify the components that construct an airport, and the activities that are inherent to it. Airport operations vary from landing aircraft to take-off as displayed in Figure 32; it depicts the area of interest given by this chapter. Once an aircraft lands, it taxis into a ramp area and parks at a gate. While the aircraft is docking at the gate, passengers disembark and board the plane. Whenever the aircraft is in position to depart, it is taken back and taxis out to a runway. After that, the aircraft takes off the airport. Involving these operations, this study concerns the optimization of ramp operations that impact directly on the accommodation of passengers.

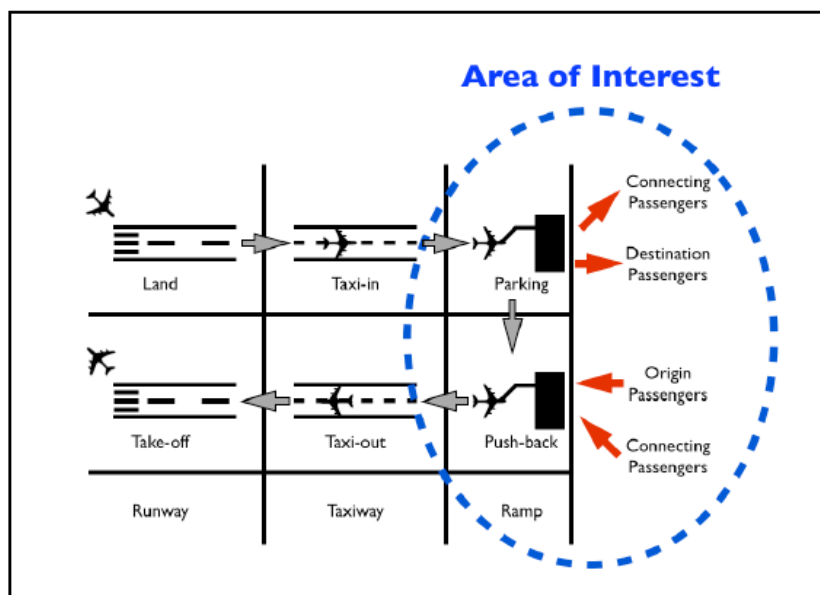
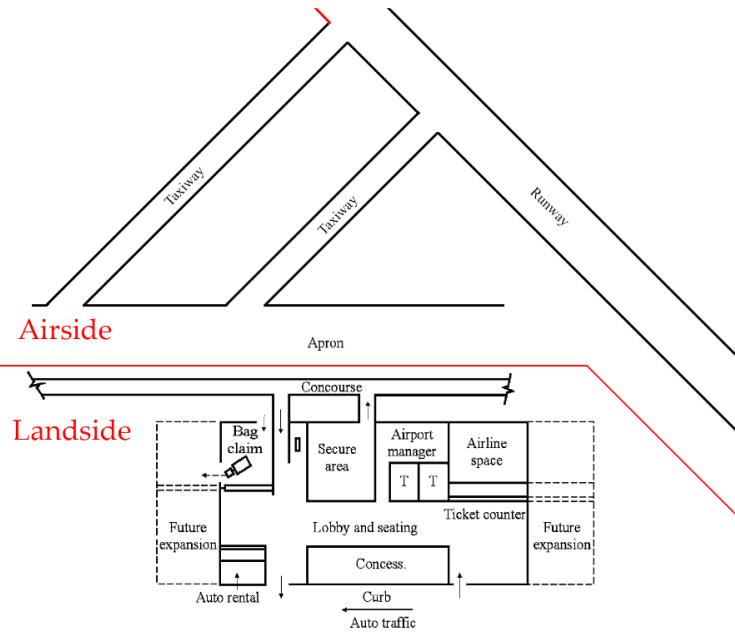


Figure 32 Area of interest of the gate assignment problem.

To acquire a better understanding of the airport functions, certain elements should be considered. From an operational perspective, an airport is the sum of many operations, infrastructures, agents, facilities, and equipment needed to enable airlines to take off, land, supply, maintenance and shelter the aircraft. The operations and sub-operations that provide the ways for passengers and freight to pass on from area to air modes of transport; and the sub-operations that offer more enjoyable passage through the airport (including shops, restaurants, etc.). Nevertheless, to depict an airport today, the definition is always partial, on account of a set of increasing activities that have been attaining importance over the years [181].

The configuration of an airport is mainly divided into two different sections, as it is noticed in Figure 33: the airside and the landside.



*Figure 33 Schematic presentation of an airport*

The airside is the part of the airport dedicated to aircraft activities including take-offs and landings and loading and unloading. The airside consists of all the areas available to the aircraft and is consisting mainly of [182]:

- Runways: regions for the landing and take-off of aircraft.
- Taxiways: circulation areas related to the runways to other airside services. Keeping a decisive effect on the capacity of the runway system, they require to enable aircraft to operate safely and fluently.
- Apron: the part that affords the interchanges among landside and airside services. They can be categorized as cargo building aprons, passenger building aprons, general aviation aprons, service, and hangar aprons or long-term parking aprons and remote aprons.
- Support Services: qualified to offer all the assistance required by the aircraft to perform their activity.
- All the infrastructures that are necessary for the passengers to cross from the surface to inside the airplane.

The landside combines all the areas usable for departure, arrival and transit passengers to arrive and accomplish their goal destination. Indicating that for departure passengers, it involves all the infrastructures, they are required to use and pass through to attain their gate; for transit passengers, the ways for them to go to their next flight; and for arriving passengers. All the areas they need to gather their baggage and get out of the airport. The access between these two areas is highly controlled by using the different manner of security monitoring.

### 1.2.2 Terminal description and configurations

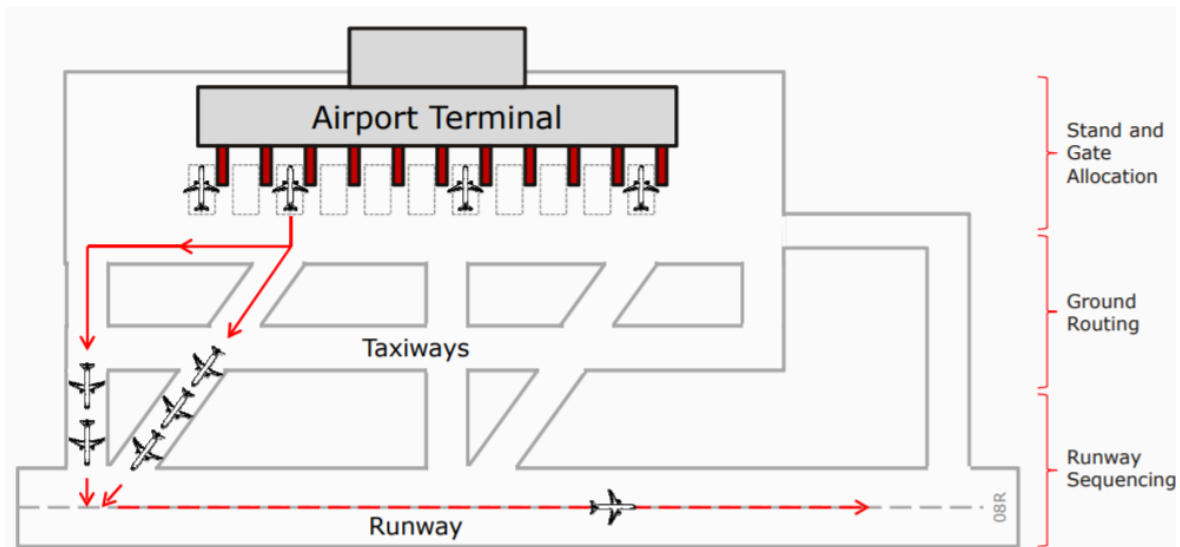
As stated by [183], the terminal's primary function is to offer a convenient service for the mode transfer, frontier activities along with the operations needed by the airlines. Its design supplies the terminal link with the land-side surface transport system,

commonly making the passengers pass over all the selling stores; apparent information throughout the series of processes; authorities control and security monitors; baggage managing system for both local and transfer bags.

In the airside component of an airport, three different airport ground resources optimization airport controllers have to manage, as depicted in Figure 34:

- Stand and Gate Allocation also called gate assignment
- Ground Routing
- Runway Sequencing

Our key area of focus in this chapter will be the first problem related to gate allocation.



*Figure 34 Airside ground resources optimization at an airport*

As outlined by [184], the configuration of the terminal building primarily is determined by the traffic they operate, getting this traffic separated into three parts: Overall volume of traffic, seasonality of traffic and the ratio of traffic that transfers between aircraft. According to the configuration, several elements of traffic will have advantages above the other. There are four basic configurations (see Figure 35):

- Linear: Individual stands are placed all along the terminal building, offering simplified access from the terminal building to the airplanes. It is used in airports with low traffic amounts; the planes are parked obliquely in order that they can easily enter and exit on their own.
- Midfield, both linear or X-shape: the stands are located aside from the terminal. The transport of the passengers to the terminal is supplied by using buses or mobile lounges.
- Satellite: every satellite construction is connected to the central terminal by using corridors or underground passageways.
- Finger piers: gate concourses are supplied to the terminal structure.

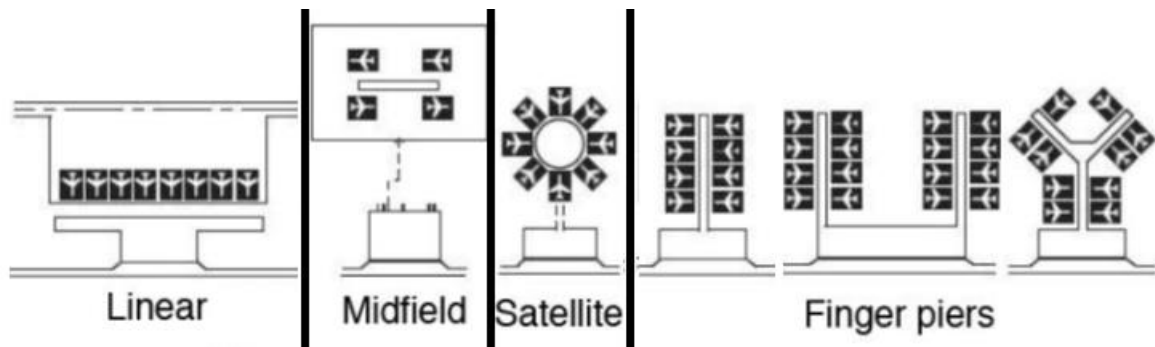


Figure 35 Terminal configurations [184]

In all those configurations, enough space must be supplied to prevent conflicts and match the preferences of large aircraft.

Throughout the years, the terminal building continues to be in continuous development and progress matching its demands. Three main considerations were in the root of these developments: improving aircraft technology, an enormous increase in passenger traffic and the attempt to boost the quality of service. As was set by the problematic sub-section 1.1, currently, airports have to maintain a good service quality and passenger satisfaction; furthermore, they grow into multi-functional facilities meant to furnish a large variety of services, more than their original features [185].

Multiple operations can be determined in the airport terminal, including Passengers Arrival, Boarding, Passengers Arrival, Baggage Handling, etc. Due to the passenger's progress through the terminal, they are facing those processes they have to pass to attain the airplane. Every single process outlined in the airport terminal uses several resources and has got its own constraints or restrictions. In this chapter, we consider the Gate assignment process, identified as the last phase of the passengers' experience through a terminal. This is the last step between the airport and the airplane, where passengers expect the opening of their particular gate so they can board the plane using the passport and boarding pass [186]. An example of airport Mohammed V configuration is addressed in the next sub-section, before the literature section.

### 1.2.3 Example of Mohammed V airport of Casablanca

In this sub-section, we describe the organization and the different elements of the infrastructure of Mohammed V international airport, as well as the variety of the configuration of this airport and its components as shown in Figure 36.

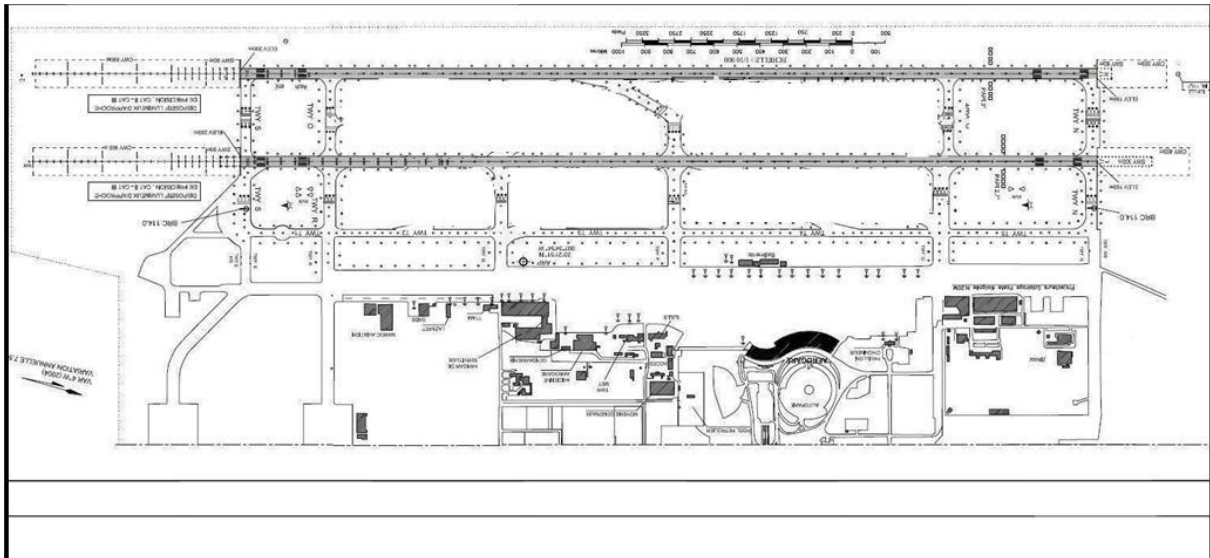


Figure 36 Mohammed V Airport Infrastructure [ONDA]

Mohammed V Airport consists of the following elements:

- Two terminals for passengers
- A freight terminal
- A control tower
- Two tracks
- 63 parking posts
- 9 Boarding Gates

The "Airside" side of the airport can be described by a loop whose node is formed by the runways and which passes through the parking stations of the aircraft. This schematization is used to illustrate the cyclical treatment performed on aircraft flows on the platform.

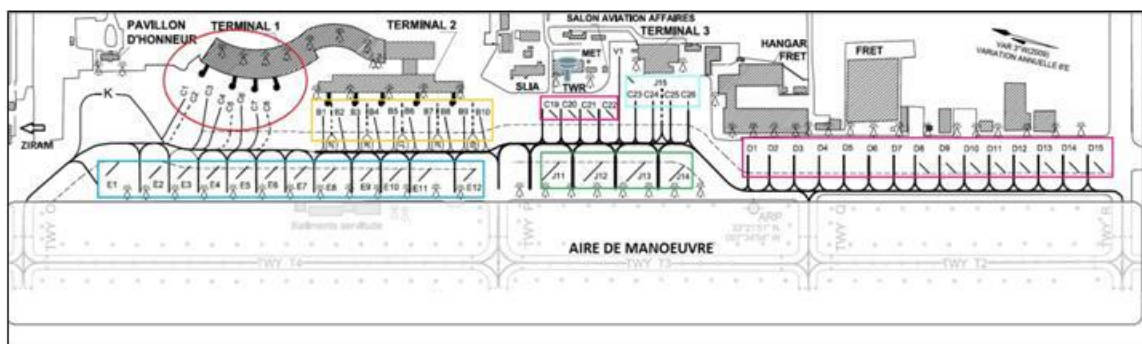


Figure 37 Mohammed V configuration

The configuration of gate areas is closely linked to passenger and cargo terminals. In Figure 37 of Mohamed V airport, we see that there are several parking systems, which follow the several configurations as Linear, Finger piers and Midfield. It can be considered as a hybrid configuration airport.

The next section will describe the literature review of gate assignment problem.

## 2. LITERATURE REVIEW

Gate assignment problem focuses mainly on assigning arriving aircrafts serving flights to available gates, or aircraft stands at the airport while satisfying some constraints and meeting some objectives.

More interest in recent years is allowed to providing advanced techniques in the air traffic framework. This is resulting from the increase in air transport traffic [187]. The main objectives are the best allocation and management of airport and airline resources in the best way effectively and efficiently. Caused by the dynamic stochastic operational environment conditions of air transport, the scheduling problems currently confronted by the airport and airline managers are leading to challenging and complex planning problems that involve innovative models and solutions. This is triggered by the significant diversity of resource segments that have to be regarded including terminals, flights, crews, baggage etc, and most are interdependent. In fact, stochastic disruptions in air traffic transport raised the complexity of the resolution models. This is progressively taken into consideration in most recent studies.

The main target of an airport is to guarantee a fluent flights traffic. Optimal assignment of aircraft must guarantee to make available over time the proper gates. If an aircraft is not assigned, it will be forced to wait on the ramp very well as in the air. This type of scenarios are quite undesirable on account of time wasting and let to flight delays. Also, ramps and airspace are resources with a limited capacity.

Gate flight assignment is an essential task of an airport; it is the primary activity in airline traffic transport management [188]. Moreover, several airports today have severe capacity constraints resulting from the increase in air traffic volume. The GAP can be regarded as such a problem of constraint resource assignment, in which gates represent resources and aircraft are considered as resource consumers.

As well, GAP is considered to be a challenging and tough problem [189] as it involves highly inter-dependent resources including gates, crews, and aircraft. Consequently, serious disruptions in the airport exposed as flight delays result from inadequate assignment that lowers the client services and generates conflicting flights and ineffective usage of gate services.

In this section, we will give the possible constraints and objectives of the GAP formulation, in addition to a survey of different used models, and resolutions methods from the literature.

### 2.1 Constraints and Objectives

All of the search methods perform on a search space. In order to produce the search space, a model of the real problem is generally created. The model reflects the main features of the real problem that may possibly impact the solution. The majority of the real world problems are to some extent uncertain. There are however several modeling techniques that provide the way to consider the uncertainty in a model in order to attain more accurate solutions.

The volume of facets of a modeled problem, identified as certain or uncertain, means the size of the search space. Considerably more complex models are the origin of much larger and most likely complicated search spaces. Therefore, building a suitable model with ample details but not so much complicated, is amongst the most challenging tasks whenever solving an optimization problem.

Mathematical programming [190] is the most commonly encountered modeling approach intended for the gate assignment problem. Ahead of providing an overview of applied GAP models, the next constraints and objectives are generally mentioned in the relevant literature:

### 2.1.1 Constraints

These constraints are principally classified as “soft” and “strict” constraints in the literature (Chun-Hung et al. [189]).

The strict (hard) constraints are inherent to the problem and can be described as follows:

- Single: Each flight must be assigned to one and only one gate. It means that No two flights can be assigned to the same gate at the same time
- Feasible: No two flights with overlapping ground times are assigned to the same gate concurrently, which means that a flight should be assigned to at most one gate.

Several soft constraints are also considered in the literature for example; those additional constraints could be associated with the model. The decision regarding adding soft constraints relies mainly on the specific airport priorities and the features of the problem the modeler aims for. The following list comprises some soft constraints outlined in the literature:

- Constraints related to various aspects of gates: just like sizes [191, 192]. This is a severe constraint which often happens in real life. It can enforce the assignment of specific airline flights to some predefined gates; also, the assignment of a large aircraft to a particular gate may imply that neighboring gates can only accept aircraft of a certain size or are even completely blocked (see Dorndorf et al. [192]. It is covered by the elaborate model included in this research (see next section).
- The preceding constraint: it implies that a flight may carry no more than one upcoming flight at the same gate [193]. It makes much simple the model formulation.
- The time gap constraint: it features a variable for every single pair of flights, which are successive on the same gate. It is also included in the objective function to maximize the time gaps amongst allocations. Maximization of the time gaps in assignment is a fundamental part of the problem as it enables minor delays on gates to be absorbed. Further robust solutions are supplied when the constraint is included in the model as in [191].

- Fixed minimum buffer time between two allocations as in [194]. It needs to be put into the model as it grants to the departing aircraft time to empty the gate. This constraint is generally included in real-world planning.
- Push back constraints: it focuses on assignments which may lead to pushing back conflicts in the locations around the gates as provided by [195].
- The shadowing constraint: it has to do with gates which are unable to be employed concurrently [196, 191], such as entry to one gate may possibly block the use of another. The shadow constraints often arise in real-world situations.
- The towing constraint: it addresses flights which are staying too long at an airport and are towed aside from the stands. It is generally modeled by dividing the flights to two or three items: arrival, departure and parking activity (or simply arrival and departure). It can be found in [195, 191].
- The passenger walking distance constraint: This constraint had to be highly common within preliminary stage researchers as [188] when the walking distance appeared to be a critical facet of the problem.
- The time window constraint: it specifies the time duration where an aircraft might stay at an airport [193]. If the time window is much longer when compared to the time an aircraft actually remains at an airport, it provides a possible assignment delay.

### 2.1.2 Objectives

Multiple objectives are usually included in the GAP models, and the objective function is commonly a weighted sum of them. Weights are established by the modelers to obtain a solution that indicate the main user concerns. Some papers even compare the results supplied for numerous weights in the objective function [197] or analyze the Pareto from a multi-objective formulation [198]. Also, several multi-objectives formulations with multi-criterion models are considered in [197, 192].

All these objectives can be divided into two big classes: passenger-oriented and airport-oriented objectives. For example, Teodorovic et al. [199] focus on total passenger delay and the number of flights cancellations in the case of irregularity of flights. In turn, Chang [200] considers the distance passengers have to carry their baggage as an objective in addition to passenger walking distance. In contrast to previous ones, airport oriented objectives like total gate preferences, number of aircraft towing procedures and others can be addressed. A list of typical objectives found in the literature are presented below:

- Minimize the total walking distance for passengers (Xu and Bailey [201]),
- Minimize the total waiting time of all passengers (Yan and Huo [197]),
- Minimize the number of un-gated aircraft activities (Lim and Wang [188]),



- Maximize the preferences of assigning certain aircraft to particular gates [198],
- Minimize the current schedule deviation from a reference schedule,
- Minimize gate conflict (Lim and Wang [188]).
- Minimization of the overall change from the originally planned schedule as in [202, 198]
- Minimization of the time an aircraft must wait for a gate. This constraint relates to the scenario when there are not enough gates at an airport, or they are wrongly allocated, and consequently, several arriving aircraft may be delayed as they need to wait for gates to be available [193].
- Minimization of the number of ungated flights such as [198, 191].
- Minimization of the number of conflicts between flights inside spaces around gates such in [203].
- Minimization of baggage handling distance [204]. It is carried out simultaneously with the passenger walking distance minimization.
- Minimization of the total passenger waiting time [194]. Where the total time that passengers pass at an airport is minimized
- Minimization of the number of towing operations [192, 202]. Every towing operation is a supplemental movement at an airport which can induce conflicts. Therefore, it is preferred to reduce the number of towing operations, specifically at the time of rush hours.
- Maximization of gates occupation time regarded as [205]. It leads to getting as many flights as possible on every gate.

## 2.2 Models of the GAP

A gate has been firstly identified by Hamzawi [206] as the area of the terminal apron designated for the parking of aircraft in order to load and unload passengers and to accomplish an aircraft ground services including refueling, baggage handling, cleaning, servicing, etc. Figure 31 displays a situation scenario from an airport. Two aircraft are observable in Figure 32; an aircraft has just arrived and is taxiing inside the direction of the assigned gate. Two other aircraft, apparent in the picture, have already been on its associated gate. The gate assignment problem is often recognized as a process of getting suitable places to park aircraft at an airport. This is certainly an optimization problem and to be able to resolve it a good model is required initially.

Formulation of the GAP can be carried out under two primary categories: deterministic models and stochastic models. In the first one, only static parameters are considered (just like flights arrival/departure, passengers, etc); this method becomes infeasible regarding stochastic variations, for example, weather conditions or flight delays. Stochastic GAP models have actually been examined to consider those variations into the model.

In this sub-section, several mathematical models found in the literature are discussed, especially stochastic models. First, the basic deterministic

mathematical model is presented before providing stochastic models from the literature, as the type of model tackled in the proposed approach.

### 2.2.1 Basic Mathematical Programming model

The mathematical programming model [207] requires a set of mathematical relationships that matches existing relations in a real problem. Many kinds of mathematical programming models are identified in the literature. They can be either linear programming models or otherwise non-linear programming models. The constraint and objective function are controlled by variables.

The traditional methodology for formulation in the literature of this problem is right from the passenger's perspective in the manner that the total passenger-walking distance is minimized as given by [201]. The objective function is in fact to minimize the total passenger walking distance. Three kinds of passengers are recognized: arriving passengers, departing passengers, and transfer passengers; with three walking distances to evaluate: the length between a gate and the arrival hall, the length between a gate and the departure hall, and the distance that separates two gates.

The classical formulation of the GAP as a mathematical programming model is given as follows. We define:

$G$  the set of gates

$F$  the set of aircrafts

$Z$  Objective function value

$x_{ik}$  decision variable which is equal to 1 when aircraft  $i$  is assigned to gate  $k$ , and equal to 0 otherwise

$a_i$  The arrival time of aircraft  $i$

$d_i$  The departure time of aircraft  $i$

$p_{ij}$  The total number of transfer passengers from aircraft  $i$  to aircraft  $j$

$p_{i0}$  The total number of arriving passengers of aircraft  $i$

$p_{0i}$  The total number of departing passengers of aircraft  $i$

$w_{kl}$  The walking distance between gate  $k$  and gate  $l$

$w_{k0}$  The walking distance between gate  $k$  and the arrival hall

$w_{0k}$  The walking distance between the departure hall and gate  $k$

Given the notation defined above, the formulated GAP model that we will be based on for our study as follows:

$$\text{Minimize } Z = \sum_{i \in F} \sum_{j \in F} \sum_{k \in G} \sum_{l \in G} p_{ij} w_{kl} x_{ik} x_{jl} + \sum_{i \in F} \sum_{k \in G} (p_{i0} w_{k0} + p_{0i} w_{0k}) x_{ik} \quad (50)$$

Subject to:

$$\sum_{k \in G} x_{ik} = 1, \quad \forall i \in F \quad (51)$$

$$x_{ik}x_{jk}(d_j - a_i)(d_i - a_j) \leq 0, \forall i, j \in F, \forall k \in G \quad (52)$$

$$x_{ik} \in \{0,1\}, \quad \forall i \in F, \forall k \in G \quad (53)$$

The Objective function (50) minimizes the total passenger walking distance. The quadratic component of the function is the total walking distance for the transfer passengers while the linear component is for the arriving and departing passengers. The constraint (51) must have every single aircraft to be assigned to only one gate. The constraint (52) is a non-linear constraint. It restricts that no two aircraft are assigned to the same gate at the same time. The constraint (53) limits all the decision variables to be 0 or 1.

This problem can be solved by taboo search algorithm, to provide an approximated solution. Various other researches attempted to enhance the offered heuristic solution by strategies as stochastic neighborhood search (Hakki et al. [205]). Multiple constraints can be integrated into the formulation. This category of models known as static GAP model and are unable to support any change in its parameters. In contrast, the GAP is always susceptible to uncertainty and could change eventually. Therefore, static models would not respond to support important variations in the flight schedule and continue to be infeasible in such a case. The next subsection will treat the stochastic models of the GAP.

## 2.2.2 Uncertainty managing Models of the GAP

However, if the assignment is robust enough, it will help the operators react to some uncertain events. Therefore, stochastic and robust GAP models have been studied extensively as well. For example, Hassounah and Steuart [208] show that planned buffer times could improve schedule punctuality. Yan and Chang [209] and Yan and Huo [197] use in their static gate assignment problems a fixed buffer time between two continuous flights assigned to the same gate in order to absorb the stochastic flight delays. Yan and Chang [209] also develop a multi-commodity network flow model as well as Binod et al. [210], who gives additional objectives as minimizing the fuel burn cost of aircraft taxi by type and expected passenger discomfort.

Shangyao et al. [211] develop a heuristic approach sensitive to stochastic flight delays embedded in a framework that includes three components, a stochastic gate assignment model, a real-time assignment rule, and two penalty adjustment methods.

In [192] mathematical models and (optimal and heuristic) procedures are proposed to provide solutions with minimum dispersion of idle time periods for the GAP. More recently, Merve et al. [212] give stochastic

programming models incorporating robustness measures based on the number of conflicting flights, idle and buffer times, formulated as large-scale mixed-integer programming problems and resolved by tabu search.

To handle uncertainty on aircraft schedule, Lim and Wang [188] modeled GAP as stochastic programming model and transformed it into a binary programming model that suffers from NP-hardness. They proposed a hybrid meta-heuristic combining a tabu search and a local search for resolution. They introduce unsupervised estimation functions without knowing any information on the real-time arrival and departure time of aircraft in advance. After that, we will use those functions as an input parameter for our own model.

More alternative models for the GAP are presented and classified as below.

*a. Simulation Approach to GAP*

Among the first papers on the GAP, [206] elaborated an interactive simulation method which in turn simulates arrivals and departures and affects gates. In fact this method did not involve a mathematical programming formulation of the problem; it outlined numerous necessary constraints which were after formally detailed by other research workers. It particularly attached aircraft-gate size constraint, airline priorities, maximum gate occupation time, minimization of walking distance and passenger delays, buffer times between flights allocated to the same gate, and maximum delay which can be brought about a flight on the gate.

*b. Fuzzy Logic Application*

The Fuzzy logic methodology [213] is relying on a level of truth instead of on the traditional true or false logic, also used in stochastic modeling approaches. The levels of truth are given to uncertain data as a way to attain the most appropriate reasoning.

Fuzzy logic was used by [214] to build a model integrating the uncertainty of the GAP. The model considers that the idle times between affectations is fuzzy. It is among the objectives that was indeed to maximize this idle time which is weighted in the objective function implementing membership levels determined by an adjustment function. The relationship between the design of the adjustment function and the given results were considered, and it was advised that the shape should be selected regarding the priorities of a specified airport. The genetic algorithm (GA) was implemented to resolve the fuzzy model of the GAP. Bringing out both the fuzzy logic and applying the GA was impressive and offered appealing results for the basic GAP model. A similar method using fuzzy logic and bee colony optimization was found in [215].

*c. Expert Systems for the GAP*

Expert system [216] is a software system that imitates a decision process as an expert. Expert systems include rather different operating concepts when compared to other mentioned approaches. This approach does not make use of mathematical models, or any objective function or constraints. In fact standard expert systems implement a knowledge base in the reasoning procedure. The knowledge base is a pair of if/then rules, structured regarding a human expert experience.

Expert systems were utilized to solve the GAP as found in [217]. Also, [218] incorporates a hybridization between a linear programming model and an expert system. The linear programming model covered just the main constraints of the GAP. The standard solution attained applying the basic LP model was altered applying the rules from the expert system.

*d. The Gate Re-assignment Problem*

The study provided in this chapter centers on the planning of the gate assignment taking in consideration stochastic disturbances. However, the plan which is produced earlier generally needs to be re-planned at the time of the daily operations caused by unexpected cases, such as delays, bad weather conditions, some emergencies. The problem which takes up on-line changes in the planning at the time of flight operations is known as the re-assignment problem, while other approaches are considered robust when they handle eventual changes. Nevertheless, the re-assignment problem is, in general, an online model of uncertainties. The re-assignment problem is commonly resolved online, and, consequently, will involve models and techniques which are quick to execute.

In [219] a genetic algorithm was offered to solve the re-assignment problem, where one individual covered one likely method of assigning the flights which must be re-assigned. The objective was in fact to re-assign the flights impacted by a delay in order that an extra delay is minimized.

Paper [204] gives an extension of that offered in [219], where a genetic algorithm with a uniform crossover which rarely leads to infeasible solutions was provided.

In [220] a mathematical integer programming model of the re-assignment problem was provided. The objective of the model is to decrease the total deviation from the firstly planned schedule. An exact solution is given on-line every time there was a need for re-assigning a flight influenced by a delay. Positive computation times and good improvement when compared to re-assignment worked by hand had been reported.

A related approach was provided in [221], where also a mathematical integer programming model using as objective the minimization of the

walking distance of transferring passengers, once re-assignment procedure is performed. The model was resolved exactly just for the flights affected by manifesting delays. Several additional mathematical models of reassignment were reported in [222].

### 2.2.3 Datasets

The provided literature uses numerous sorts of datasets. Some authors arbitrarily generated the input datasets [193] even while others produced them employing real datasets like [197]. A number of the authors regarded just a part of a real dataset (as they deal only with delayed flights as in [221] or with flights and gates at one particular airport [191]) once they assumed it is acceptable to implement it. Smaller sized datasets were commonly much easier to solve, employed to validate the effectiveness of new methods and models as in the proposed approach.

## 3. STOCHASTIC PROPOSED APPROACH OF THE GAP

Various GAP models and techniques are identified in the literature. Static as well as stochastic models are developed. Working with methods with an exact solution can be more suitable. However, [223] states that these kinds of exact methods are actually ineffective to resolve real problems. This is because flights in static models are allocated to gates depending on the expected flight schedule using fixed parameters. Nonetheless, in real operations, stochastic disruptions occur frequently, leading to real-time adjustments of gate assignments and flight delays. Consequently, stochastic methods have been widely motivated in recent researches.

Consequently, to build a significantly better gate flight assignment approach, it has to include in the model the possibilities of stochastic flight delays that may arise in real operations.

When it comes to stochastic environments, Markov Decision Processes (MDPs) [142] have confirmed to be effective in optimal decision making. Other derived version of MDPs like the multi-agent Markov decision process [154] and the time-dependent Markov Decision Process are also developed to manage some challenges in the standard GAP based MDP introduced. Three MDP based models are presented in this section.

### 3.1 MDP MODEL

Markov decision processes (MDPs) are a discrete time stochastic control process that can be used to model a sequential decision making with uncertainty. MDP Takes into account reward of the current decision and future opportunities. In this approach, an agent also considered as decision maker is interacting with the world (the environment decision). At each

time period  $t$  of the planning horizon, the system state  $s$  provides the agent with all the information necessary for choosing an action  $a$ . As a result of executing action  $a$  with system state  $s$ , the agent receives a reward  $r$  and the system change to the next state  $s'$  with a definite probability.

We use an MDP to represent the problem of gate assignment. Our formulation will be more an airport-oriented approach. However, this does not prevent having the possibility of a passenger-oriented approach with MDP formulation. We made a choice to focalize on the airport-oriented point of view in this thesis. We define some preliminary requirements before giving the MDP model of the GAP.

### 3.1.1 Aircraft size constraints model



Figure 38 Example of gate disposition

An airport disposition, as in Figure 38, can be represented by a graphical schema. Figure 39 gives an example of gates representation in an airport; it can be represented as a constrained graph having as constraints distances between gates, which will represent a constraint on aircrafts size to assign to each gate. (see Figure 39)

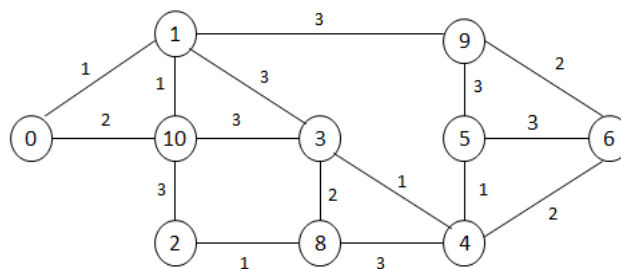


Figure 39 Graph representation of gates

Let  $G = (V, E)$  be an undirected graph, where  $V = \{v_1, v_2, \dots, v_n\}$  the set of vertices  $E = \{(v_i, v_j)/v_i, v_j \in V\}$  is the set of edges.

$M = (m_{i,j})^{n \times n}$ : Symmetric compatibility matrix defined as:

- $n$  represents the number of vertices of the graph.
- $m_{i,j}$  with  $i \neq j$  is a quantitative constraint representing the minimum required distance separation between two vertices  $v_i$  and  $v_j$ .
- $m_{i,j} = 0$  Means that there is no constraint between vertices  $v_i$  and  $v_j$ .

Let:  $f : V \rightarrow C$  a function that associates to each one of vertices a certain value  $c$  of  $C$ .  
With  $C = \{1, \dots, k\}$ .

So, the constraints to consider on this graph between every  $v_i, v_j \in V$  are:

$$\frac{1}{2}m_{i,j} (f(v_i) + f(v_j)) \leq m_{i,j}^2 \quad (54)$$

### 3.1.2 Airport Configuration

In the GAP we have to optimally assign flights to gates with respect of “strict” and some “soft” constraints that we mentioned in the literature section. We try here to respect these constraints under stochastic conditions that occur with flights disturbance and causes what we call flight conflicts defined as when two flights with overlapping ground times are assigned to the same gate.

We define:

Set of Gates:  $A = \{a_1, a_2, \dots, a_n\}$ , Where  $n$  is the number of gates

Set of aircraft  $S = \{s_1, s_2, \dots, s_m\}$ , Where  $m$  is the number of aircraft.

For each aircraft  $s_i$  ( $1 \leq i \leq m$ ):

- $a_i$ : scheduled arriving time;
- $d_i$ : scheduled departure time;
- $a'_i$ : real arriving time;
- $d'_i$ : real departure time;

$b$ : buffer time between two consecutive aircraft assigned to the same gate. In other words, the gate is locked for serving aircraft  $s_i$  in  $[a'_i - b, d'_i + b]$ ;

Definition: Gate Conflict [188] :

Two aircrafts  $s_i$  and  $s_j$  have gate conflict if both the following two conditions hold:

- Aircraft  $s_i$  and  $s_j$  are assigned to the same gate;
- There is an overlap between the two-time durations during which the aircraft will lock the gate,  $([a'_i - b, d'_i + b] \cap [a'_j - b, d'_j + b]) \neq \emptyset$



Our approach does not take into consideration as variables the arriving and departure time; those variables are only used to calculate the conflict probability introduced later.

### 3.1.3 MDP parameters

We consider the flight arrivals as a stochastic process for the MDP. It takes its values from the state space. Then, as outlined by the previous chapter, MDP models will require to define the following four elements:

- A finite set of states S: This is given by a set of aircrafts S.
- A finite set of actions A: Each action corresponds to assign a gate  $a$  to a flight  $s$ . If there is no best flight, the action consists of keeping the gate assigned to the flight.
- Transition probability: it reflects the stochastic aspect of our problem. We define transition probability as follow :

$$T_{s,s'}(a) = \psi_{s,s'} \quad (55)$$

With  $\psi_{s,s'}$  is the probability of having the two flights  $s$  and  $s'$  in conflict if they are assigned to the same gate. This probability includes the possibilities of disturbances that make  $s$  and  $s'$  in conflict.

Lim and Wang [188] introduced an estimation function to estimate the expected value of the probability of the gate conflict between flight  $i$  and  $j$ . they calculate  $(p(i,j))$  : the expected probability of gate conflict for flights which are assigned to the same gate.

We use the algorithm 24 to compute  $\psi_{s,s'}$  based on  $E(p(i,j))$ .

- Reward function: is defined as follow :

$$R(s, a) = - \sum_{s,s'} \rho_{s,s'} \sigma_{s,s'} + Pref(s, a) \quad (56)$$

$\rho_{s,s'}$ : a weight associated with the constraint  $m_{i,j}$  cited earlier.

$d_{s,s'} = \frac{1}{2}(f(s) + f(s'))$  : Minimum distance separation between flights  $s$  and  $s'$  in the ground, and  $f$  a function that associate to each flight a scale of size for the used aircraft; for example,  $f$  has a value from one to 5 according to the size of the aircraft.

Thus:

$$\begin{cases} \sigma_{s,s'} = 0 & \text{if } m_{a,\pi(s')} \cdot d_{s,s'} \geq (m_{a,\pi(s)})^2 \\ \sigma_{s,s'} = (m_{a,\pi(s')} - d_{s,s'})^{-1} & \text{else} \end{cases} \quad (57)$$

$Pref(s, a)$ : Preference value for choosing the action  $a$  for the flight  $s$ .

Hence, the reward function gives a penalty for the flight that does not respect the aircraft size constraints and a bonus to flights corresponding to air company preferences.

Let  $D = (d_{s,s'})^{n \times n}$  a matrix where  $d_{s,s'}$  minimum distance separation between flights  $s$  and  $s'$ .

We suppose as an initial solution of our algorithm the solution of the determinist problem without delay consideration (initial schedule). Delay probabilities can be calculated from the history of past flights or using some other estimation techniques that will be discussed in the next paragraph.

**Algorithm 24 :** Algorithm for computing probabilities

```

Let the probability  $\psi_{s,s'}$ .
For  $s:=1$  to  $n$  do
  sum:=0
  For  $s':=1$  to  $n$  do
    sum:=sum+ $E(p(s,s'))$ 
    For  $s' := 1$  to  $n$  do
       $\psi_{s,s'} := E(p(s,s'))/\text{sum}$ 
  End
End

```

The transition probability does not depend on  $a$ , so the solution  $V_\pi$  can be simplified as follow:

$$V_\pi = \frac{R_\pi}{1-\gamma P} [224] \quad (58)$$

To find a solution to our problem, we have to use the policy iteration algorithm. We implement this algorithm using Matlab in order to evaluate this approach of resolution.

### 3.2 MULTI-AGENT MDP FORMALISM

Multi-agent systems (SMA) are a branch of Distributed Artificial Intelligence. Their applications are wide: humanities, game theory, economics, and real-world applications such as air traffic control, networking, and robotics. SMA approaches are interested in interactions between autonomous entities. This situation is mostly studied in SMA as the cooperation that involves complex mechanisms.

In SMA, we can have agents that collaborate to achieve their assigned goals; so, interaction with other agents is crucial. Reinforcement learning is a

promising technique for creating agents that co-exist (Yanco et al. [225]), but the mathematical framework that justifies it is inappropriate for multi-agent environments. The theory of Markov Decision Processes (MDP's) (Howard [142]), which underlies much of the recent work on reinforcement learning, assumes that the agent's environment is stationary and as such contains no other adaptive agents. The theory of games (von Neumann et al. [226]) is explicitly designed for reasoning about multi-agent systems. Markov games (see e.g., Van Der Wal [227]) are an extension of the game theory to MDP-like environments. Multi-Agent Markov Decision Processes (MMDP) allows us to widen the MDP view to include multiple adaptive agents with interacting or competing goals (Michael L. Littman [151]). Many applications have been done with MMDP, even in the field of air transportation see [152].

### 3.2.1 Multi-agent control mechanism

MMDP problem is formulated as a set of autonomous learners interacting agents. These agents must learn to coordinate and to cooperate to achieve their goal. Thus, this approach combines two areas: distributed to the aspects of the interaction between agents and techniques of machine learning (Machine Learning) associated with a point of view for decentralized decision-making aspects (Peter Stone et al. [153]).

So, we use the formalism of Markov decision processes in the multi-agent framework (Michael L. Littman [151]). We assume having a centralized controller who has all information about the system (Figure 40), such as the global state of the system, actions, and rewards, and even distributes individual commands; therefore, the controller has the decision power and maintains the information shared between agents.

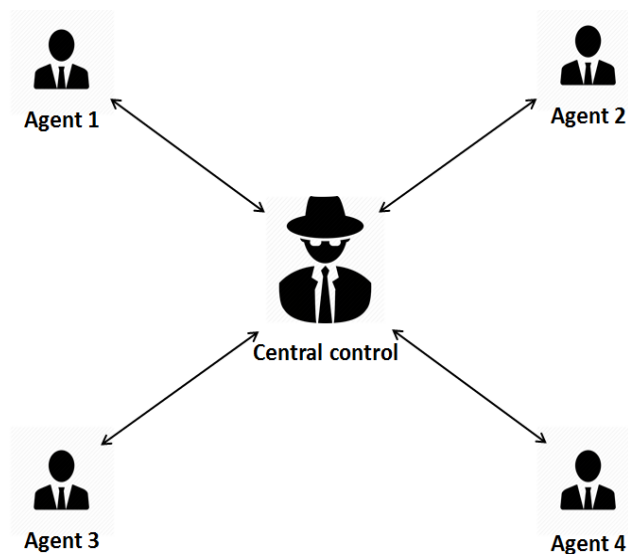


Figure 40 Centralized control in MMDP

We add real-time values to agent concept, to include time evolution into the multi-agent system dynamics. A Time-dependent Markov Decision Process (TMDP) is proposed by [150] to provide this extension. This model consists of stochastic state transitions and stochastic time-dependent action durations. The actions in the TMDP model are stochastic and time-varying:

$$a(t) \sim \text{policy}(s, a(t)) \quad (59)$$

Resulting policies are actions to be performed by agents in every time sequence. Then, we can enlarge the real planning window to problems under uncertainty changing with time.

So, we consider in our formulation an assignment-based decomposition approach that is intermediate between the joint MDP approach and the independent agent approach. We assume a centralized controller that has relevant information about the states of all agents to assign tasks and allocates tasks and resources to agents based on task-level value functions of agents. Once the tasks are assigned to agents, the lower-level actions of agents are decided by the task-level value functions until the tasks are reassigned by the central controller. We call such domains with multiple tasks and multiple agents Multi-agent Assignment MDPs (MMDPs). Then, adding time dependence behavior will give a more real representation of our problem. Also, we are inspired by TMDP coupled with the MMDP approach, to provide a new formalism of time-dependent Multi agent MDP. This approach will help us to have real-time policies to apply in every case of disturbance for the GAP problem.

### 3.2.2 Multi-agent MDP model

There are many contributions in the literature that are trying to deal with uncertainty (see literature section). With the same target of building a robust mechanism that can absorb flights disturbance, we develop this approach based multi-Agent Markov Decision Process. Our choice for this technical background to model the problem can be considered as the preferred alternative due to its capabilities of stochastic decision optimization on a discrete time Markov chain. Intelligent agents are gaining wide acceptance as a useful and a powerful tool for solving complex problems and seems to be a promising alternative. Also, the advantages of multi-agent reasoning include distribution of processing, support for the more flexible peer-to-peer model, decentralization of control, the reduction of network bandwidth use, etc.

Using this theory, the centralized controller (see Figure 41) will have in advance a solution composed of all decisions that can be made during the planning horizon of flights assignment or allocation to gates. So, with MMDP, no need for real-time optimization because we assume having predefined solutions for all possible cases of disturbances. Thus, for a given

gate allocation combination, the solution gives the best decision of gate assignment to make.

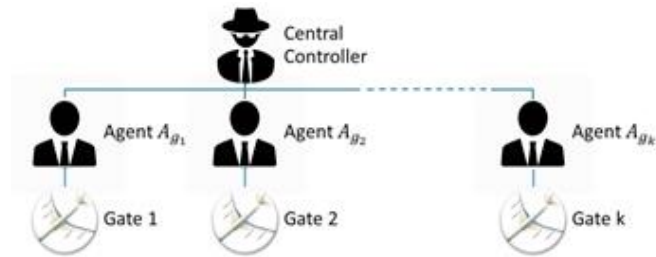


Figure 41 Agents representation

Before extending the model of GAP to be time-dependent, an earlier formulation like in [228] of the gate assignment problem with MMDP is presented. The model is given by a tuple  $\langle K, S, A, P, R \rangle$  as a follow (see Figure 41):

The State  $S = S_1 \times \dots \times S_K$  is a vector giving the diverse feasible combinations of flights indexed by its assignment position  $S_i = (s_1, \dots, s_k)$ , where  $k$  is number of gates and  $s_i \in V$ .  $V$  represents the set of flights to be allocated to gates during the planning horizon (one day in general).

The set of actions  $A = A_1 \times \dots \times A_K$  describes the set of joint actions for the agent,  $A_i$  gives the set of local actions of the agent  $i$ . For every single agent, performing a  $a \in A_i$ , will match an action of allocation a flight  $a \in V$  to the gate  $i$ .

Therefore, each agent is in charge of handling a particular gate, and a  $a \in A_i$  for agent  $i$  considers that there is a set of feasible flights to be affected to gate  $i$ .  $A_i \subset V$  that are appropriated to be allocated to gate  $i$ . This supposition is regarded as a feasibility constraint that describes the possible assignment.

Defining:  $A_{i,t}$  set of feasible flights for the gate  $i$  at a discrete time  $t$ . Then:

$$A_i = \sum_t A_{i,t} \quad , \quad (i \leq k) \quad (60)$$

$P(s, s', a)$  gives the probability of transition as :

$$P: S \times S \times A \rightarrow [0, 1] \quad (61)$$

It represents the probability of the going from state  $s$  into another state  $s'$  when agents perform a joint action  $a \in A$ . This probability is viewed as the possibility of modifying assignment combination from  $s$  to  $s'$  resulting from executing a re-assignment action.

The probability  $P$  is integrating the complete stochastic information about assignment of gates including stochastic delays as well as additional disturbances that impact gate assignment and computed as a probability of occurrence. This probability utilizes other estimation techniques to build the probabilistic model of GAP under possible disruptions.

The way how transition probabilities are defined is essential for building the robustness of the GAP based MMDP model. The state transition stochastic matrix  $P$  defines all likely possible state transition probabilities ( $p_{ij}$ ):

$$P = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ \dots & \dots & \dots & \dots \\ p_{n1} & p_{n2} & \dots & p_{nn} \end{bmatrix} \quad (62)$$

Where:

$$\sum_{j=1}^n p_{ij} = 1 \quad (i = 1, 2, \dots, n), p_{ij} \geq 0 \quad (i, j = 1, 2, \dots, n)$$

Various statistical estimation methods could be applied to calculate state transition probabilities described above. The method as in [229] is applied using statistical data of state transition. Actions corresponding to flights combination are identified, and the arising states are collected from data. The collected values from observed data are:  $k_1(a)$  that corresponds to the case without disruption on state  $s_1$  performing an action  $a$ , and  $k_{12}(a)$  that corresponds to the case of disruption observed between state  $s_1$  and state  $s_2$  performing an action  $a$ . Therefore the transition probability between  $s_1$  and  $s_2$  performing an action  $a$  is estimated from observed data as :

$$p(s_1, s_2, a) = k_{12}(a) / k_1(a) \quad (63)$$

$R(s, a, s')$  corresponds to the reward acquired once transiting from a state  $s$  to a state  $s'$  performing an action  $a$ . This involves costs as negative reward or positive reward as the benefits of each reassignment.

$R$  is thought as:  $R: S \times S \times A \rightarrow R$

Where its function is defined as:

$$R(s, a, s') = -\lambda \delta_{ss'} p(s, a, s') + \gamma (1 - \delta_{ss'}) p(s, a, s') \quad (64)$$

Where:

- $\delta_{ss'} = 1$  if  $s = s'$  and 0 otherwise
- $\lambda$  Penalty unit
- $\gamma$  Recompense unit

The main task of a decision maker is to compute a policy as:

$$\pi: S \rightarrow A \quad (65)$$

A state-action sequence of decisions that maximize the expected total reward is denoted as  $\pi^*$ , and corresponds to the policy optimal.  $V^*(s)$  gives the maximum cumulative reward attained by the optimal policy beginning with states. Therefore, the optimal decision in a state  $s$  is to choose an action  $a$  maximizing the sum of the immediate reward  $R(s, a, s')$  and the value  $V^*$  of the immediate successor state, discounted by  $\gamma$  ( where  $0 \leq \gamma < 1$ ) :

$$\pi^*(s) = \text{arg max}_a [R(s, a) + \gamma V^*(p(s, a))] \quad (66)$$

The solution concerns obtaining an optimal stationary policy  $\pi^*$  that maximize for each state  $s$  and for all agents the expected discounted future reward.  $\pi^*$  contains the optimal decisions to make in every gate considering the assignment state.

MMDP model representing the GAP problem is solved using the value iteration function determined by Howard algorithm (see [142]).

### 3.3 TIME-DEPENDENT MULTI-AGENT GAP FORMULATION

The real interest is given to sequential decision problems. The theoretical aspect based on MDPs gives the best well-known tool to model and solve them, giving optimal results. However, real-world problems have additional and specific behavior, which is time dependence. MDP reflects only fixed time steps between decision epochs, which can be easily modeled as iteration steps. This property does not reflect the real evolution of problems like the subject of gate assignment. To bypass this limitation, Time-dependent MDP (TMDP) has been proposed in those models (see the previous section). The transition between states is not instantaneous but proceeds in a specific time  $t$ . Also in TMDP, the time is always observable; optimal policies give to the agent the best moment to make a decision or execute an action due to the state of the system.

Inspired by other occurrences like the truck dispatching system where decisions about truck assignments and destinations are made in real-time [230], choosing to benefit from temporal aspect and to project it to gate assignment problem. Therefore, the rewards associated with action outcomes in the time-dependent frameworks will be represented as time-dependent functions including more real evolution information of the problem.

Based on the same approach as the previous model, this sub-section presents another model with Multi-Agent reasoning but including the time evolution aspect of the gate assignment problem. The considered Time-dependent Multi-agent Markov Decision Problem is illustrated in Figure 42.

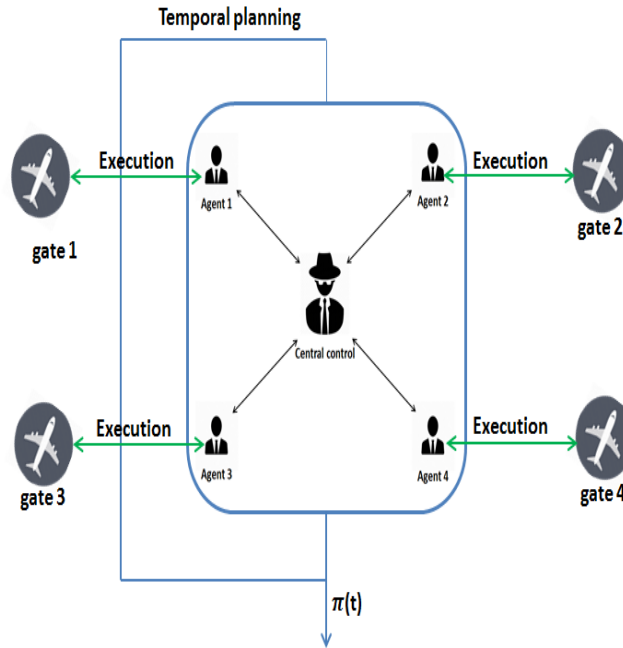


Figure 42 Agents distribution and temporal planning

Let  $K$  is the Number of agents; it also corresponds to the number of gates. Taking the same actions definition from the previous model, the set of actions  $A = A_1 \times \dots \times A_K$  defines the set of joint actions of agents, being also for every agent  $i$  assigning a flight  $a \in V$  to a gate  $I$ ,  $V$  is the set of flights.

Additional temporal information will be included first in the Discrete set  $M$  set of outcomes, of the form  $\mu = (s'_\mu, T_\mu, P_\mu)$  :

- $s'_\mu \in S$ : the resulting state space
- $S = S_1 \times \dots \times S_K$  gives different possible combinations of flights  $a \in V$ .
- $T_\mu \in \{ABS, REL\}$ : Type of the time distribution (absolute or relative).
  - If  $T_\mu = ABS, P_\mu(t')$  will be a pdf over absolute arrival times of  $\mu$  and corresponds to distribution time associated to some gates assignment configuration action.
  - If  $T_\mu = REL, P_\mu(\delta)$ : pdf will be over durations of  $\mu$  that corresponds to the duration needed to establish the assignment configuration action.
- $L$ :  $L(\mu|s, t, a)$  is the likelihood of outcome  $\mu$  given state  $s$  of gate assignment  $s$ , time  $t$  and action of next assignment to execute  $a = (a_1, \dots, a_n), a_i \in V$ .
- $R$ :  $R(\mu, t, \delta)$  is the reward for the outcome  $\mu$  at time  $t$  with duration  $\delta$ , corresponding to reward of spending  $\delta$  duration at time  $t$  with airport assignment action  $\mu$ . The reward includes as the previous model two components :
  - A benefit from the gate assignment outcome  $\mu$ .
  - A penalty to assignment outcomes  $\mu$  that causing a possible disturbance at time  $t$  and with duration  $\delta$ .

The purpose of defining TMMDP formalism of the GAP is to model and solve large real GAP planning under uncertainty taking into account either cooperative aspect of



the problem and property of time evolution. Resulting policies are actions to be performed by agents in every time sequence.

### 3.4 EXPERIMENT

Experiment on each of the defined approaches is addressed in this section. It presents the main results obtained after tests performed over several instances of gate assignment problem.

#### 3.4.1 Single Agent model experiment

We conduct a computational study to test the effectiveness of the implemented algorithm. We use real data from Hong Kong International Airport (given in [188]) to conduct our experimentation.

We consider for illustration 3 gates and 6 aircraft to assign in the time window from 11 am to 5 pm, as shown in figure 13.

*Table 13 Data from Hong Kong international airport*

Flight	Arrival	Departure	Route	Airline
CA101/102	11:25	12:45	Beijing-Hong Kong-Beijing	Air China
LH738/739	11:30	13:10	Frankfurt-Hong Kong-Frankfurt	Lufthansa
TG600/601	11:45	12:45	Bangkok-Hong Kong-Bangkok	Thai Airway
JL710/702	13:15	15:00	Osaka-Hong Kong-Osaka	Japan Airlines
BR869/870	14:25	15:30	Taipei-Hong Kong-Taipei	EVA Air
SQ862/861	14:20	16:00	Singapore-Hong Kong-Singapore	Singapore Airlines

Therefore, early arrivals or flight delays frequently arise in real-time operation. Depending to the schedules of two flights assigned to the same gate, although there is no overlap of time durations when they occupy the gate, there can be a possible conflict between the two flights as a result of an early arrival or a flight delay. As an example in the table above, the first flight has not been in conflict in the planning of the scheduled arrival and departure time, but the second one became delayed and developed into conflict with the first one. After JL710/702 flight arrives at Hong Kong International Airport, it will be obligated to wait on the ramp or merely in the air. Consequently, considering the real-time flight disturbance, we will need to look at the arrival time and departure time of a flight as random parameters instead of deterministic forms, in order to design a robust airport gate planning to secure the system from uncertainty in operation.

We give first the values of  $E(p(s, s'))$  calculated by Lim and Wang [188] as follows:

$$E(p(s, s')) = \frac{\max\{e(s, s')\} - e(s, s')}{\max\{e(s, s')\} - \min\{e(s, s')\}} \quad (67)$$

With:  $e(s, s')$  is an estimation function to estimate the expected value of the probability of the gate conflict between flight  $s$  and  $s'$  only based on the scheduled time gap between the two flights without considering the historical data. Table 14 gives  $E(p(s, s'))$  values.

*Table 14 Matrix of  $E(p(s, s'))$  for numerical example*

	CA101/102	LH738/739	TG600/601	JL710/702	BR868/870	SQ862/861
CA101/102	-	0.47	0.30	0.04	0.00	0.00
LH738/739	-	-	1.00	0.16	0.01	0.02
TG600/601	-	-	-	0.05	0.01	0.01
JL710/702	-	-	-	-	0.35	0.41
BR868/870	-	-	-	-	-	1.00
SQ862/861	-	-	-	-	-	-

Using the algorithm given in table 15 we can calculate a stochastic matrix associated with:

*Table 15 Stochastic Matrix of  $\psi_{s,s'}$*

	CA101/102	LH738/739	TG600/601	JL710/702	BR868/870	SQ862/861
CA101/102	0.00	0.58	0.37	0.05	0.00	0.00
LH738/739	0.00	0.00	0.84	0.13	0.01	0.02
TG600/601	0.00	0.00	0.00	0.71	0.14	0.14
JL710/702	0.00	0.00	0.00	0.00	0.46	0.44
BR868/870	0.00	0.00	0.00	0.00	0.00	1.00
SQ862/861	0.00	0.00	0.00	0.00	0.00	0.00

In addition to this method, we can also estimate conflict probability by either using simple inference estimation by interval, or as cited in the literature [231], we can estimate those probabilities by a Monte Carlo method.

As the initial policy, we adopt the solution given by the deterministic approach of the problem based on linear programming. We take random values of input parameters such as reward values. We try to evaluate by experimenting the feasibility of the proposed resolution algorithm.

The initial policy is not feasible because of delay related to Flight LH738/739 (figure 43), and we have a conflict in the assignment. So we take this solution as initial policy in policy iteration algorithm and we try the resolution.

Gate	11:00	12:00	13:00	14:00	15:00	16:00
gate 1		CA101/102			BR869/870	
gate 2		LH738/739	Conflict	JL710/702		
gate 3		TG600/601			SQ862/861	

Figure 43 Initial policy

The solution given by our approach (figure 44) can handle this type of disturbance and gives a robust solution to flight delays.

Gate	11:00	12:00	13:00	14:00	15:00	16:00
gate 1		CA101/102			BR869/870	
gate 2		LH738/739			SQ862/861	
gate 3		TG600/601		JL710/702		

Figure 44 given solution

To evaluate performances of this approach, we have simulated with one hundred flights and twenty gates, and the results were promising with a time execution not up to 30 seconds.

The promising benefits of applying the Bellman theory are to have a robust assignment policy that can be adopted in the case of stochastic flight delays, which causes conflict in the assignment. We experiment within this approach by a simulation of the related policy iteration algorithm. It gives a solution in acceptable resolution time.

### 3.4.2 Multi-Agent Model experiment

Computational analysis is done to test the efficiency of the used Multi-Agent MDP approach, and utilizing a simple data example to conduct experimentations.

For simplification, data includes two gates and three aircraft to allocate in a discrete window of time between  $T_0$  and  $T_3$ .

$V_i \in V$ , is set of flights and for  $i = 0$  it match a vacant assignment gate.

As a sample, in this experimental instance exist three possible states:

$s_1 = (V_1, V_2)$ ,  $s_2 = (V_3, V_0)$ ,  $s_3 = (V_0, V_3)$ . Two agents are affiliated to the two gates, therefore actions are:  $a_1 = (V_1, V_2)$ ,  $a_2 = (V_3, V_0)$ ,  $a_3 = (V_0, V_3)$ .

As an initial policy as in table 16, the solution provided first by a deterministic approach to the problem from literature is used. Simple values are used as input parameters only for simulation. The preliminary policy is :  $\pi_0 = (a_2, a_1, a_1)$ .

Table 16 Initial policy without disruption

		$T_0$	$T_1$	$T_2$
Agent 1	Gate 1	$V_1$		$V_3$
Agent 2	Gate 2	$V_2$		

It is designed regarding observations, transition probabilities, and rewards are shown in figure 45.

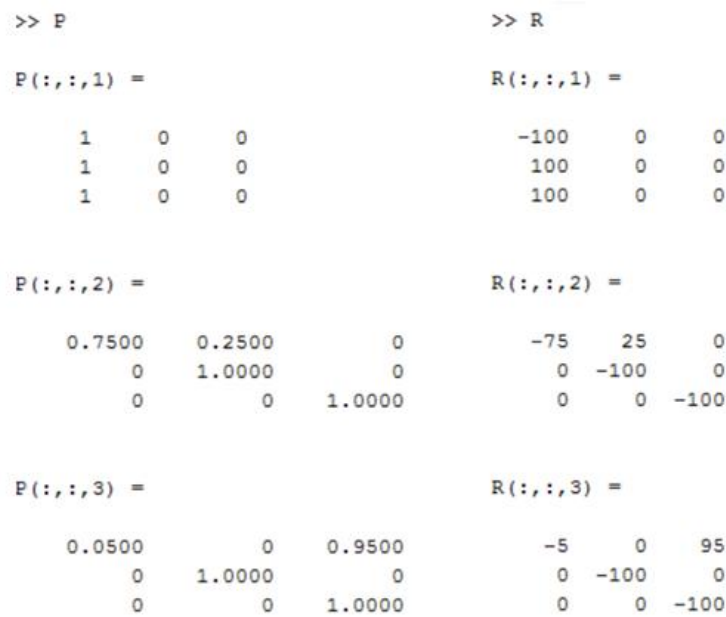


Figure 45 Transitions and rewards matrixes

With  $\lambda = \gamma = 1$ .

$p(s_1, s_1, a_2) = 75\%$  expresses a probability of disruption performing action  $a_2$  on  $s_1$ , which corresponds to the situation in Table 17 ( $V_1$  is delayed and still allocated to gate 1 that  $V_3$  cannot be re-assigned, which results in conflict):

Table 17 Conflicting assignment in initial policy due to delay

		$T_0$	$T_1$	$T_2$
Agent 1	Gate 1	$V_1$	Conflict	$V_3$
Agent 2	Gate 2	$V_2$		

Simple experimentation is done to demonstrate the feasibility of the suggested resolution method.

The initial policy is not possible as a result of a delay of the flight  $V_1$  (Table 18), which causes a conflict in gate allocation. Therefore this solution is used as an initial policy in the policy iteration algorithm; then the algorithm is performed.

After the execution of the value iteration algorithm in MatLab, the provided solution offers another order in gate assignment, optimal policy is  $\pi^* = (a_3, a_1, a_1)$  identified as in table 18:

Table 18 Optimal policy

		$T_0$	$T_1$	$T_2$
Agent 1	Gate 1	$V_1$		
Agent 2	Gate 2	$V_2$		$V_3$

Table 18 shows that the proposed approach can give a solution that is more robust to delays. Compared with the sample agent MDP in [228], this approach is more representative of the problem structure because of the Multi-agent distribution of processing, that simplify its conception. Also, MMDP gives gate assignment configurations in multi-dimensional policies instead of having in MDP a single gate to flight assignment.

However, MMDP model gives only fixed time steps between decision epochs (iteration steps), which does not reveal the real evolution of gate assignment, where the time is different from the iteration step and always observable. Next sub-section gives an experiment with time dependency.

### 3.4.3 Time-Dependent Multi-Agent Model experiment

In this sub-section, experimentation is conducted on the Time-Dependent Multi-Agent MDP modeled earlier.

For simplification, every action possesses a single outcome. Hence actions and outcomes can be directly recognized ( $a_i \leftrightarrow \mu_i$ ) and actions thought to be deterministic with regard to the discrete component of the state. This is expressed as:

$\forall i$  Such that  $a_i$  is feasible in state  $s$ ,  $L(\mu_i|s, t, a_i) = 1$

A real data from six flights of Hong Kong international airport is used, as in Table 13; three gates are dedicated to those flights.

A Gate conflict is detected between flights LH738/739 and SQ862/861 due to some disturbance.

Starting with a specific state of the system  $s_1$  corresponding to the airport gate assignment:  $s_1 = (CA101/102, LH738/739, TG600/601)$

Moreover, exploiting other possible actions is done to apply adaptive assignment to arriving flights representing a change in gate configuration.

- $a_1 = (BR869/870, JL710/702, SQ862/861)$

- $a_2 = (\text{BR869/870}, \text{SQ862/861}, \text{JL710/702})$
- $a_3 = (\text{JL710/702}, \text{SQ862/861}, \text{JL710/702})$

Figure 46 below shows the state transition corresponding diagram.

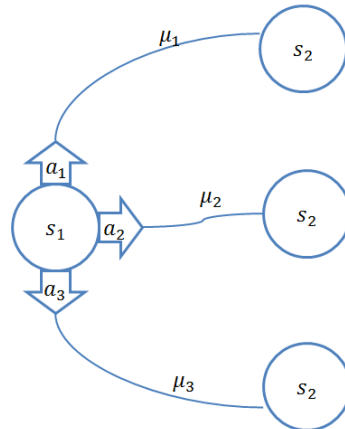


Figure 46 State transition diagram

Just for simplification, all outcomes have the parameter  $T_\mu = ABS$ . So, outcomes with durations are not considered. The probability density functions  $P_\mu$  are the defined for every outcome (see as example Figure 47).

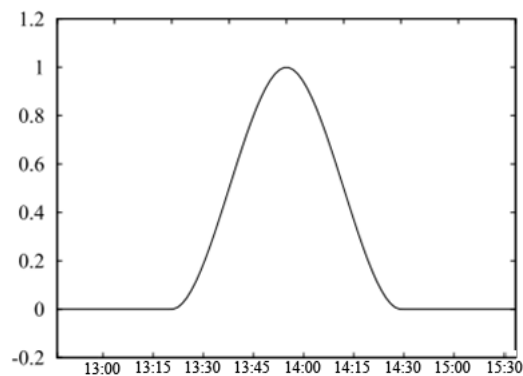


Figure 47 probability density functions of  $\mu_2$

This probability includes stochastic information related to action execution. Rewards are given in a way to score every action of assignment in the airport.

So, implementing the resolution algorithm, the value iteration algorithm gives an exact resolution [150]. The given solution consists of time-dependent policy choosing outcome  $\mu_2$  that avoids the disturbance situation. Then, the solution given by this approach is robust and handles flight delays. The fact of including the information about the possible disturbances improve more the GAP solution quality.

## 4. CONCLUSION

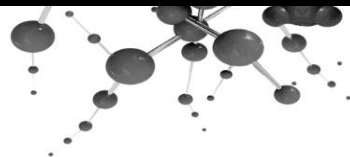
The proposed MDP model aims to constitute a robust mechanism that will give a time valuated approach dealing with disturbances in every time sequence. The provided solution is all of the decisions at every time that could be performed at the time of the planning horizon of flights assignment. This kind of model takes into account real-time optimization because it assumes to have a solution at every time which manages disturbances.

Experimentations on this approach using real sample data by simulation of the associated value iteration algorithm provides the best feasible solution that the deterministic model.

The aim behind this reflection is to offer to controllers at the airport a robust time valuated solution that takes into consideration possibilities of gate conflict, even if may take more time for resolution, it can manage the risk of disturbances in gate assignment. Also; Markov decision processes and its variants as proposed in this chapter, provide a suitable modeling tool for the gate assignment problem. The model flexibility of MDP and its stochastic behavior can imitate possible disturbances in the flight traffic. Otherwise, a reinforcement multi-agent learning approach, like the HMM-QPSO, which is given in chapter 2, can also be used to solve the problem as a perspective and an extension to this work.

As an additional perspective, this reflection about this type of model can be more extended to consider other real constraints of gate assignment.

---



# CHAPTER V: STOCHASTIC MODEL OF THE CREW PAIRING PROBLEM AND A TUNED PSO SOLUTION

---

Several complex scheduling problems are considered as challenging chore in The airline industry; this includes a multitude of operational decision and problems and deals with a significant amount of very interdependent resources. In this chapter, we consider the problem of crew pairing, where the crew have to be allocated to flights in a schedule with a minimal cost. The objective behind crew pairing is to determine a set of pairings or also known as Tours of Duty designed for a crew group to minimize the planned cost of executing a flight schedule.

The primary objective that aims to guarantee optimal allocation of crews to flights, should select the set of pairings with a minimal planning cost. The frequently applied algorithms suppose having no disruptions in planning operations. Nevertheless, airline operations usually experience stochastic disturbances that must be considered to be able to minimize the real operating cost. Most recently, impressive interest has been directed at robust crew scheduling with the attention of the stochastic character of disruptions such as bad weather conditions or maybe technical breakdowns. The purpose is to use stochastic information to obtain more robust solutions that got higher withstand disruptions. This sort of resolution methodology is necessary since we can proactively consider the impact of particular scheduling decisions. By finding out more robust schedules, cascading delay impact is reduced. In this chapter, a stochastic model of the crew pairing problem is provided based on multi-agent Markov decision process; hence, the problem would be processed as selecting the optimal policy to work with in stochastic situations of disruptions. We also perform a computational study to make sure of the validity of our proposed model. Additionally, to the stochastic model of crew pairing, a resolution method is proposed as an application to the PSO tuned method by HMM as an application to the first chapter.



## 1. THE CREW PAIRING PROBLEM

The purpose of this section is to present an insight into the manpower of airline planning generally. The focus will be on the components most useful for the particular problem of crew pairing for airlines. Before starting crew-scheduling problem resolution, series of planning problems have to be solved in airline scheduling: first of all, decisions in the schedule design problem identify the schedule of flights the airline manages. Every flight has to be described by an origin, destination, departure date, departure time and then a duration. Supposed the set of flights in the schedule, a solution of fleet assignment model establishes which aircraft type executes every flight. The goal is to maximize profit with respect to the number of available aircraft and additional resource constraints. After that, the aircraft routing problem searches for a minimal cost assignment of available aircraft for flights. A routing is allocated to every individual aircraft in a way that every single flight is included in specifically one routing. Routings have to fulfill maintenance constraints. Once aircraft types are assigned to flights, the aircraft routing problem is usually solved for every aircraft type independently. In an identical method to the aircraft routing problem, the crew pairing problem (or also called Tour-of-Duty planning problem TOD) which usually includes building sequences of flights to crew the flight schedule and assigns crew to flights in a minimal cost. A set of generic crew pairings is built being subject to various rules in order that every flight is covered precisely once. Within the supposition that the crew is only permitted to operate a single aircraft type (that is commonly the case for pilots, for instance) the crew pairing problem can likewise be solved independently for each aircraft type. All of the flights in the given schedule during the planning horizon are partitioned into sequences of flights. In every single sequence of flights that a crew member can fly, it is known as a Tour-of-Duty (ToD) or pairing, referred to as rostering in the recent airline planning problems. The built pairings are assigned, along with other types of activities, to the actual crew, in conformity with the qualifications and already assigned activities, known as pre-assignments, of the crew. The goal is to find feasible assignments that minimize costs taking into consideration perturbations in planning. A problematic is addressed before providing details of the planning process and crew pairing as follow.

### 1.1. PROBLEMATIC

Airline scheduling is associated with numerous complexities, integrating a network of flights, several aircraft types, air traffic control limitations, finite numbers of gates, environmental regulations, rigorous safety preferences, a numerous of crew work guidelines and complex payment components,

and dynamic situations of the environment where passenger requirements are uncertain, and pricing tactics are complicated.

Airline operations are subjected to diverse sources of disruptions like airspace congestion, bad weather conditions, or some technical breakdowns, etc. In many of these situations, the resource schedules could possibly be disrupted, so that these are potential to become infeasible. Disruptions require fast recovery measures that result in flight delays or cancellations to recover many resources that are necessary for operating flights (see [8]). Rather than taking the risk of online disruption management, robust planning has been considered by some airlines as an effective way of handling possible disruptions in their schedules.

The availability of crew is certainly one of the critical triggers of delays in the airline operations. This can be especially true if the crew switch from one aircraft to one other at the time of a duty period, most importantly if there exists a minimal ground time in between flights. In the event of the arriving flight that has become delayed for whatever reason, in that case both equally that aircraft and the next aircraft that on which the crew is swapping will leave late in time. This is a kind of delay propagation that can trigger more severe disturbances in the operational flight schedule. Our proposed methodology attempts to manage this problematic.

## 1.2. AIRLINE RESOURCE PLANNING PROCESS

In large airlines, the planning and scheduling of aircraft and crews is a highly difficult process and is consequently commonly split up into many planning phases. The planning process is detailed in Figure 48, which displays the logical sequential sequence of these phases, where the solution from one stage uses the data for the next. In fact, The four stages are typically assigned to four distinct divisions which simultaneously work on their part of the resource planning problem. Each one of these divisions communicate to be able to adapt their plans to each other and to update each other with variations. The first work of plans for a given planning period may be updated several times until the last plans are fixed and published. In the operational phase, these plans may still be altered. To illustrate, if a flight may be canceled or delayed and crew may report sick. Airlines deal with these problems by having some buffer resources (for example stand-by crew) and by using robustness of the plans as one of the problems to be optimized



Figure 48 The resource planning process in airlines

Each stage of the planning process (see figure 48) is described as follow :

- **Timetable construction:** First of all, the timetable is designed. Its objective is to meet the goals of the marketing division with the available fleets and with constraints on the network, such as the time slots available for the airline at distinct airports. The result of this process is an amount of flight legs (as non-stop flights) which the airline chooses to perform.
- **Fleet assignment:** Second, the resolution of the assignment of aircraft to the flight legs is made. The estimated income of a given flight leg is determined by the size of the aircraft attached to the leg. Likewise, some aircraft would possibly not have the ability to maneuver from some airports. The main concern is to guarantee the feasibility of the timetable provided the available fleet. If this is unattainable, the timetable has to be modified definitely. Provided feasibility the objective is to maximize the expected income decreased by operating costs [232].
- **Crew pairing:** Third, pairings are built. A pairing is, in fact, a sequence of flight legs for some crew member beginning and finishing at the equal crew base. The crew member will often be operating on these legs; however, a pairing may as well include what is named deadheads, when the crew member is not operating yet is just moved as a passenger. Legs are normally gathered in and known as duty periods, which are split up by lay-overs or also named overnight stops. In short and medium transport problems a pairing may contain one as long as five duty periods. In long transport problems, longer pairings are generally allowed. Legal pairings have to satisfy a considerable amount of governmental regulations and collective arrangements which differ from airline to airline.
- **Crew assignment:** Fourth, it is about to delegate pairings to named persons. This problem is referred as the crew assignment or recently as the rostering problem. The objective is to cover all pairings along with working preferences, vacations etc., also constraints like work rules and regulations have to be satisfied. Additionally, crew costs are to be minimized. If the crew is remunerated a fixed salary, the problem aims to maximize the employment of crew so that to minimize the number of crew members.

After fuel costs, crew costs represent the major immediate operating cost of airlines. The crew costs rely upon the quality of the solution to the pairing problem in addition to the assignment problem, however, since it is not possible to replace with poor pairings in the assignment problem, it is appropriate to expect that cost savings in the pairings problem will result in cost savings in total crew costs. The next section presents the crew pairing.

### 1.3. CREW PAIRING

In general, a crew scheduling problem is split up into two problems. The first is crew pairing, and the other is crew assignment. Primary crew pairing problem is resolved, and good pairings are determined by providing flight schedule as an input. Subsequent to solving crew pairing problem, crews are allocated to each of these pairings. To build a good complete solution, the input data as crew pairings should have superior quality. A high quality of input data will not mean proper results although with poor input data practical solution is unattainable. For this reason, crew pairing is one of the most crucial stages of airlines planning even more than the crew scheduling problem.

#### 1.3.1 PROBLEM DESCRIPTION

Prior to detailing the problem a couple of definitions need to be provided. A flight leg is a single nonstop flight. A duty period is mainly an operating day of a crew that includes a sequence of flight legs with short rest periods. Likewise, the duty begins with a brief period and terminates with a debrief period. A pairing is a sequence of duties, and every pairing starts and terminates at the same crew base. In a pairing there exists an overnight rest between every single duty. Crew base is a city in which crews are stationed. To shift a crew from one base to another base, a pairing may include crews as passengers, and this sort of flight is called deadhead. Generally, deadheads employed to transport a crew wherever they are required to cover a flight or to go back to their residence base.

Commonly, the provided timetable which is usually the schedule of a month to the crew pairing problem may include daily, weekly and monthly flights. The crew pairing problem might also be described as daily, weekly and monthly problem to manage all types of flights in the timetable. The daily problem considers that all flights are flown each day. Every single pairing is flown by a different crew, and it begins daily. Additionally, in a daily problem, a flight cannot show up much more than on one occasion in a pairing. To consider flights that are not functioning every single day of the week, an altered version of the daily problem can be used, and it is known as a weekly problem. There are likewise other flights distinct from daily and weekly flights in the regular monthly schedule. Airlines may possibly change a number of its flights that were previously in the timetable. In this situation, a monthly problem is helpful to find crew schedule for covering up flights during a transition among older and newer flight schedule. In our research, the only daily problem is regarded.

In the crew pairing problem, the objective is to find a set of pairings from all likely pairings. When choosing pairings, all flights should be covered precisely once, and cost of the all picked pairings must be maintained at the minimum. Also chosen pairings must be legal depending on the required regulations.

The problem is commonly separated into two distinctive phases: pairing generation, and pairing optimization. Whereas pairing generation is simple, even for comparatively small instances millions of legal pairings may be generated. This big size problem produces a difficult to determine a comparatively small set of legal pairings with a minimum cost that covers all the flight legs in the schedule.

Applied Models for the crew pairing problem including deterministic and stochastic models will be illustrated in the next section of literature.

### **1.3.2 Crew restrictions**

The crew scheduling problem concerns setting crew members to flights. Crew scheduling at airlines encounters two sorts of restrictions. First of all, general restrictions on operating and flying time which usually applied for all crew members; the second sort of restrictions is applied to individual crew members or particular flights. For instance, those restrictions can be some particular language requirements on specified flights or also the unavailability of certain crew because of vacations.

To be legal, pairings have to follow governmental rules and collective agreements, functional considerations, and contractual rules that determine the structure and cost of legal duty periods and pairings, such as:

- Each crew pairing begins at one crew base with a briefing of 60 minutes and terminates at the same crew base with a de-briefing of 30 minutes.
- The minimum duration of a night rest is 10 hours.
- The maximal number of flying hours in a duty is 8.
- The maximum duration of a duty with flying tasks is 10 hours.

There are likewise other obligations besides pairing and duty rules just as crew base constraints. For every crew base, there exist several values of the total volume of flying hours that may be assigned to the crew. By working with this rule, it is guaranteed that crews at different bases will all have about the equal volume of flight hours for every single work month. In this study, we assume pairings are generated according to those rules and regulations. The next sub-section describes how pairing generation is performed.

### **1.3.3 Pairing Generation**

In pairing generation step, all likely duty periods are generated soon after the generation of duties. All possible pairings are generated by incorporating these duties. In duty generation, for every flight leg, a tree is built as their root node given by that flight leg. The root node has children that symbolize every single possible linking flight legs. At each consecutive level inside the tree, every single node has a child for nearly every possible connection. The depth of the tree could be established by the maximum

permitted number of flights during a duty. By way of depth-first-search procedure, going to every node of the tree, all feasible duty periods are determined. Each route coming from the root node to some of its descendants in the tree denotes a duty period. Workable duty periods are primary checked to make sure that they are legal according to the legal restrictions. If a duty is legal, its own cost is computed [233].

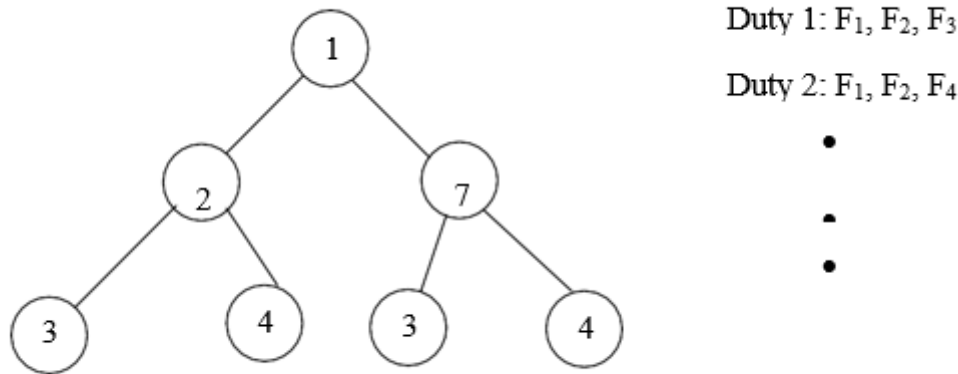


Figure 49 Example of flight tree for a flight

Pairings could also be identified by pursuing the precise same procedure. In such a case, a tree is built for all duty and duty periods are symbolized by nodes in the tree (see Figure 49). Likewise once again all feasible pairings are examined to not break regulation rules but exclusively the rules that are unable to be used in duties since all applicable rules are previously inspected when generating duties. One of many instances of those rules is the fact a pairing should begin and end at the same crew base. Additionally a pairing may not include a flight multiply. Therefore, any duty cannot be pursued by another duty containing an equal flight with the first one. When a pairing is legal therefore its cost is as well computed.

### 1.3.4 Crew Pairing Optimization

Crew pairing realizes minimum cost generic strings of duties that start and finish at crew bases. Through this problem feasible pairings are evaluated with regards to the costs of layovers (overnight stays on outstations), time away from the base and various other incremental costs. Pairings are built from duty periods which are usually mainly shifts or a day's work. Duties have to start from where a prior duty ended, except if dead-heading (crew carried as passengers) is utilized. The crew pairing problem is crucial from a reserve crew scheduling perception since it establishes how detrimental uncovered crew-related delays could be, within conditions of cancellation because of crew absence and just how uncovered delays might propagate throughout the schedule.

A solution to a pairing problem is a number of pairings so that all flights in the planning horizon are covered as many times as required. We will represent this as a fully dated solution as it identifies particular days for the process of the pairings. This problem is not attacked straight, as the number of flight legs inside a planning horizon of one month is, for instance, multiple thousands. Instead, a sequence of three problems of which the first two are estimated of the fully dated problem is solved.

The supposition of the daily problem is that the timetable is the same every day, and practically all research on crew pairing treated this problem. The implied supposition is that a good daily solution is the most difficult component of the problem. Additional presumptions may be found including the timetable that repeats itself every single week. In this chapter, we consider the daily problem. A solution to the daily problem includes a number of pairings so that every leg is covered during one day. From this solution, it is possible to build a fully dated solution which usually repeats itself daily.

#### **1.4. ROBUST SCHEDULING**

Besides disruption control on the day of operations, the situation of disruptions could also be taken into consideration at the time of the schedule construction. This is known as robust scheduling. The objective of robust scheduling is to build schedules which stay feasible and close to optimal cost within different modifications of the working environment.

Measures of the robustness of a schedule have been tackled in the literature. The work in [234] differentiates two facets of robustness: stability and flexibility. While stable schedules are most likely to stay feasible and near-cost optimal in a varying functioning environment, flexible schedules can be effectively designed to a varying operating environment. Robust schedules also can be stable and flexible simultaneously. In the airline subject, this involves that a schedule is stable given that it is still feasible at reasonable costs with no kind of manual adjustments to the schedules. A schedule is flexible if in the event of disruption recovery actions are likely to be able to reestablish the operational feasibility of the schedule while not noticeably raising costs. In the situation of crew and aircraft schedules, reactionary delays may be regarded as a manual recovery action as well as an automatic consequence if no manual recovery is performed. Thus the number and duration of reactionary delays is a primary measure of the stability and flexibility of airline schedules. Further measures of the stability of a schedule are the amount of disruptions which are unable to be automatically fixed by the reactionary delay. Supplemental measures for the flexibility of a schedule are the costs of recovery as well as the impact on other schedules as found in [235].

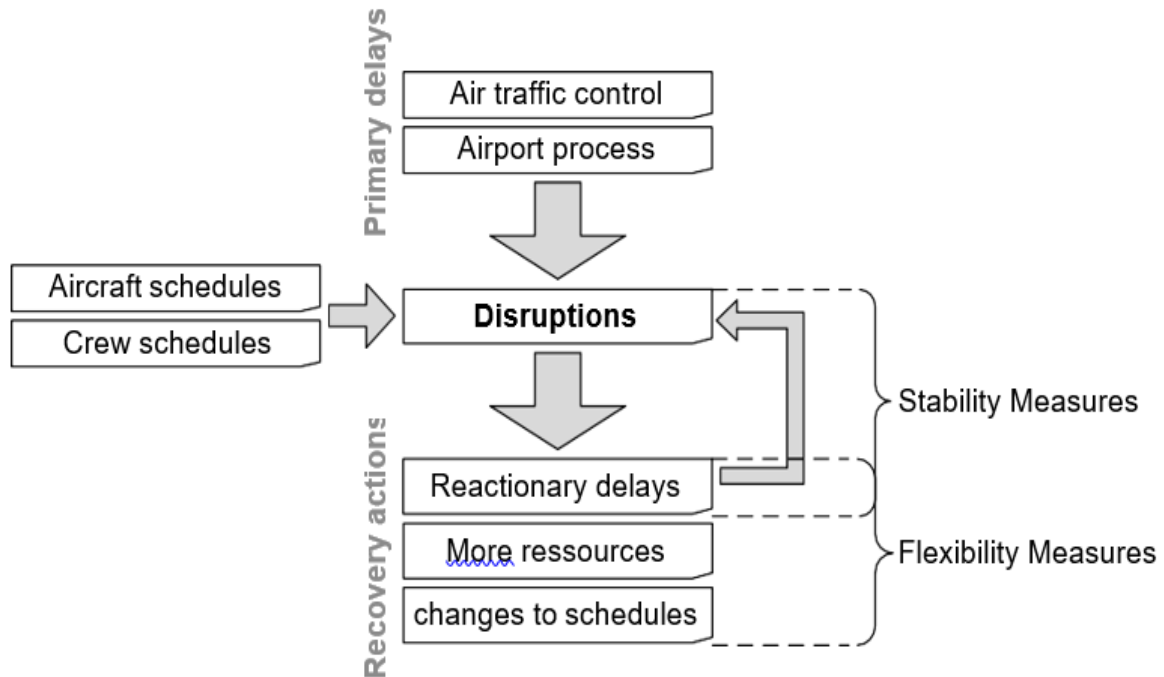


Figure 50 Robustness in air scheduling [235]

Figure 50 summarizes the robust measures in air transport, displaying the relationship between disruptions for aircraft and crew schedules. Further sources of prime delays and further recovery actions can be taken into consideration. In our approach, we try out to build a more stable model of crew pairing.

The different approaches dealing with crew pairing optimization problem are defined in the next section.

## 2. LITERATURE REVIEW

Crew planning and scheduling problem is the designation of the number of crew with particular skills and assignment of the crew to satisfy the demand to ensure that total cost is minimized when regulatory and legal restrictions are attained. The crew scheduling is an NP-hard constrained combinatorial optimization problem, and therefore, it cannot be exactly solved in an affordable computational time. Also, crew scheduling is easily influenced by disturbances like delays or mechanical failure. This makes the resolution more and more complicated when considering the stochastic events that cause distributions.



## 2.1 WORKS FROM LITERATURE

In the literature of airline scheduling, the crew scheduling problem has been usually split into crew pairing and crew rostering [236]. Our focus is granted to the crew pairing problem, which is described as selecting pairings in a way that all flight legs are covered at minimum cost. Authors in [237] have examined the major differences in terms of crew categories, fleet types, network structures, rules and regulations, regularity of flight timetables, and cost structures.

Usually, the aircraft routing problem is solved prior to the crew-pairing problem, although the two problems are interdependent. However, pairing construction and pairing assignment can be done in a single step, see Claude et al. [238], provide solution techniques based on simple tree search with a column generation and shortest-path algorithms.

Generally, the crew pairing problem is resolved in two stages: first the generation of all legal pairings and their associated costs calculated. Then the best subset of these pairings is selected to cover all the flight legs. [239, 240] proposed a smart enumeration and compact storage schemes for pairings to be of moderate size.

The main challenge in this problem is that there is not any general method to work well with all kinds of nonlinear cost functions and constraints [241]. Meanwhile, this problem goes to a complicated problem when the number of inputs increases.

The crew pairing problem is commonly modeled as a set partitioning problem [242]. Most of crew scheduling models observed in the literature are thought to be deterministic in a stationary environment. Several approaches deal with major difficulty in solving crew pairing that is a large number of pairings, which render it hard to solve exactly.

[243] provided a linear cost structure to solve the daily crew pairing problem applying a heuristic method for branching based on graph theory to deal with large size problems. In [244], they chose a nonlinear cost structure to each pairing cost to solve the crew pairing problem by column generation with branch and bound approach. Formulated in [245] as an integer nonlinear multicommodity network flow problem, crew pairing has been solved as a set partitioning problem.

A new pricing scheme for the column generation approach is proposed by [246]. [247] used a new deterministic formulation of the daily problem with a nonlinear cost structure for each pairing.

[248] implemented the set partitioning model with additional constraints and [249] proposed an integer programming model for the crew pairing problem making use of CPLEX for resolution.

Most of this extensive variety of deterministic models have developed an improvement in finding better crew schedules; however, crew scheduling and air transport in general frequently have to deal with disruptions. In

several approaches, disruptions are regarded as a divided problem of crew recovery or also called disruption management [8]. The objective is to rectify the broken pairings so that the total process can return to the original schedule successfully within a minimum period.

[250] applied a FIFO procedure to assign crew members to new flight aircraft schedules. [251] evolved a heuristic-based search algorithm for the recovery problem. [252] offered an integer programming model for the problem with LP relaxations.

Recently, [253] considered the flight aircraft and crew rescheduling problems simultaneously instead of sequentially. The objective is to minimize the complete cost produced by flight delays and cancellations and crew schedule adjustments.

Recovery plans could result in additional costs called reactionary, which is usually larger than planned costs. Recent approaches from literature addressed this problem by robust crew scheduling, where potential disruptions are already included throughout the scheduling operation.

An approach to integrate stability into crew schedules has been given by [254] using the expected costs computed by simulations of standalone pairings. This approach limits delay propagations between pairings. Another bi-criteria approach is given by [255] to resolve the robust crew pairing problem by taking into account the stability features of crews switching aircraft.

[256] makes use of a non-robustness indicator for iterative aircraft routing and crew scheduling. [257] elaborates a crew pairing model with non-linear stochastic recourse to evaluate probable delay propagations among both of crews and aircraft.

Also in [258], a stochastic model is built with delay propagation integrating an indicator for the stability of airline crew and aircraft schedules. The formulation has been divided into linear problems connected by the objective function.

A further robust formulation in [259] where authors suppose that flight and connection times are random and vary within an interval. The resolution is performed applying a Lagrangian relaxation to detach the nonlinear terms in the sub-problem contributing to a new robust formulation of the shortest path problem having resource constraints.

## **2.2 BASIC MODELS OF LITERATURE:**

In this sub-section, two main models of crew pairing found in the literature are presented, namely, one deterministic model and another stochastic model.

### 2.2.1 Deterministic model

A crew schedule is a set of pairings that partitions the legs to be flown by a singular fleet. Crew scheduling problems are solved by generating pairings and solving an integer program. The daily crew scheduling problem is solved within the supposition that each leg is flown every single day.

The crew scheduling problem is generally modeled as a set partitioning problem the deterministic crew pairing problem is identified as follow [260]:

$$\min \sum_{j=1}^n c_j x_j \quad (68)$$

$$\text{Subject to: } Px = e \quad (69)$$

$$x_j \in \{0,1\} \text{ for } j = 1,2,\dots,n$$

The right-hand side vector  $e$  is a vector with  $m$  entries equal to one. Every single row of the matrix  $P$  represents a flight leg, and every single column symbolizes a legal pairing. The cost vector  $c$  is so that  $c_j$  is the cost related with the  $j$ th column or pairing. The binary decision vector  $x$  is so that  $x_j$  is a 0-1 variable related to the  $j^{th}$  pairing. If a column  $j$  is designated  $x_j = 1$ , otherwise and  $x_j = 0$ . The matrix  $P$  is created as follows:

$$p_{ij} = \begin{cases} 1 & \text{if flight leg is covered by pairing } j \\ 0 & \text{otherwise} \end{cases} \quad (70)$$

Further constraints on the crew-scheduling problem named the “crew base constraints” may be requested at times. These constraints limit the total number of flying hours that a crew can use away from its base location to be inside specified bounds. These constraints significantly restrict the allocation of available crews between flights and lead to the difficulty of dealing with the crew scheduling problem. So that to resolve this issue, the proposed approach in the next section will try to enhance the resolution of the deterministic problem and reduce its complexity by a tuned PSO based resolution.

### 2.2.2 Stochastic model

This method manages the assignments of the crew during the planning phase to keep reducing the cost of the crew when establishing more robust solutions by discovering pairings that are less sensitive to disturbance planning. Late costs supplied in the deterministic objective function reflect how delays impact constraints on flight legs.

Assuming that disturbances are observable with cost recovery, the disturbance is modeled as a rise in the time of flight operations, just like ground time and air time. These times are a random vector  $\xi(\omega)$ , where  $\omega$  is

a random element in some space  $\Omega$ . Each  $\omega$  represents a scenario of disruption. Moreover, for each  $\omega$  there is a different use.  $\Omega$  is finite and each cardinal  $\omega$  happens with probability  $p_\omega$ .

The formulation of the stochastic program will, therefore:

$$\text{minimize } c^T x + Q(x) \quad (71)$$

$$\text{subject to : } Ax + b \quad (72)$$

$$x \in \{0,1\}, x \text{ integer}$$

$c$ : vector that represents the single expected cost of flying in a pairing without consideration of other pairings. This is the cost of the link without considering delays due to switching between aircraft crews.

$Ax = b$ : Limits of coverage and also other constraints of the crew.

$$Q(x) = \int_{\omega} R\omega Q(x, \omega)P(d\omega) \quad (73)$$

Where:  $Q$  is expected future actions due to disruptions in the original schedule value.

This formulation is a standard two-stage stochastic program recourse (as in [257]).

Also, [261] formulate a stochastic model with delay propagation measuring for the crew pairing problem. They propose a decomposition strategy to be able to solve the model by iterative crew scheduling and aircraft routing adapted from [256].

[262] proposes recovery policies in a random environment for the crew problem based on semi-Markov process.

### 3. TUNING RESOLUTION METHOD BY PSO

The aim of this section is to adapt the formulation of the HMM-based tuner to the crew scheduling problem. Therefore, we depict the classical crew pairing model, and we describe the PSO algorithm used to solve this model. This resolution constitutes a direct application of the tuning method described in the first chapter.

#### 3.1 EXPERIMENTAL MODEL AND TUNING ALGORITHM

We describe both the model to resolve and the algorithm tuned for resolution.

### 3.1.1 The crew scheduling problem: Crew pairing

The objective of the crew scheduling problem is to determine a minimum-cost set of pairings so that every flight leg is assigned a qualified crew and every pairing satisfies the set of applicable work rules.

In particular, crew pairing is very often considered as the first part of the airline crew scheduling procedure.

The crew pairing problem can mostly be formulated as a set covering problem (SCP) or set partitioning problem (SPP). The SPP formulation can be expressed as follows [260]:

$$\min \sum_{j \in P} c_j x_j \quad (74)$$

$$\text{Subject to } \begin{cases} \sum_{j \in P} a_{ij} x_j = 1 \quad \forall i \in F \\ x_j \in \{0,1\} \quad \forall j \in P \end{cases} \quad (75)$$

The description of the parameters is detailed in [260], as well as in the previous section.

### 3.1.2 Binary particle swarm optimization

The binary PSO (BPSO) algorithm was introduced by [263] to enable PSO to operate in discrete and binary search spaces. It follows the same approach of the canonical PSO. Each particle  $i$  adjusts its velocity  $v_i$  and position  $x_i$  in each generation  $t$  according to the following formula: (the description of PSO parameters can be found for instance in [263])

$$\begin{cases} v_i^j(t+1) = w v_i^j(t) + c_1 r_1 (p_i^j(t) - x_i^j(t)) \\ \quad + c_2 r_2 (p_g^j(t) - x_i^j(t)) \\ v_i^j(t+1) = \text{sig}(v_i^j(t)) = \frac{1}{1 + e^{-v_i^j(t)}} \end{cases} \quad (76)$$

$$\begin{cases} x_i^j(t+1) = 1 \text{ if } \text{rand}_i \leq \text{sig}(v_i^j(t+1)) \\ x_i^j(t+1) = 0 \text{ else} \end{cases} \quad (77)$$

## 3.2 GENERATION OF INSTANCES AND CONFIGURATIONS OF CREW PAIRING

Instance generation is an important issue in the tuning problem. In this study, we have generated problem instances manually as following (table 20):

*Table 19 Instances of the crew problem*

Instance	1	2	3	4	5	6	7
Flight leg	10	14	20	20	35	40	49
Number of pairing	8	8	10	20	8	20	8

Concerning the definition of configurations, as presented in the literature section, various approaches have been proposed to define PSO parameters. In this paragraph, we have adopted a number of commonly used values to define the possible configurations of the PSO algorithm. On the one hand, for population size, we consider the work in [264] to define the first and last configuration (100 and 10). The other one is the most adopted in the literature. On the other hand, for the acceleration factors, the difference between these configurations corresponds to the degree of exploration and exploitation. That is, the first configuration enhances the exploration of the algorithm at the expense of rapid convergence while the second one gives a better ability to exploitation. The last one presents a trade-off between exploration and exploitation [265]. In the two first propositions, we have respected the proposition that  $c_1 + c_2 < 4$  as in [61] and for the last one, we have adopted  $c_1 + c_2 > 4$  as proposed in [266].

*Table 20 Configuration parameters*

Parameter values	1	2	3
Population size	100	50	10
$(c_1, c_2)$	(2.5, 0.5)	(0.5, 2.5)	(2.05, 2.05)

We examine all possible combinations of these parameters of table 21 (nine configurations) which are: [(100, (2.5, 0.5)), (100, (0.5, 2.5)), (100, (2.05, 2.05)), (50, (2.5, 0.5)), (50, (0.5, 2.5)), (50, (2.05, 2.05)), (10, (2.5, 0.5)), (10, (0.5, 2.5)), (10, (2.05, 2.05))].

### 3.3 TUNING TEST PHASE

After defining configuration and instances of the crew problem, tuning simulations of the configurations over instances are executed according to Algorithm 2 which is defined in chapter 1.

Before running tests, and given a maximum running time for the PSO iterations number, we define two metrics to be used for performance evaluation:

- Metric 1: defined as the cost at termination.
- Metric 2: defined as the running time.

The tuning approach is tested over these two metrics. Also, a comparison of running tests is made by a ranking function corresponding to mean executions of a specific instance with a specific configuration. Then, the instances can have metric values, based on information regarding cost and running time on the crew scheduling problem instances.

Tests are performed twenty times on each configuration. Moreover, instance and mean values of the metrics are retained and ranked according to the order of each configuration performance on an instance. Algorithm 2 of the test phase is executed multiple times to obtain sufficient data for the next phase.

Table 22 gives an example of the results obtained from one test phase execution on instance 7:

*Table 21 Example of one test output*

Configuration	Metric 1	Rank of Metric 1	Metric 2	Rank of Metric 2
1	391	4	0.50	6
2	244	1	0.53	9
3	289	3	0.49	1
4	301	8	0.49	2
5	293	5	0.51	7
6	339	9	0.50	5
7	298	6	0.50	3
8	244	1	0.50	4
9	300	7	0.51	8

This execution shown in the table above is executed twenty times, and ranks are considered as a stream data of observation sequences and states of HMM.

### 3.4 TUNING EVALUATION PHASE

We consider all execution data from the previous phase as an input for this evaluation phase. As mentioned before, tuning is performed according to two metrics which are the running time and the termination cost.

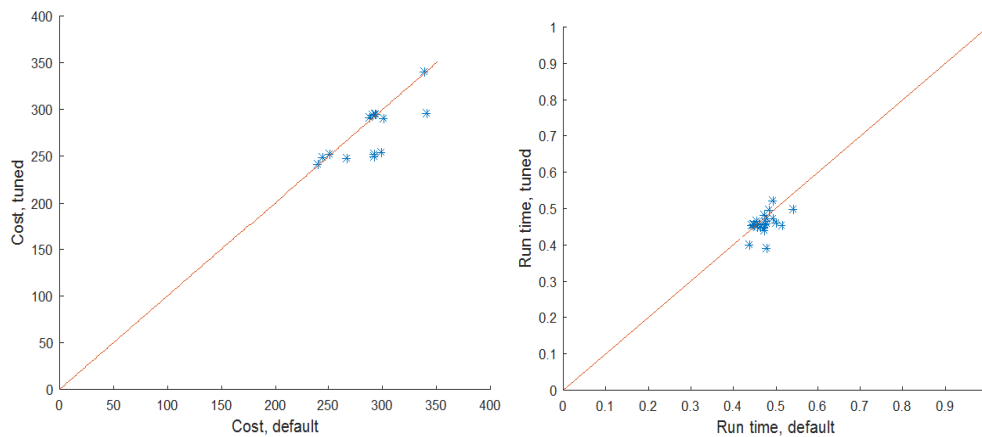
The stream data of ranks for the two metrics are considered as observation sequence for HMM, and corresponding configurations for each rank are states of HMM.

So, we execute Baum-welch algorithm on the data of the test phase. We drop the results of the tuning in Table 23.

*Table 22 Representation of probabilities for each metric*

configuration	Metric 1	Metric 2
1	0.75	0.12
2	0.76	0.10
3	0.77	0.10
4	0.79	0.06
5	0.75	0.10
6	0.82	0.18
7	0.80	0.07
8	0.77	0.15
9	0.79	0.09

Table 23 shows the probabilities of each configuration to be the best (rank one) for the two metrics: cost and run time. Maximum probability of rank one for tuning by metric 1 of run time is given by configuration 6, and maximum probability of rank one for tuning by metric 2 of costs is configuration 6 also.

*Figure 51 Comparison of default and tuned configurations by cost and runtime (instance 7).*

Furthermore, as in [267], we adopt a graphical comparison between the HMM-based tuner and the classical PSO (non tuned) as depicted in Figure 52. That is, after testing obtained best configuration for each metric and comparing to a default configuration (Figure 51), the configuration given by our tuning method gives greatest results in terms of cost and time. So, HMM can successfully tune PSO metaheuristic for the crew problem.



## 4. MARKOV DECISION PROCESS MODEL OF CREW PAIRING

Many contributions in the literature are trying to deal with uncertainty (see the literature paragraph). With the same target of building a robust mechanism that can absorb flight disturbance, we develop this approach based on multi-Agent Markov Decision Process.

### 4.1 WHY MDP MODEL FOR THE CREW PAIRING

Schaefer et al. [268] talk about the possibility of modeling robust crew scheduling under uncertainty based on a Markov decision process (MDP) but assume that is intractable. As Schaefer describes, the state of the system gives every aspect of the system that is relevant for operational decisions. Thus, this state contains information about the current status and the current operating environment such as weather, congestion, etc. The first stage consists of the planning period, where we have the initial crew schedule. Operational decisions are made in subsequent stages. The number of stages is quite large since the planning horizon is the same as the planning time and the state of the system can change within minutes. Also, the action space consists of all possible, feasible decisions such as canceling flights, rerouting planes and passengers, rescheduling crews, and so on. So, the number of states and actions is so much big, which justifies why this approach is intractable and can engender a precious resolution time. Regarding how much the problem is critical and the fact that it does not allow a wait time for a decision establishing in every stage, an online resolution would not happen without serious risk.

Based on those critics of MDP formulation, we try here to give another formulation of pairing problem in crew scheduling. We distinguish first between the planning horizon and planning time. Therefore, the approach given below has prior solution and does not wait until the online mechanism.

### 4.2 THE MDP MODEL

This theory gives as a solution to a centralized controller, in advance, all decisions that can be made during the planning horizon of legs assignment to pairings. So, with MMDP, no need to real-time optimization because we assume having predefined solution that is the best possible that can handle disturbances.

We consider as a stochastic process for the MDP, the flight arrivals as in the previous chapter. It takes its values from the state space. The corresponding solution is defined by a policy given by resolving the following MMDP.

With this purpose, we formulate the pairing assignment problem as a tuple  $\langle K, S, A, P, R \rangle$  where :

The set of states  $S = \{s_1, \dots, s_N\}$  where  $S$  gives all flight legs object of our problem, they form the set of flights to be covered. A flight leg is defined as an airport to airport flight segment that starts at a specific departure time and connects two stops of a flight. A state  $s_i \in S$  is a flight leg characterized by departure and arrival time between two stations.

The set of actions  $A = A_1 \times \dots \times A_N$  would be the set of all possible pairings,  $N$  is the number of pairings. A pairing is defined as a sequence of flight legs or segments that begin and end at a crew base.

A pairing  $A_l = \{a_{l1}, \dots, a_{lN}\}$  where  $A_l \in A$  and  $a_i \in \{0,1\}$ ,  $a_i = 0$  if the pairing  $l$  covers the leg  $i$  and 0 otherwise, in other words,  $a_i$  defines if the agent  $A_{g_i}$  will cover the leg  $i$ .

At a state  $s_i$ , possible actions are those pairings that can cover the leg  $i$ .

$A_l$  is a possible action in state  $s_i$  if  $a_{li} = 1$ .

Also every agent  $A_{g_i}$  can cover only one and unique leg.

The transition function gives the probability  $P(s, a, s')$  of the system transition to a state  $s'$  when agents run the joint action  $a \in A$  from state  $s$ . It corresponds to a probability of changing the assignment combination from  $s$  to  $s'$  resulting from executing a reassignment action. That means the probability of reassigning the pairing  $a$  to the leg  $s'$  while it was assigned to  $s$ .

We assume that the probability  $P$  includes the completely stochastic information for the pairing assignment problem. It aggregates possibilities of disturbances such as stochastic flight delay or also other probabilistic disturbances that affect pairing assignment. The computation of this probability uses other techniques to form a probabilistic model of crew disturbances.

$R(s, a, s')$  is the reward obtained by the system when changing from one state  $s$  to a state  $s'$  by executing the action  $a$ . This includes negative costs and positive benefits of every reassignment.

We need so to evaluate policies in order to have the optimal policy and solve the MDP problem. The two main algorithms used for resolution are: value iteration (Bellman[141]) and policy iteration (Howard [142]).

We define the value function  $V_\pi: S \rightarrow \mathfrak{R}$ , which represents the expected objective value obtained following policy  $\pi$  from each state in  $S$ .  $V_\pi$  is given as follows [142] :

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V_\pi(s') \quad (78)$$

At least one optimal policy exists, and all optimal policies have the same value function  $V^*$ [142]:

$$V^*(s) = \max_{a \in A} (R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s')) \quad (79)$$

(78) and (79) are called Bellman equations, where  $\gamma$  ( $0 < \gamma < 1$ ) is a discount factor that is used to give more or less importance to future rewards. This factor is required to consider an infinite horizon discounted model (Kaelbling et al. [166]) in which the number of iterations is in principle infinite. However, the discount factor  $\gamma$  implies the convergence in a polynomial number of steps (Givan[269]). Over iterations, the initial policy is successively improved until an optimal one.

To solve the MMDP associated with the pairing problem, we use the value function based on Howard algorithm (Howard [142]).

According to Howard [142], this algorithm is guaranteed to converge.

Then, the solution is about finding an optimal stationary policy  $\pi^*$  for all agents that maximize the expected discounted future reward for each state  $s$ . The  $\pi^*$  includes optimal decisions to make regarding the assignment state. The optimal policy gives the set of optimal pairing to select. It can be regarded as optimal actions to take for pairing assignment.

### 4.3 EXPERIMENTATION

We conduct a computational study to test the effectiveness of the implemented algorithm. We use sample data from [240]

*Table 23 FLIGHT SCHEDULE*

Leg number	Dep. station	Dep. time	Arr. station	Arr. time
1	A	08:00	B	13:00
2	B	15:00	A	20:00
3	A	07:00	C	10:00
4	C	12:00	A	15:00
5	B	07:00	C	10:00
6	C	11:00	B	14:00

Table 24 shows a flight schedule of six flight legs to be flown every day of the month. A row in the flight schedule represents a single flight leg defined by single flight number from one to six.

Just for illustration, only some simple rules are used to classify legal pairings such as the number of duties in a pairing, the overnight rest (OR) between duties and the total flying time of a pairing. Thus, legal pairing are described as follows (table 25):

*Table 24 Legal pairings*

Pairing number	Flight leg sequence
1	1 $\Rightarrow$ OR $\Rightarrow$ 2
2	1 $\Rightarrow$ OR $\Rightarrow$ 5 $\Rightarrow$ 4
3	3 $\Rightarrow$ 4
4	3 $\Rightarrow$ OR $\Rightarrow$ 4
5	3 $\Rightarrow$ 6 $\Rightarrow$ OR $\Rightarrow$ 5 $\Rightarrow$ 4
6	3 $\Rightarrow$ OR $\Rightarrow$ 6 $\Rightarrow$ 2

The problem is first solved as a set partitioning problem (SSP) [239] to have an initial policy according to a given cost associated with every pairing assignment. The initial policy would be: pairings one and five. This solution is optimal under the assumption of no disturbances and deterministic resolution.

If we assume now having disturbance as delay in leg 3, and we have a serious possibility of disturbances between leg 3 (arrives at 10:00) and 6 (departure at 11:00) included in the chosen pairing 5. So, if we choose this policy, it will be a missed connection between 3 and 6. This probability is included in our model as a transition probability between state 3 and 6 executing action 5.

Now we use the Howard algorithm of policy iteration [142] on the initial policy. The result of resolution including potential probabilities of distortions is pairings two and six. This is a better policy that can handle disturbances between legs 3 and 6. The delayed flight leg in this new resolution is isolated between two nights so the delay cannot propagate or influence the scheduling.

Thus, the use of this MMDP approach allows to prevent disturbances in advance, and construct a robust resolution method.

## 5. CONCLUSION

In this work, crew pairing problem is modeled as a Markov Decision Processes (MDP). This approach aims to constitute a robust mechanism that can absorb disturbances. Thus, the risk of taking real-time optimization can be avoided as maximum as historical data is analyzed. The given priori solution is the set of pairings that can handle the best disturbances in flight operations. So, we do not take into account real-time optimization.

The experimentation on this approach by simulation of the related policy iteration algorithm gives the best feasible solution.

The objective behind this reflection is to give to controllers at the airport a robust priori solution instead of taking the risk of online schedule

modifications to handle uncertainty. Even if it takes more time to resolution, it can exclude online treatment of optimization.

On the other hand, the experiments using the tuned PSO by HMM was promising as a resolution method for a deterministic model of the gate, where, the resolution complexity has been reduced. Otherwise, HMM-QPSO can also be used as a perspective to solve the proposed MDP model of the crew pairing using a reinforcement learning approach.

As perspective, more computational methods can be integrated to compute effective transition probabilities and other input parameters for this proposed approach. Also, more real constraints, restrictions and real data have to be considered.

---



## CHAPTER VI:

# A REINFORCEMENT LEARNING MODEL OF THE MAINTENANCE ROUTING PROBLEM

---

In airline scheduling, a multitude of operational decision and planning problems need to be solved. We consider the problem of aircraft maintenance routing (AMR) as a crucial component in flight operation. Aircraft maintenance routing is of basic significance to the safe and efficient operations of an airline. However, the timely efficiency of the airline flight schedule is susceptible to various factors during daily operations. Air traffic often undergoes some random disruptions that expose maintenance routing to random flight delays, which have to be considered to ensure safe and operational flight schedule.

The temporal Performance of the airline flight schedule is predisposed to different factors during daily operations. Maintenance routing as an important critical component of the flight schedule might be the subject of random flight delays because of numerous factors as hard weather conditions, airspace congestion, extended ground holding, etc. Flight operations can be severely disturbed without considering such events.

A robust approach will help to prevent such unwanted situations of missed maintenance checks. A decision support system data based on dealing with the maintenance disturbances problem is developed targeting to assist policy makers in handling disturbances. A Markov Decision process model was selected in this thesis to remedy this problem and design the maintenance needs of an aircraft taking past data information into account. Maintenance actions are modeled with stochastic state transitions. This can offer the opportunity to solve the maintenance routing problem deliberating and handling flight disturbances through computational tests on real data of a Moroccan airline company.

## 1. AIRCRAFT MAINTENANCE PROBLEM

The maintenance of an aircraft and its elements is a necessary and rigorously controlled duty to guarantee the safety of an aircraft and its operations. Also, the aircraft maintenance is considered among the main direct operating costs for an airline and constitutes an important factor in providing a greater service quality. Maintenance, consequently, should be executed at the minimum possible cost, offer the best level of service and provide competitive delivery times without reducing quality and safety. To attain these objectives, commercial aviation maintenance is arranged in an organized and well-structured maintenance program of scheduled tasks.

Effective decision making of large maintenance systems is determined by several independent sources of information and is thereby complex. The current state situation of every device, daily, weekly and monthly plans of maintenance, the state profile of the machine, maintenance costs, etc. The system level control is the most practical to generate decisions during the maintenance once having envisioned the information about the aircraft, regarding departments and various other information as inputs. These inputs are matched against production preferences which are established by the company. In order to increase productivity, rise stability and responsiveness, the maintenance operations in larger and complex airline activities have to be facilitated by control, design and good management.

Delays and disruptions were a frequent concern during the achievement of maintenance checks, keeping a significant operational impact on the overall airline performance. It was also noticed that this issue was primarily due to the complexity of managing the resources and the incidence of unplanned maintenance tasks. It was also observed that the used approaches were not as successful as wanted. Therefore, it is recommended to solve the problem from a distinct and more effective perspective than all those previously applied. However, due to the large multitude of flights planned every day, that may easily attain thousands and thousands for a major airline.

To be able to considerably better realize the aircraft maintenance process and to describe the effect and the relevance of this study, this section provides the maintenance problem as contextualized, explaining the role of the airline and the issues it has been working on. The importance and significance of aircraft maintenance for airlines are due to the technical character of the subject, and the important rules that are outlined with the overall structure of aircraft maintenance. This section gives and identifies the problem and its context.

## 1.1. PROBLEMATIC

The most challenging issue in the AMR problem is that before an aircraft goes in into a heavy maintenance check, a complete plan is identified indicating all the scheduled maintenance tasks to execute and identifying the needed resources for achieving them. However, during the execution of the maintenance check, especially at the time of the inspection stage, damage and failures are identified that have to be fixed by programming further maintenance activities that are not regarded in the first plan. These unscheduled tasks stop the planning and charge of the whole maintenance service as they require additional resources and force adjustments to the initial plan, causing disruptions throughout the execution of the service, which for most cases leads to delays by the end of the maintenance check. In contrast, delays can also cause a disruption in the overall airline planning and therefore, a perturbation in the execution of scheduled maintenance checks. Delays can trigger a supplementary operational cost of maintenance services were not associated with only one company. When disturbances produce, it is difficult to realize the planned maintenance task. The controller at the airline has to establish another maintenance routing depending on airline policy, to lower the negative impact of these perturbations.

The large mass data existing for the airline regarding previously past-executed plans and operations have an important factor to examine and avoid that kind of maintenance routing failures. The target of our research is on enhancing the robustness of a planned schedule by redistributing these maintenance possibilities, while simply building limited adjustments to the originally-planned maintenances. Our objective is to minimize the overall likely number of maintenance failures. Operationally, this supplies a large chance of having the ability to disturb tails to guarantee maintenance feasibility without additional disruption to the planned maintenance checks.

## 1.2. AIRLINE PLANNING PROCESS OF MAINTENANCE

To start the planning process, the airline establishes its schedule of daily flights for a time period, such as a quarterly schedule. Offered the set of flights, the fleet assignment problem can be solved, assigning every single flight a certain aircraft type. Right after fleeting is done, the schedule could be decomposed by fleet type and for every single fleet type, the crew scheduling problem and maintenance routing problems are then resolved. This planning process is described in figure 52.



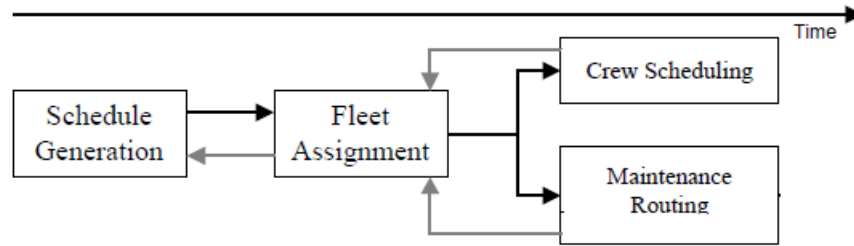


Figure 52 typical airline planning scheme

During maintenance routing, the lines of flight considered as flight legs are built. These legs are then utilized to create feasible aircraft rotations for maintenance checks.

### 1.3. MAINTENANCE PROBLEM DESCRIPTION

Ahead of solving the aircraft maintenance routing, an airline schedule is created. It is absolutely built as being a set of flight legs that the airline makes to fly. Then, it is about to identify allocated resources to the fly schedule properly. The airline needs to specify equipment types just like a Boeing 757, etc, and then each leg is assigned to an aircraft having constraints satisfied. This is known as the fleet assignment. The latter is supplied to the aircraft maintenance routing process so that each individual aircraft can be allocated particular routes between those provided for its special type, with the full satisfaction of maintenance constraints.

Aircraft maintenance must be planned, started performing and controlled as outlined by prescribed procedures and standards, achieving extremely specific and rigid requirements, seeking to maximize system performance at the minimum cost [270]. [271] clarifies that aircraft maintenance is organized as a systematic and scheduled program that is mutually authorized by the aeronautical authorities and manufacturers of aircraft and parts. The maintenance program definitely identifies how, and when, every specific scheduled maintenance task needs to be performed. The achievement of this program avoids damage, and deterioration of an aircraft and its components, conserving the typical levels of reliability and guaranteeing aircraft safety [272].

It was noticed that even though the usage of information systems for controlling resources and handling the process, once the volume of unscheduled maintenance tasks begins increasing above the level originally predicted, handling the service turns extremely difficult, since one change may impact the execution of different tasks or the availability of resources for several other activities. The whole maintenance process gets into a crucial alert, because of that immediate issues resolution can malfunction the problem even more. For this intent, this approach is provided in this chapter.

#### 1.4. MAINTENANCE TYPES

[273] mentions that a maintenance plan contains three primary components:

- Aircraft inspections which consist of repeated routine inspections, minor checks, and tests;
- Scheduled maintenance that includes systems servicing, replacing parts, routine overhauls, and particular inspections;
- Unscheduled maintenance, targeting to solve unpredicted failures, generally noticed from inspections.

Maintenance regulations and constraints require for example that all aircraft undergo maintenance after flying a certain number of hours. Every aircraft involves several types of aircraft maintenance as checks. The checks are known as A, B, C, and D that change in their frequency, scope, and timeframe. Among these types of checks, just type A are considered as routine maintenance that needs to be frequently done (every sixty-five flight hours) and inspects all the main systems. B checks are performed every 300 to 600 flight hours and involve a visual examination and lubrications. Level C and D maintenance checks are commonly classified as heavy maintenance and are usually carried out about once every one to four years because of the high associated workload. The C and D sorts of checks take more than 24 hours and are simply modeled by reducing the number of available aircraft. Therefore, A and B checks are integrated into maintenance routing model, and their maintenance constraints are incorporated. When a check is not performed inside the specified period, the aircraft is not allowed to fly. Hence the importance and criticality of satisfying those maintenance requirements.

Likewise, maintenance is mainly separated into two main strategies (as in figure 53):

- Preventive maintenance: this approach of maintenance is performed usually based on maintenance manuals in certain time periods described by the original aircraft producer. The preventive maintenance keeps the aircraft and helps prevent defects from arising.
- Corrective maintenance: When a deficiency is definitely discovered, the executed maintenance is called corrective maintenance. The purpose of corrective maintenance is to place the aircraft on condition again.

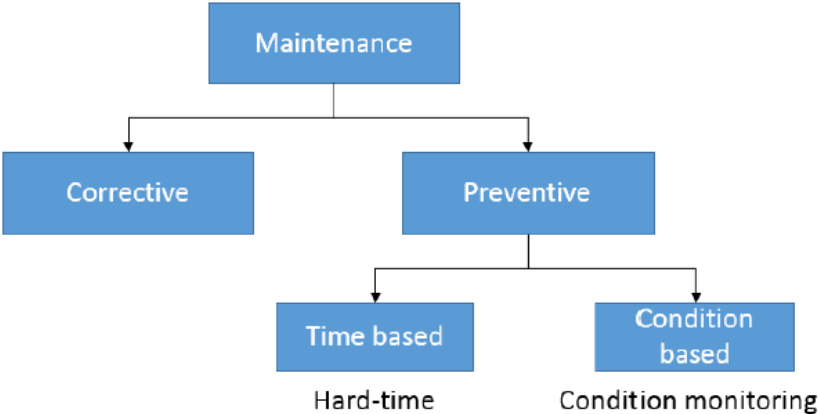


Figure 53 Maintenance types

The particular maintenance routing problem is to find a routing of the aircraft that will meet the short-term procedure maintenance prerequisites (Checks type A and B).

**1.5. RESOURCES MANAGEMENT IN MAINTENANCE**

Within the heavy maintenance, many resources are in continuous connection: aircraft, ground services, parts and materials, tools and equipment, technical details and workers. [274] recognizes operators, equipment, documentation, and tasks as the primary communicating components in aircraft maintenance systems and mention that these components communicate over time with each other and with an amount of exterior circumstances. [275] additionally discuss that aircraft maintenance is a complicated system, where staff performs diverse tasks with critical time limitations, small feedback and environmental and working conditions that are sometimes difficult. Figure 54 depicts the interactions and interdependence concerning the different resources included in main checks.



*Figure 54 Maintenance Resources management*

As a result of the large variety of distinct resources associated with the achievement of heavy maintenance and, moreover, due to the complicated interconnection among them, the planning, supply, and control of all resources should be properly handled. A deficiency in the supply of one resource can impact the control of the others and consequently impact the whole maintenance services. [276] states that maintenance planning is important to reduce aircraft downtime while keeping positive efficiency and inventory costs since it handles the execution of maintenance tasks, work synchronization, tools and the supply of needed parts..

## **1.6. MAINTENANCE TASKS**

Scheduled maintenance tasks (see Figure 55) are generally appointed accurately since they, and their preferences, are plainly identified in the maintenance program. Therefore, in the distinction of the short-term program, it is conceivable to forecast and plan the resources included and every single activity that should be performed. In this context, [277] clarifies that planned maintenance function is expected and regular, and consequently can be generated with a reasonable level of precision. However, additional highlights considering that their scheduling can call for calculating aircraft utilization and required maintenance periods much before the maintenance check that generally causes a sub-utilization of the aircraft and the various other resources.

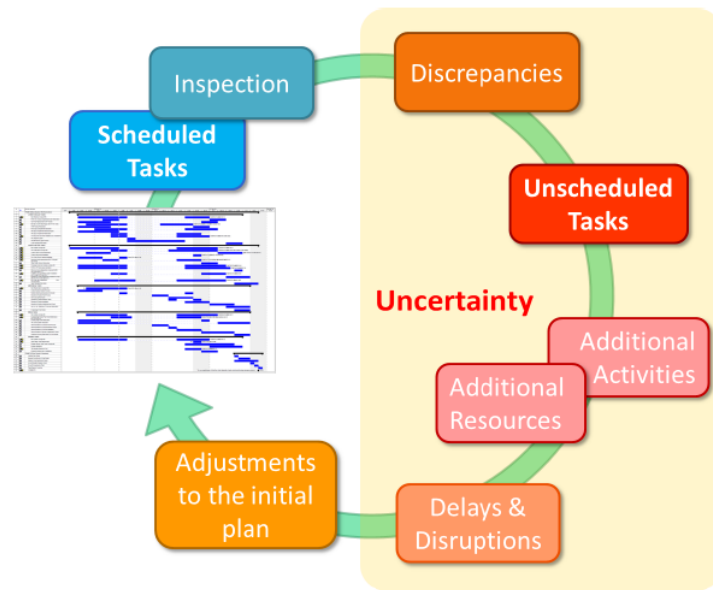


Figure 55 Maintenance tasks

In addition, inside a scheduled maintenance system, unscheduled and unexpected activities, typically called non-routine tasks, occur at the time of the operation of aircraft. As [277] claims, unpredicted behavior out of specification normally happens but is hard to predict. Also in [278], the authors stated that in fact about a half of most of the maintenance activities are unexpected and unscheduled occurrences.

The next section will describe the most relevant maintenance approaches from literature.

## 2. MAINTENANCE LITERATURE

The main objective of this section is to review the literature of the aviation industry, mainly focused on aircraft maintenance. Before addressing the different approaches from the literature, the most common maintenance model of aircraft maintenance routing is presented.

### 2.1 GENERAL MAINTENANCE ROUTING MODEL

Before giving our model approach and formulation, we provide first light on the most commonly used model of maintenance routing. We will base on this approach to describe our own method in the next section.

As mentioned before in the literature, there are two different ways to model the maintenance routing. The first is a network-based model, while the second type is the string based model. The approach described in this chapter is a string based model as introduced first time by [279] and adopted by many types of research as in [280]. The choice of this model is

for its convenience to integrate disturbance management. Mainly because delays propagate through the aircraft routes, it is hard to make use of network-based models to monitor delay propagation. Thus, a routing string based model is considerably more suitable. Such type of model, a formulation for robust aircraft maintenance routing with the aim to reduce overall estimated propagated delay, is provided in the next section. A string is a sequence of connected flight legs that start and finish at maintenance stations (probably distinct).

Let  $S$  be the set of feasible strings,  $F$  be the set of daily flight legs,  $F^+$  be the set of flight legs originating at a maintenance station, and  $F^-$  be the set of flight legs terminating at a maintenance station. We denote the set of ground variables (including the overnight or wraparound arcs to ensure that the flight schedule can repeat daily) as  $G$ , the set of strings ending with flight leg  $i$  as  $S_i^-$ , and the set of strings beginning with flight leg  $i$  as  $S_i^+$ . We have one binary decision variable  $x_s$  for each feasible string  $s$ . We have ground variables denoted by  $y_g$ , which are used to count the number of aircraft on the ground at maintenance stations. Let  $a_{is}$  equal 1 if flight leg  $i$  is in string  $s$ , and equal 0 otherwise. Ground variables  $y_{i,d}^+$  equal the number of aircraft on the ground before flight leg  $i$  departs, and ground variables  $y_{i,d}^-$  equal the number of aircraft on the ground after flight leg  $i$  departs; ground variables  $y_{i,a}^-$  equal the number of aircraft on the ground before flight leg  $i$  arrives, and ground variables  $y_{i,a}^+$  equal the number of aircraft on the ground after flight leg  $i$  arrives.  $r_s$  is the number of times string  $s$  crosses the count time, a point in time when aircraft are counted,  $p_g$  is the number of times ground arc  $g$  crosses the count time, and  $N$  is the number of planes available.

The String model is as follows:

$$\min \sum_{s \in S} c_s x_s \quad (80)$$

Subject to:

$$\sum_{s \in S} a_{is} x_s = 1 \quad \forall i \in F, \quad (81)$$

$$\sum_{s \in S_i^+} x_s - y_{i,a}^- + y_{i,d}^+ = 0 \quad \forall i \in F^+, \quad (82)$$

$$-\sum_{s \in S_i^-} x_s - y_{i,a}^- + y_{i,d}^+ = 0 \quad \forall i \in F^-, \quad (83)$$

$$\sum_{s \in S} r_s x_s + \sum_{g \in G} p_g y_g \leq N \quad (84)$$

$$y_g \geq 0 \quad \forall g \in G,$$

$$x_s \in \{0,1\} \quad \forall s \in S$$

The objective (80) concerns the minimization of the expected total full cost of the selected strings. Constraints (81) guarantees that each flight leg is in exactly one string as cover constraints. Constraints (82) and (83) ensure the flow balance constraints between the number of aircraft arriving at and departing from a specific location. Constraint (84) is considered as the counting constraint that ensure the total number of aircraft being used at the count time will not surpass the number of aircraft in the fleet. Constraints (83) and (84) make that the number of aircraft on the ground is non-negative and the number of aircraft allocated to a string to be 0 or 1, since variable  $y$  is a sum of binary  $x$  variables.

## 2.2 AIRCRAFT MAINTENANCE STUDIES

Generally solved following the fleet assignment, maintenance routing problem is to identify how an specific aircraft is maintained within the rotation of overall maintenance constructed as checks [281].

Two main types of maintenance routing models are found in the literature [282]. First called Flight-Based Formulation Models and the second is considered as String-Based Models.

As a flight-based model, [281] presents the aircraft maintenance routing problem using a mathematical programming formulation, and it is viewed as an asymmetric traveling salesman problem. In a simplified connection network, arcs symbolize possible connections, and nodes symbolize flight segments. The goal is to maximize the profit resulting from constructing specific connections, referred to as through value. The presented model can capture several maintenance considerations. It is solved with a Lagrangian relaxation procedure and provides near-optimal solutions.

In [283], the rotation of aircraft planning problem as a mathematical integer programming for some European airline. The objective function is defined to decrease the risk induced by the total delay, which is supplied by two factors: one by distinct flight connections and another by using the total path made by flight legs. the problem is solved using a Lagrangian relaxation approach.

Network flow models are offered in [284] to deal with the aircraft maintenance routing problem in case of single type maintenance. This model constructs a number of maintenance arcs without having overnight arcs. It begins at the end of every day at a station and finishes at the start of the exact same station timeline. The objective function is 0 since it only regards the maintenance routing problem just as a feasibility problem. The Solution is a set of arcs utilized in the optimal rotation recognized as a Euler tour. Therefore, a polynomial time algorithm is applied to determine the arcs sequence. Two model developments have also been supplied in [284]

to generate the profit or cost of the maintenance routing problem by forming certain operational arcs on the time-space network.

A collection of algorithmic solutions are listed in [285] and [286]. They considered a configuration where one-day trips are fixed. An algorithm of polynomial time is supplied for obtaining a three-day maintenance Euler tour once there is determined one in a flight network.

Also, [287] gives another formulation of the operational aircraft maintenance routing problem, that incorporates a set-partitioning where decision variables symbolize feasible aircraft routes. This formulation contains maintenance of resource availability constraints, and two sets of equivalent resource constraints are made in this model. To solve the problem, a branch-and-price algorithm is used with changed branch-on-follow-on rule.

The second type of models, String-Based Model was proposed in [279] to solve the aircraft maintenance routing problem for a single maintenance type. A group of connected flights determines a string, which match some conditions. This group needs to start and finish at maintenance stations having a maintenance check in the end. The group should meet maintenance feasibility and the flow balance. The minimization of the cost of the selected route strings is the objective function.

A model for dealing with the weekly schedule is given by [288] with two types of maintenance. Primarily, a set of one-day trips is built; then a time-space network is constructed on those trips. The model constitutes a flight segment where included by just a single trip side constraint for two types of maintenance.

Many researchers have targeted on resolving over than one optimization problem simultaneously. This produces better incomes and reduces the cost for the airlines. The aircraft maintenance routing problem has been also associated with different integrated scheduling problems.

In [289], A multi-commodity fleet assignment model is provided based on a set of constraints for maintenance routing. The two types of maintenance have been considered in their approach, namely the long and short maintenances.

Several works present the integrated planning for maintenance routing associated with crew pairing problems. Like in [290], the aircraft maintenance routing problem is seen as a feasibility problem where the crew cost is the majority in the total cost of the integrated problem.

A standard integrated model for the crew pairing and the maintenance routing problems was provided in [291]. This model ensures maintenance feasibility. The cost of maintenance routing is taken into consideration explicitly in this model.

An extended crew pairing model was offered in [292] to solve the combined maintenance routing problem and crew pairing.



Researchers are starting to give more consideration to potential delays and disruptions in the routing problems of airline transport. In a previous work [293], historical data as problem instances are used to tune and improve the parameters of resolution algorithms in airline transport. Other works have been applied Markov process in the same field of airline transport as a robust model to handle possible disturbances like in [152] and [294]. Another approach is given by [280] based String model tried to reduce delay propagation in the maintenance routing by intelligently routing aircraft. This problem was formulated as a mixed integer programming problem with stochastic generated inputs.

Several approaches and formulations for maintenance routing problems have been well addressed in the literature. Each one has its advantages and drawbacks. Our approach uses another different manner to model and to resolve this problem, based on historical data to extract more robust solutions. Our data-driven method relies on a constrained Markov process model. Theoretical backgrounds about our chosen methodology will be described in the next section.

### 3. MARKOV DECISION PROCESS MODEL

#### 3.1 FLIGHT DELAYS PROPAGATION

As mentioned before, a leg is a non-stop flight from an origin to a destination with specified departure and arrival instances. Moreover, a string is a sequence of connected flight legs that begins and ends at maintenance stations.

As shown in Figure 56 below, a delay in the flight leg  $f1$ , it will be  $f'1$  and causes departure delay for the next flight  $f2$  in the same string that is using the same aircraft (to be  $f'2$ ) if there is not enough minimum turn time (MTT) between these two flights.

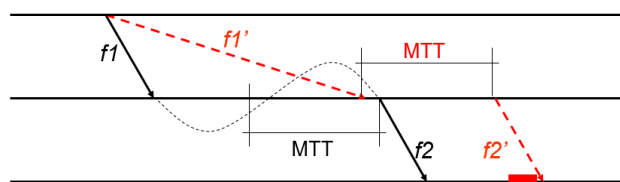


Figure 56 Delay propagation

Delay that arises when the aircraft used for a flight leg is delayed on its prior flight leg. This delay can be propagated to influence overall aircraft's routing.

Due to the fact that every aircraft routing is a sequence of flight legs flown by a single aircraft, an arrival delay can lead to a departure delay if there is not sufficient slack between two successive flight legs in the routing. This is called delay propagation; this event leads to many disruptions in air traffic, regarding crews, passengers as well as maintenance routing.

### 3.2 DELAYS DETECTION FROM DATA

Maintenance strings are composed of flight legs. Then, each propagated delay is a feature of maintenance routing. Therefore, while historical values for propagated delay can be calculated for every flight leg in current routings, no like values are readily available for routings that have not really been earlier realized. Nevertheless, just because independent arrival delay is not a function associated with routing, independent arrival delay can be computed for each flight leg monitoring actual routings of every single aircraft. Total propagated delays of flight legs in any routing will then be constructed from historical aircraft data, as described following in the algorithm 25. Previous work has given a similar notion of delay propagation in [280].

Let TAD be the total arrived delay, IAD is the independent arrive delay and PD in the propagated delay.

#### Algorithm 25: Delays extraction from DATA

```

Input: Strings  $S$ , flight legs  $F$ , Flights data history DATA( $TAD, Slack$ )
for each  $i, j \in F$  from DATA
     $PD_{ij} = \max(TAD_i - Slack_{ij}, 0)$ 
end for
for each  $i \in F$  from DATA
     $IAD_j = TAD_j - PD_{ij}$ 
end for
for each  $s \in S$ 
    for each  $i, j \in s$ 
        if ( $i$  is the first)
             $TAD = IAD$ 
        Else if
             $PD_{ij} = \max(TAD_i - Slack_{ij}, 0)$ 
             $TAD_j = IAD_j + PD_{ij}$ 
        end if
    end for
end for
Output : PD, TAD and  $IAD$  of each String  $s$ .

```

### 3.3 MDP MODEL DATA BASED

After determining the propagated delay for each flight legs in a string of legs, we can address our data-driven model based on those calculated values from data in the previous paragraph. So, the problem of uncertainty will be addressed in the context of Markov processes. The choice of MDP context is due to its flexible concept for stochastic and dynamic optimization problems. It differs from the model in paragraph 3 from the side that takes into account possible disruptions in flight operations caused by delays.

We define model components and variables at the light of the basic model given in paragraph 3. The use of MDPs requires defining its components. Five main features describe the constrained MDP formulation: state space, action space, state transition probabilities, global constraints and rewards.

We suppose as the assumption that the system behaves with well-known states and actions, and the use of available data is to compute the transition trajectories or also other rewards. The Objective is to find a solution as a stationary maintenance routing that succeeds to undergo uncertainties.

We consider as a stochastic process for the MDP, the flight arrivals as in the previous chapter. It takes its values from the state space.

A finite constrained Markov decision process is a 5-tuple  $\{X, U, P, R, D\}$ , as formal description, it is represented as fellow [295] :

- $X$  is a finite set of possible finite states; it matches the set of flight legs  $i$ .
- $U$  is a finite set of possible actions, each action  $u$  corresponds to assign a flight leg  $x \in X$  a to a string  $s \in S$ . If there is no best leg to assign, the action consists of retaining the leg assigned to the already affected leg  $x$ . Hence, the action set fits as set of possible strings to affect to state (legs).
- We denote  $U_x \subset U$  as set of possible strings to affect to a leg  $x$ .
- $R : X \times U \rightarrow \mathfrak{R}$  Is a real-valued reward function. It is the value of reward received when executing action  $u$  in state  $x$ . We note  $R(x, u)$ .
- Reward function is defined as follow:
  - o  $R(x, u) = -\rho \cdot c_u(x) - \rho' \cdot TAD_x(u)$  (85)
  - o Where:  $c_u(x)$  a cost of assigning a leg  $x$  to a string  $u$ .  $TAD_x(u)$  defines an associated cost to total arrived delay of flight leg  $x$  (as state) in a string  $u$  (as action). Its values are extracted from history in order to give a penalty to assignments leading to a total delay.  $\rho$  and  $\rho'$  are two weights associated to delay cost and affectation cost respectively
- $T : X \times U \rightarrow T$  is the transition function as a probability distribution of undergoing actions over the next states. It reflects the data-driven aspect of our problem. Hence, we define the transition probability from the previously computed delays data as: To each action and state of the environment, the function  $T(x, a, x')$  is defined as the probability  $p_{x,x'}(u)$  that the system is in state

$x \in X$  goes to state  $x' \in X$  when the agent chooses action  $u$  from actions space  $A$ . It is a probability of leg assignment changing from a string (action) to another.

$$T_{x,x'}(u) = \frac{PD_{x,x'}(u)}{PD(u)} \quad (86)$$

- It is a probability of having a propagated delay between two flight legs  $x$  and  $x'$ . So, a transition will force an assignment changing if a propagated delay is detected.
- The set  $D$  called a set of feasible constraints; it defines a cost conditioned constraint on policies. Constraints (80), (81) and (82) are defined as feasible constraints for the CMDP.
- We identify a policy  $\pi$  as a mapping from  $X$  to  $U$ , giving for every state  $s$  a corresponding action  $u = \pi(x)$  to be performed in state  $x$ .

As a constrained Markov decision process,  $\pi^*$  is the optimal policy for constraint costs  $D$ . The resolution consists of finding a policy that maximizes the total reward subject to constraint  $D$ . This model can be better than others from the literature like [279] in terms of incorporating propagated delays as stochastic measures. This will help to avoid and plan in advance a robust solution that can handle possible disturbances. Experimentation in the next sub-section will show how this model can be more robust based on past data.

### 3.4 REINFORCEMENT LEARNING MODEL BASED DATA

Reinforcement learning is determined by measuring the rewards provided by the Markov Decision Process and selecting actions depending on a particular policy. An episode indicates a running of the system where an agent uses actions from a primary state till it attains a final state. In this approach, the same formalism as in the previous paragraph is taken into consideration to build the reinforcement learning approach.

The first feature of machine learning is the fact that it can handle offline or online learning, as well as a combination between the two just like this approach. Two mechanisms are working simultaneously in this data-driven approach, an online model part for gathering data needed to build the model. The latter will be built in an offline learning part giving the corresponding robust resolution of the maintenance routing. The process is explained in Figure 57.

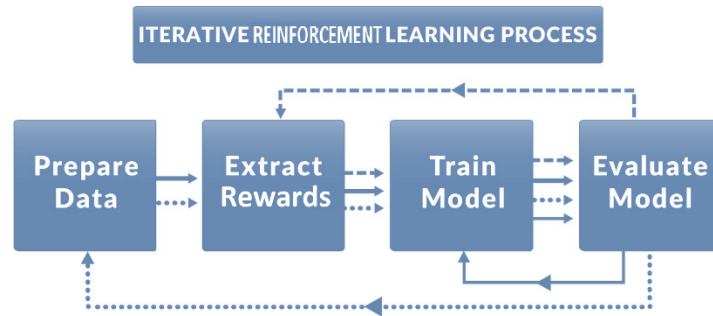


Figure 57 Reinforcement learning iterating from data.

When looking at the maintenance routing problem, we have a training dataset that increases over time. This dataset gradually continues to get too sizeable and therefore too costly to analyze or to store. The reinforcement learning model specifically matches our requirements in the maintenance problem, as the agent is the airline controller making changes to the maintenance schedule via actions and with those actions, it eventually varies, for enhanced or even worse, the maintenance issue, which in turn is the state of the environment. Eventually, the air traffic controller requires to maximize some long-term reward, or else minimize a cost function. Reinforcement learning can also be suitable for online learning [151].

Before building the corresponding reinforcement learning model, some measurements from data are made to extract the evaluation parameters causing disruptions. It is identified as delays in the flight schedule.

The resolution algorithm used is the HMM-QPSO of chapter 2.

Delays related Information extracted by the algorithm 25 will be incorporated to calculate the reward of actions to determine the best action with the minimum delay.

We used the HMM-QPSO of chapter 2, as a variant of the Q-learning algorithm. It is a model-free reinforcement learning algorithm. The HMM-QPSO algorithm needs to identify the encoded state, possible actions, and it has to calculate the reward between two succeeding states applying some reward function. Hence, for the maintenance routing problem, the formalism is the same as the MDP with the addition of delay reward in the reward function as:

$R(s, a, s')$  is the reward obtained when altering from one state  $s$  to a state  $s'$  by executing action  $a$ . It corresponds to reassigning a flight leg to a pairing. This includes negative costs and positive benefits of every reassignment.

In our case, if pairing  $p$  flies leg  $i$  between  $x$  and  $x'$ , an immediate reward  $r_i$  is generated. Its defined as the negative cost of propagated delay:

$$r_i = -PD_i(p) \quad (87)$$

Then, the total accumulated reward  $R$  for a single episode can trivially be expressed as the sum of all of the obtained rewards:

$$R = \sum_{0 \leq i \leq n} r_i \quad (88)$$

An episode constitutes a single run of the system in which an agent takes actions from an initial state until it reaches a terminal condition.

### 3.5 EXPERIMENTATION

We accomplish a computational study to examine the efficiency of the used methodology. We make use of real trial records data to conduct our experimentation.

This analysis is performed on a data set consisting of flight delay records of a Moroccan airline (RAM) provided by a database given by OpenFlights [296]. Records are related to flight legs grouped in maintenance routes considered in the model as strings. For example for the Moroccan airline company RAM, it has about 240 routes.

The data are used to calculate delays related to measures as in the algorithm 25 and other model parameters such as transition probabilities and rewards.

Table 26 shows a small sample of used flight legs for experimentation.

*Table 25 Flights data sample*

<b>From</b>	<b>To</b>	<b>Company</b>	<b>Distance</b>	<b>Duration</b>
RAK	TLS	RAM(AT)	975	02:27
RBA	CMN	RAM(AT)	67	00:38
RBA	ORY	RAM(AT)	1116	02:43
ROB	CMN	RAM(AT)	1882	04:15
ROB	FNA	RAM(AT)	254	01:00
RUH	CAI	RAM(AT)	1000	02:30
SIN	AUH	RAM(AT)	3655	07:48
SSG	CMN	RAM(AT)	2298	05:05
SSG	LBV	RAM(AT)	232	00:57
SVO	CMN	RAM(AT)	2635	05:46

All flights belong to a specific maintenance string. We suppose as an initial solution of our algorithm the solution without delay consideration (the

original routing). The resolution has been performed with Matlab based on MDP toolbox [297].

The solution consists of constructing a delay robust maintenance routing. Using the proposed methodology, we compute the propagated delay rate of the CMDP optimal policy compared to the initial routing (calculated by the original formulation). Table 27 shows how the new solution is more beneficial in terms of propagated delay.

*Table 26 Propagated delay diminution*

<b>Strings</b>	<b>Original PD</b>	<b>New PD</b>	<b>Delay decrease rate</b>
S1	341	219	35.7%
S2	369	306	17%
S3	203	134	33.9%
S4	316	200	36.7%
S5	187	64	65.7%

The new solution gives a clearly best solution in terms of delay propagation. The decrease rate varies in our solution from 17% to more than 65%, with a promising average rate of 35.7%. This leads to more robust maintenance routing planning.

## 4. CONCLUSIONS

Using a Markov decision process model based on historical data to resolve the maintenance routing problem, a Markov decision problem model has been established with input parameters based on flights data. Aimed to construct a model that can handle possible uncertainties in maintenance routing. This approach calculates from historical data the propagated delay, the source of disturbances in maintenance routing, which are given as input to the MDP model. This model constitutes a preliminary attempt to model the maintenance routing incorporating the past recorded data. This method looks advantageous in terms of taking into account previous executions of the routing solution to build more robust ones. Simulation of the MDP on some simple real data gives promising results in terms of reducing the propagated delays on the maintenance strings.

Otherwise, the used HMM-QPSO, as a variant of reinforcement learning, can be a model-free algorithm of maintenance routing. The HMM-QPSO algorithm can identify the encoded state, possible actions, and it calculates the reward between two succeeding states applying the defined reward

function instead of calculation transition probabilities as Dynamic programming resolution algorithms. This resolution method for the MDP gives more competitive results for the maintenance routing based on historical data.

Future research will attempt to apply the proposed approach to bigger data sets in order to evaluate its limits. Besides, this method can be extended to other problems in airline transport and establish integrated planning of maintenance routing with other air transport planning.



---



# Conclusion

---

Markov Model as a stochastic state space model involves random transitions between states where the probability of the jump is only dependent upon the current state, rather than any of the previous states. This simple formalism in the context of probabilistic machine learning has been a key to many variants used in this thesis in three different aspects: performance enhancement of the PSO algorithm based HMM, a new multi-agent Reinforcement learning resolution method and finally modeling and resolution in the field of airline transport.

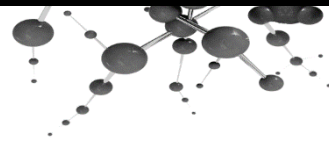
HMM has been used in this thesis to control and adapt PSO key parameters in both online and offline configuration techniques. This constitutes a machine learning technique for the control PSO search. It looks valuable from the view that HMM is a robust stochastic classification tool that takes into account past information about the population to control and adapt the algorithm. Our Proposed PSO variants based on the attached hidden Markov chain provides the best particles swarm control of parameters during the search. According to each swarm, acceleration coefficients and inertia weight are updated. Experimental results have established very competitive performances in comparison to several chosen PSO variants. We can deduce from the obtained results that associating a PSO to a probabilistic machine learning strategy enhances PSO performances significantly.

In terms of reinforcement learning, Markov decision processes constitute a promising formulation of sequential decision problems under uncertainty. Its classical resolution methods converge slower to optimal solutions in large state space problems than small state space problems. A new resolution method has been proposed based on the cooperation between particles of the enhanced PSO. Learners modeled as particles cooperate to accelerate the learning process using a sharing of knowledge procedures. The enhanced PSO is formed in this method by independent Q-learners as particles that share their personal Q-values by following a sharing strategy after performing

some episodes learning independently. The given results of experimentations were promising and give better performances than the classical resolution methods.

To make use of the obtained results in practice, three key problems of airline transport have been chosen for resolution by these non-conventional resolution methods. A Markov decision process is applied for modeling. Investigated results reflect MDP as a promising new mathematical framework for decision making. With the advances in resolution strategies that enhanced their solution performances as those presented in this thesis, MDPs can be widely used for the solving of real-world problems of both planning and control as they are surprisingly capable of capturing the essence of purposeful activity in a variety of situations.

Future research and perspectives should attempt to use parallel computing to improve time performances during simulation, which can handle the consumed additional CPU time computing cost of machine learning. It can be applied to both approaches of PSO enhancement and Reinforcement resolution methods. In terms of modeling, future axes will attempt to incorporate more real constraints of airline transport presented models, also, generalizing those methods to other problems in airline transport, which can be more advantageous.



---

# References

---

- [1] J. Sun, C.-H. Lai, and X.-J. Wu, *Particle swarm optimisation: classical and quantum perspectives*. Crc Press, 2016.
- [2] J. Kennedy and R. . Eberhart, "Particle swarm optimization," *Proceedings of IEEE international conference neural networks*, vol. IEEE, pp. 1942–8, 1995.
- [3] J. Schaeffer, P. Lu, D. Szafron, and R. Lake, "A re-examination of brute-force search," in *Proceedings of the AAAI Fall Symposium on Games: Planning and Learning*, 1993, pp. 51–58.
- [4] N. Durand and J.-M. Alliot, "Genetic crossover operator for partially separable functions," in *GP 1998, 3rd annual conference on Genetic Programming*, 1998, pp. pp-xxxx.
- [5] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.
- [6] C.-W. Chou, J.-H. Lin, C.-H. Yang, H.-L. Tsai, and Y.-H. Ou, "Constructing a markov chain on particle swarm optimizer," in *Innovations in Bio-Inspired Computing and Applications (IBICA), 2012 Third International Conference on*. IEEE, 2012, pp. 13–18.
- [7] C.-W. Chou, J.-H. Lin, and R. Jeng, "Markov chain and adaptive parameter selection on particle swarm optimizer," *International Journal on Soft Computing*, vol. 4, no. 2, p. 1, 2013.
- [8] J. Clausen, A. Larsen, J. Larsen, and N. J. Rezanova, "Disruption management in the airline industry-concepts, models and methods," *Computers & Operations Research*, vol. 37, no. 5, pp. 809–821, 2010.
- [9] R. Bouffanais, *Design and control of swarm dynamics*. Springer, 2016.
- [10] D. J. Sumpter, *Collective animal behavior*. Princeton University Press, 2010.
- [11] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *ACM SIGGRAPH computer graphics*, vol. 21, no. 4. ACM, 1987, pp. 25–34.
- [12] S. Granville, "Optimal reactive dispatch through interior point methods," *IEEE Transactions on power systems*, vol. 9, no. 1, pp. 136–146, 1994.
- [13] A. E. Hassanien and E. Emary, *Swarm intelligence: principles, advances, and applications*. CRC Press, 2016.
- [14] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

- [15] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm intelligence*. Elsevier, 2001.
- [16] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Transactions on evolutionary computation*, vol. 8, no. 3, pp. 211–224, 2004.
- [17] S. Sun, G. Ye, Y. Liang, Y. Liu, and Q. Pan, "Dynamic population size based particle swarm optimization," in *Advances in Computation and Intelligence*, ser. Lecture Notes in Computer Science, L. Kang, Y. Liu, and S. Zeng, Eds. Springer Berlin Heidelberg, 2007, vol. 4683, pp. 382–392. [Online]. Available: [http://dx.doi.org/10.1007/978-3-540-74581-5\\_42](http://dx.doi.org/10.1007/978-3-540-74581-5_42)
- [18] O. Aoun, M. Sarhani, and A. E. Afia, "Particle swarm optimisation with population size and acceleration coefficients adaptation using hidden markov model state classification," *International Journal of Metaheuristics*, vol. 7, no. 1, pp. 1–29, 2018.
- [19] F. Bergh and A. P. Engelbrecht, "Effects of swarm size on cooperative particle swarm optimisers," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2001, pp. 892–899.
- [20] J. Fulcher, "Computational intelligence: an introduction," in *Computational intelligence: a compendium*. Springer, 2008, pp. 3–78.
- [21] F. Van Den Bergh *et al.*, "An analysis of particle swarm optimizers," Ph.D. dissertation, University of Pretoria South Africa, 2001.
- [22] F. Van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information sciences*, vol. 176, no. 8, pp. 937–971, 2006.
- [23] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 2. IEEE, 2002, pp. 1671–1676.
- [24] J. KENNEDY, S. WORLDS, and M. MINDS, "Effects of neighborhood topology on particle swarm performance: proceedings of the congress on evolutionary computation," *S*, vol. 1, pp. 1931–1938.
- [25] A. Ismail and A. P. Engelbrecht, "Global optimization algorithms for training product unit neural networks," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 1. IEEE, 2000, pp. 132–137.
- [26] R. Brits, A. P. Engelbrecht, and F. Van den Bergh, "A niching particle swarm optimizer," in *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, vol. 2. Singapore: Orchid Country Club, 2002, pp. 692–696.
- [27] F. van den Bergh and A. P. Engelbrecht, "A new locally convergent particle swarm optimiser," in *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, vol. 3. IEEE, 2002, pp. 6–pp.
- [28] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *International conference on evolutionary programming*. Springer, 1998, pp. 591–600.
- [29] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information processing letters*, vol. 85, no. 6, pp. 317–325, 2003.
- [30] F. van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.
- [31] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review," 2017.
- [32] K. Premalatha and A. Natarajan, "Hybrid pso and ga for global maximization," *Int. J. Open Problems Compt. Math*, vol. 2, no. 4, pp. 597–608, 2009.
- [33] J. van den Akker, C. Kemenade, and J. Kok, "Evolutionary 3d-air traffic flow management," 1998.
- [34] P. Siarry, *Metaheuristics*. Springer, 2016.

- [35] K. Suresh, S. Ghosh, D. Kundu, A. Sen, S. Das, and A. Abraham, "Inertia-adaptive particle swarm optimizer for improved global search," in *2008 Eighth International Conference on Intelligent Systems Design and Applications*, vol. 2. IEEE, 2008, pp. 253–258.
- [36] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1. IEEE, 2001, pp. 101–106.
- [37] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on evolutionary computation*, vol. 3, no. 2, pp. 124–141, 1999.
- [38] A. P. Engelbrecht, *Fundamentals of computational swarm intelligence*. John Wiley & Sons, 2006.
- [39] G. Karafotias, M. Hoogendoorn, and E. algorithm), "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, pp. 167 – 187, 2015.
- [40] A. Carlisle and G. Dozier, "An off-the-shelf pso," in *in: Proceeding of Workshop on Particle Swarm Optimization*, Indianapolis, USA, January 2001 2001, pp. 1–6.
- [41] L. Zhang, Y. Tang, C. Hua, and X. Guan, "A new particle swarm optimization algorithm with adaptive inertia weight based on bayesian techniques," *Applied Soft Computing*, vol. 28, p. 138-149, 2015.
- [42] D. Chen and C. Zhao, "Particle swarm optimization with adaptive population size and its application," *Applied Soft Computing*, vol. 9, no. 1, pp. 39 – 48, 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1568494608000318>
- [43] M. Clerc, *Particle swarm optimization*, I. scientific and technical encyclopaedia, Eds. Hoboken: Wiley, 2006.
- [44] C. Lei, "The study on dynamic population size improvements for classical particle swarm optimization," in *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, vol. 1, Dec. 2011, pp. 430–433.
- [45] B. Soudan and M. Saad, "An evolutionary dynamic population size pso implementation," in *International Conference on Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd*, April 2008, pp. 1–5.
- [46] N. Lynn and P. N. Suganthan, "Distance based locally informed particle swarm optimizer with dynamic population size," in *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems-Volume 2*. Springer, 2015, pp. 577–587.
- [47] W.-F. Leong and G. G. Yen, "Pso-based multiobjective optimization with dynamic population size and adaptive local archives," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 5, pp. 1270–1293, 2008.
- [48] F. Ma and X.-B. Chen, "Application of varying population size particle swarm optimization algorithm to agc of power systems," in *The Sixth World Congress on Intelligent Control and Automation, 2006. WCICA 2006.*, vol. 1, 2006, pp. 3310–3314.
- [49] M. de Oca, T. Stutzle, K. Van den Eenden, and M. Dorigo, "Incremental social learning in particle swarms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 2, pp. 368–384, April 2011.
- [50] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999, pp. 1931–1938.
- [51] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 3, pp. 281–295, 2006.
- [52] X. Yu and X. Zhang, "Enhanced comprehensive learning particle swarm optimization," *Applied Mathematics and Computation*, vol. 242, pp. 265–276, 2014.

- [53] Z. Zhan, J. Zhang, Y. Li, and Y. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, pp. 832–847, 2011.
- [54] Y. Tang, Z. Wang, and J. an Fang, "Feedback learning particle swarm optimization," *Applied Soft Computing*, vol. 11, pp. 4713–4725, 2011.
- [55] H. Huang, H. Qin, Z. Hao, and A. Lim, "Example-based learning particle swarm optimization for continuous optimization," *Information Sciences*, vol. 128, p. 125-138, 2012.
- [56] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE transactions on evolutionary computation*, vol. 8, no. 3, pp. 204–210, 2004.
- [57] C. Yang, W. Gao, N. Liu, and C. Song, "Low-discrepancy sequence initialized particle swarm optimization algorithm with high-order nonlinear time-varying inertia weight," *Applied Soft Computing*, no. 0, pp. –, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1568494615000058>
- [58] G. Ardizzon, G. Cavazzini, and G. Pavesi, "Adaptive acceleration coefficients for a new search diversification strategy in particle swarm optimization algorithms," *Information Sciences*, vol. 299, pp. 337–378, 2015.
- [59] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceeding of the IEEE World Congress of Computational Intelligence*, pp. 69–73, 1997.
- [60] Z.-H. Zhan, J. Zhang, Y. Li, and H.-H. Chung, "Adaptive Particle Swarm Optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.
- [61] R. Perez and K. Behdinan, "Particle swarm approach for structural design optimization," *Computers and Structures*, vol. 85, pp. 1579–88, 2007.
- [62] X. Yang, J. Yuan, J. Yuan, and H. Mao, "A modified particle swarm optimizer with dynamic adaptation," *Applied Mathematics and Computation*, vol. 189, no. 2, pp. 1205–1213, 2007.
- [63] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," *Computers & Operations Research*, vol. 33, no. 3, pp. 859–871, 2006.
- [64] T. Ting, Y. Shi, S. Cheng, and S. Lee, "Exponential inertia weight for particle swarm optimization," in *International Conference in Swarm Intelligence*. Springer, 2012, pp. 83–90.
- [65] J. Ding, J. Liu, K. R. Chowdhury, W. Zhang, Q. Hu, and J. Lei, "A particle swarm optimization using local stochastic search and enhancing diversity for continuous optimization," *Neurocomputing*, vol. 137, pp. 261–267, 2014.
- [66] K. Kentzoglanakis and M. Poole, "Particle swarm optimization with an oscillating inertia weight," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 1749–1750.
- [67] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1. IEEE, 2000, pp. 84–88.
- [68] A. O. Adewumi and M. A. Arasonwan, "On the performance of particle swarm optimisation with (out) some control parameters for global optimisation," *International Journal of Bio-Inspired Computation*, vol. 8, no. 1, pp. 14–32, 2016.
- [69] M. A. M. De Oca, T. Stutzle, M. Birattari, and M. Dorigo, "Frankenstein's pso: a composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.
- [70] M. Taherkhani and R. Safabakhsh, "A novel stability-based adaptive inertia weight for particle swarm optimization," *Applied Soft Computing*, vol. 38, pp. 281–295, 2016.
- [71] A. Khare and S. Rangnekar, "A review of particle swarm optimization and its applications in solar photovoltaic system," *Applied Soft Computing*, vol. 13, pp. 2997–3006, 2013.

- [72] S. M. Kamalapur and V. H. Patil, "Impact of acceleration coefficient strategies with random neighborhood topology in particle swarm optimization," *International Journal of Emerging Technologies in Computational and Applied Sciences*, vol. 3, no. 1, pp. 37–42, 2012.
- [73] P. Subbaraj, R. Rengaraj, S. Salivahanan, and T. Senthilkumar, "Parallel particle swarm optimization with modified stochastic acceleration factors for solving large scale economic dispatch problem," *International Journal of Electrical Power & Energy Systems*, vol. 32, no. 9, pp. 1014 – 1023, 2010. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0142061510000566>
- [74] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on evolutionary computation*, vol. 8, no. 3, pp. 240–255, 2004.
- [75] M. Epitropakis, V. Plagianakos, and M. Vrahatis, "Evolving cognitive and social experience in particle swarm optimization through differential evolution: A hybrid approach," *Information Sciences*, vol. 216, pp. 50–92, 2012.
- [76] S. S. Jadon, H. Sharma, J. C. Bansal, and R. Tiwari, "Self adaptive acceleration factor in particle swarm optimization," in *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications*. Springer India, 2013, vol. 1, pp. 325–340.
- [77] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 3, pp. 627–646, 2012.
- [78] Y. Hamadi, "Autonomous search," in *Combinatorial Search: From Algorithms to Systems*. Springer, 2013, pp. 99–122.
- [79] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.
- [80] M. Birattari and J. Kacprzyk, *Tuning metaheuristics: a machine learning perspective*. Springer, 2009, vol. 197.
- [81] A. Eiben, Z. Michalewicz, M. Schoenauer, and J. Smith, "Parameter control in evolutionary algorithms," in *Studies in Computational Intelligence*. Springer Verlag, 2007, vol. 54, no. 54, pp. 19 – 46.
- [82] H. H. Hoos, "Automated algorithm configuration and parameter tuning," in *Autonomous search*. Springer, 2012, pp. 37–71.
- [83] S. L. Epstein and S. Petrovic, "Learning a mixture of search heuristics," in *Autonomous Search*. Springer, 2012, pp. 97–127.
- [84] X.-S. Yang, S. Deb, M. Loomes, and M. Karamanoglu, "A framework for self-tuning optimization algorithm," *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 2051–2057, 2013.
- [85] J. Swan, J. Woodward, E. Ozcan, G. Kendall, and E. Burke, "Searching the hyper-heuristic design space," *Cognitive Computation*, vol. 6, no. 1, pp. 66–73, 2014. [Online]. Available: <http://dx.doi.org/10.1007/s12559-013-9201-8>
- [86] B. Crawford, R. Soto, E. Monfroy, W. Palma, C. Castro, and F. Paredes, "Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization," *Expert Systems with Applications*, vol. 40, pp. 1690 – 1695, 2013.
- [87] M. Birattari, *Tuning Metaheuristics: A Machine Learning Perspective*, ser. Studies in Computational Intelligence, J. Kacprzyk, Ed. Springer, 2006, vol. 197.
- [88] E. Yeguas, M. V. Luzón, R. Pavón, R. Laza, G. Arroyo, and F. Dáz, "Automatic parameter tuning for evolutionary algorithms using a bayesian case-based reasoning system," *Applied Soft Computing*, vol. 18, pp. 185–195, 2014.

- [89] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer Science & Business Media, 2003.
- [90] R. Durbin, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, ser. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, 1998. [Online]. Available: <https://books.google.co.ma/books?id=R5P2GIJvigQC>
- [91] Y.-j. Gong and J. Zhang, "Small-world particle swarm optimization with topology adaptation," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '13. New York, NY, USA: ACM, 2013, pp. 25–32. [Online]. Available: <http://doi.acm.org/10.1145/2463372.2463381>
- [92] S. Sahebi, F. Oroumchian, and R. Khosravi, "Applying and comparing hidden markov model and fuzzy clustering algorithms to web usage data for recommender systems," in *ADIS European Conference on Data Mining*, 2008.
- [93] N. Di Mauro, F. Esposito, S. Ferilli, and T. Basile, "Avoiding order effects in incremental learning," in *AI\*IA 2005: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, S. Bandini and S. Manzoni, Eds. Springer Berlin Heidelberg, 2005, vol. 3673, pp. 110–121. [Online]. Available: [http://dx.doi.org/10.1007/11558590\\_12](http://dx.doi.org/10.1007/11558590_12)
- [94] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 31, no. 4, pp. 497–508, Nov. 2001. [Online]. Available: <http://dx.doi.org/10.1109/5326.983933>
- [95] O. Cappé, "Online em algorithm for hidden markov models," *Journal of Computational and Graphical Statistics*, vol. 20, no. 3, 2011.
- [96] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, pp. 164–171, 1970.
- [97] O. Aoun, M. Sarhani, and A. El Afia, "Hidden markov model classifier for the adaptive particle swarm optimization," in *The XI Metaheuristics International Conference (MIC)*, Agadir, Morocco, 7–10 June 2015 2015.
- [98] R. Sedgewick, "Implementing quicksort programs," *Commun. ACM*, vol. 21, no. 10, pp. 847–857, Oct. 1978. [Online]. Available: <http://doi.acm.org/10.1145/359619.359631>
- [99] L.-l. LIU and X.-b. GAO, "An adaptive simulation of bacterial foraging algorithm," *Basic Sciences Journal of Textile Universities*, vol. 4, p. 022, 2012.
- [100] W. Jiang, Y. Zhang, and R. Wang, "Comparative study on several pso algorithms," in *Control and Decision Conference (2014 CCDC), The 26th Chinese*, May 2014, pp. 1117–1119.
- [101] Z. Wu, "Optimization of distribution route selection based on particle swarm algorithm," *International Journal of Simulation Modelling (IJSIMM)*, vol. 13, no. 2, 2014.
- [102] G. Zeng and Y. Jiang, "A modified pso algorithm with line search," in *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, Dec 2010, pp. 1–4.
- [103] S. W. C. H. G. Wang, "Dream effected particle swarm optimization algorithm," *JOURNAL OF INFORMATION & COMPUTATIONAL SCIENCE*, vol. 11, no. 15, p. 5631, 2014. [Online]. Available: [http://manu35.magtech.com.cn/Jwk\\_ics/EN/abstract/article\\_2638.shtml](http://manu35.magtech.com.cn/Jwk_ics/EN/abstract/article_2638.shtml)
- [104] H. Jianxiu and Z. Jianchao, "A two-order particle swarm optimization model [j]," *Journal of Computer Research and Development*, vol. 11, p. 004, 2007.
- [105] G. E. Box, J. S. Hunter, and W. G. Hunter, *Statistics for experimenters: design, innovation, and discovery*. Wiley-Interscience New York, 2005, vol. 2.



- [106] J. Derrac, S. Garcá, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.
- [107] A. P. Engelbrecht, "Heterogeneous particle swarm optimization," in *International Conference on Swarm Intelligence*. Springer, 2010, pp. 191–202.
- [108] T. Blackwell, J. Branke *et al.*, "Multi-swarm optimization in dynamic environments," in *EvoWorkshops*, vol. 3005. Springer, 2004, pp. 489–500.
- [109] S. Mirjalili, A. Lewis, and A. S. Sadiq, "Autonomous particles groups for particle swarm optimization," *Arabian Journal for Science and Engineering*, vol. 39, no. 6, pp. 4683–4697, 2014.
- [110] Y. Shi, H. Liu, L. Gao, and G. Zhang, "Cellular particle swarm optimization," *Inf. Sci.*, vol. 181, no. 20, pp. 4460–4493, Oct. 2011. [Online]. Available: <http://dx.doi.org/10.1016/j.ins.2010.05.025>
- [111] J. Zhang and X. Ding, "A multi-swarm self-adaptive and cooperative particle swarm optimization," *Engineering applications of artificial intelligence*, vol. 24, no. 6, pp. 958–967, 2011.
- [112] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Transactions on Evolutionary Computation*, 2010.
- [113] M. A. M. De Oca, T. Stützle, K. Van den Enden, and M. Dorigo, "Incremental social learning in particle swarms," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 2, pp. 368–384, 2011.
- [114] M. Aydin, "Coordinating metaheuristic agents with swarm intelligence," *Journal of Intelligent Manufacturing*, vol. 23, no. 4, pp. 991–999, 2012. [Online]. Available: <http://dx.doi.org/10.1007/s10845-010-0435-y>
- [115] D. M. Diaz, "Agent-based configuration of (metaheuristic) algorithms," Ph.D. dissertation, Humboldt University of Berlin, 2005.
- [116] B. Bengfort, P. Y. Kim, K. Harrison, and J. A. Reggia, "Evolutionary design of self-organizing particle systems for collective problem solving," in *Swarm Intelligence (SIS), 2014 IEEE Symposium on*. IEEE, 2014, pp. 1–8.
- [117] J. Li and X. Xiao, "Multi-swarm and multi-best particle swarm optimization algorithm," in *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*. IEEE, 2008, pp. 6281–6286.
- [118] Y. Liu, Z. Qin, Z. Shi, and J. Lu, "Center particle swarm optimization," *Neurocomputing*, vol. 70, no. 4–6, pp. 672–679, 2007.
- [119] B. Niu, Y. Zhu, X. He, and H. Wu, "Mcpso: A multi-swarm cooperative particle swarm optimizer," *Applied Mathematics and computation*, vol. 185, no. 2, pp. 1050–1062, 2007.
- [120] G. Beni, "From swarm intelligence to swarm robotics," in *Swarm robotics*. Springer, 2004, pp. 1–9.
- [121] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.
- [122] E. Vidal, F. Thollard, C. De La Higuera, F. Casacuberta, and R. C. Carrasco, "Probabilistic finite-state machines-part ii," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 7, pp. 1026–1039, 2005.
- [123] R. J. Elliott, L. Aggoun, and J. B. Moore, *Hidden Markov models: estimation and control*. Springer Science & Business Media, 2008, vol. 29.
- [124] N. J. Cheung, X.-M. Ding, and H.-B. Shen, "Optifel: A convergent heterogeneous particle swarm optimization algorithm for takagi-sugeno fuzzy modeling," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 4, pp. 919–933, 2014.

- [125] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.
- [126] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Autonomous agents and multi-agent systems*, vol. 1, no. 1, pp. 7–38, 1998.
- [127] G. Weiss, *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.
- [128] S. A. Rahal, D. Mokeddem, N. Bensaid, J. Bigus, J. Bigus, H. Bjurn, L. Champciaux, M. Cote, K. Decker, V. Lesser *et al.*, "Intelligent agents: Theory and practice, the knowledge engineering review." *Information Technology Journal*, vol. 6, no. 1, pp. pp-456, 1998.
- [129] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette, "Evolutionary algorithms for reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 11, pp. 241–276, 1999.
- [130] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Organisational abstractions for the analysis and design of multi-agent systems," in *International Workshop on Agent-Oriented Software Engineering*. Springer, 2000, pp. 235–251.
- [131] P. Stone and M. Veloso, "Layered learning," in *European Conference on Machine Learning*. Springer, 2000, pp. 369–381.
- [132] P. Mathieu, Y. Secq *et al.*, "Environment updating and agent scheduling policies in agent-based simulators." in *ICAART (2)*, 2012, pp. 170–175.
- [133] A. Kabysh, V. Golovko, and A. Lipnickas, "Influence learning for multi-agent system based on reinforcement learning," *International Journal of Computing*, vol. 11, no. 1, pp. 39–44, 2014.
- [134] P. J. Hoen, K. Tuyls, L. Panait, S. Luke, and J. A. La Poutre, "An overview of cooperative and competitive multiagent learning," in *Proceedings of the First international conference on Learning and Adaption in Multi-Agent Systems*. Springer-Verlag, 2005, pp. 1–46.
- [135] B.-N. Wang, Y. Gao, Z.-Q. Chen, J.-Y. Xie, and S.-F. Chen, "A two-layered multi-agent reinforcement learning model and algorithm," *Journal of Network and Computer Applications*, vol. 30, no. 4, pp. 1366–1376, 2007.
- [136] Y. Shoham, R. Powers, T. Grenager *et al.*, "If multi-agent learning is the answer, what is the question?" *Artificial Intelligence*, vol. 171, no. 7, pp. 365–377, 2007.
- [137] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.
- [138] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.
- [139] T. Balch, "Behavioral diversity in learning robot teams," Georgia Institute of Technology, Tech. Rep., 1998.
- [140] E. Bonabeau, D. d. R. D. F. Marco, M. Dorigo, G. Théraulaz, G. Theraulaz *et al.*, *Swarm intelligence: from natural to artificial systems*. Oxford university press, 1999, no. 1.
- [141] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.
- [142] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.
- [143] D. White, "Finite horizon markov decision processes with uncertain terminal payoffs," *Operations research*, vol. 43, no. 5, pp. 862–869, 1995.
- [144] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, ser. Wiley Series in Probability and Statistics. Wiley, 1994. [Online]. Available: <https://books.google.co.ma/books?id=tsiiQgAACAAJ>
- [145] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine learning*, vol. 13, no. 1, pp. 103–130, 1993.

- [146] D. Bertsekas, "Distributed dynamic programming," *IEEE transactions on Automatic Control*, vol. 27, no. 3, pp. 610–616, 1982.
- [147] D. Wingate and K. D. Seppi, "Efficient value iteration using partitioned models." in *ICMLA*. Citeseer, 2003, pp. 53–59.
- [148] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.
- [149] M. P. Wellman, K. Larson, M. Ford, and P. R. Wurman, "Path planning under time-dependent uncertainty," in *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1995, pp. 532–539.
- [150] J. A. Boyan and M. L. Littman, "Exact solutions to time-dependent mdps," in *in Advances in Neural Information Processing Systems*. MIT Press, 2000, pp. 1026–1032.
- [151] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *IN PROCEEDINGS OF THE ELEVENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING*. Morgan Kaufmann, 1994, pp. 157–163.
- [152] O. Aoun and A. El Afia, "A robust crew pairing based on multi-agent markov decision processes," in *Complex Systems (WCCS), 2014 Second World Conference on*, Nov 2014, pp. 762–768.
- [153] J. v. d. Wal, "Stochastic dynamic programming : successive approximations and nearly optimal strategies for markov decision processes and markov games / j. van der wal," 1981, includes indexes.
- [154] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *In Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK96)*, 1996, pp. 195–210.
- [155] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, Nov. 2005.
- [156] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Mathematics of operations research*, vol. 27, no. 4, pp. 819–840, 2002.
- [157] S. Proper and P. Tadepalli, "Solving multiagent assignment markov decision processes," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '09. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 681–688. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1558013.1558107>
- [158] S. Ermon, C. Gomes, A. Sabharwal, and B. Selman, "Taming the curse of dimensionality: Discrete integration by hashing and optimization," in *International Conference on Machine Learning*, 2013, pp. 334–342.
- [159] M. Mahmud and S. Ramamoorthy, "Learning in non-stationary mdps as transfer learning," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1259–1260.
- [160] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *Journal of machine learning research*, vol. 4, no. Nov, pp. 1039–1069, 2003.
- [161] C. Daoui, M. Abbad, and M. Tkiouat, "Exact decomposition approaches for markov decision processes: A survey," *Advances in Operations Research*, vol. 2010, 2010.
- [162] A. G. Barto and T. G. Dietterich, "Reinforcement learning and its relationship to supervised learning," *Handbook of learning and approximate dynamic programming*. John Wiley and Sons, Inc, 2004.
- [163] J. Kober and J. Peters, "Reinforcement learning in robotics: A survey," in *Reinforcement Learning*. Springer, 2012, pp. 579–610.

- [164] K. Rawlik, M. Toussaint, and S. Vijayakumar, "Path integral control by reproducing kernel hilbert space embedding," in *IJCAI*, 2013, pp. 1628–1634.
- [165] W. V. de Abreu, A. C. Gonçalves, and A. S. Martinez, "Analytical solution for the doppler broadening function using the kaniadakis distribution," *Annals of Nuclear Energy*, vol. 126, pp. 262–268, 2019.
- [166] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1-2, pp. 99–134, May 1998.
- [167] J. Andrianus and A. Kurniawan, "Sejarah, teori dasar dan penerapan reinforcement learning: Sebuah tinjauan pustaka," *Jurnal Telematika*, vol. 12, no. 2, pp. 113–118, 2018.
- [168] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [169] J. Brunnström and K. Kaminski, "Exploring deep reinforcement learning algorithms for homogeneous multi-agent systems," 2018.
- [170] M. Fang, F. C. Groen, H. Li, and J. Zhang, "Collaborative multi-agent reinforcement learning based on a novel coordination tree frame with dynamic partition," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 191–198, 2014.
- [171] K. Tumer and A. Agogino, "Distributed agent-based air traffic flow management," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 2007, p. 255.
- [172] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews*, 38 (2), 2008, 2008.
- [173] H. Cuayáhuitl, I. Kruijff-Korbayová, and N. Dethlefs, "Nonstrict hierarchical reinforcement learning for interactive systems and robots," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 4, no. 3, p. 15, 2014.
- [174] R. Kraemer, G. Whiteman, and B. Banerjee, "Conflict and astroturfing in niyamgiri: The importance of national advocacy networks in anti-corporate social movements," *Organization Studies*, vol. 34, no. 5-6, pp. 823–852, 2013.
- [175] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.
- [176] J. Jiang, M. S. Kamel, and L. Chen, "Aggregation of multiple reinforcement learning algorithms," *International Journal on Artificial Intelligence Tools*, vol. 15, no. 05, pp. 855–861, 2006.
- [177] H. Iima and Y. Kuroe, "Swarm reinforcement learning algorithms based on particle swarm optimization," in *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*. IEEE, 2008, pp. 1110–1115.
- [178] — —, "Swarm reinforcement learning method based on an actor-critic method," in *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2010, pp. 279–288.
- [179] B. H. Abed-Alguni, D. J. Paul, S. K. Chalup, and F. A. Henskens, "A comparison study of cooperative q-learning algorithms for independent learners," *Int. J. Artif. Intell.*, vol. 14, no. 1, pp. 71–93, 2016.
- [180] P. Crook and G. Hayes, "Learning in a state of confusion: Perceptual aliasing in grid world navigation," *Towards Intelligent Mobile Robots*, vol. 4, 2003.
- [181] D. T. McRuer, D. Graham, and I. Ashkenas, *Aircraft dynamics and automatic control*. Princeton University Press, 2014, vol. 740.
- [182] A. Norin, "Airport logistics: modeling and optimizing the turn-around process," Ph.D. dissertation, Linköping University Electronic Press, 2008.
- [183] A. Kazda and R. E. Caves, *Airport design and operation*. Emerald Group Publishing Limited, 2010.

- [184] T. Reynolds, R. Neufville, P. Beloboba, and A. Odoni, "Airport systems, planning, design and management," 2013.
- [185] N. J. Ashford, S. Mumayiz, and P. H. Wright, *Airport engineering: planning, design, and development of 21st century airports*. John Wiley & Sons, 2011.
- [186] S. Kalakou, "Performance evaluation of passenger-related processes at an airport with a simulation model acknowledgments," *Lisbon, Portugal: Instituto Superior Técnico*, 2012.
- [187] B. Pearce, "The state of air transport markets and the airline industry after the great recession," *Journal of Air Transport Management*, vol. 21, pp. 3–9, 2012.
- [188] A. Lim and F. Wang, "Robust airport gate assignment," in *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on*, Nov 2005, pp. 8 pp.–81.
- [189] C.-H. Cheng, S. C. Ho, and C.-L. Kwan, "The use of meta-heuristics for airport gate assignment," *Expert Syst. Appl.*, vol. 39, no. 16, pp. 12430–12437, Nov. 2012.
- [190] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of constraint programming*. Elsevier, 2006.
- [191] U. M. Neuman and J. A. Atkin, "Airport gate assignment considering ground movement," in *International Conference on Computational Logistics*. Springer, 2013, pp. 184–198.
- [192] U. Dorndorf, F. Jaehn, and E. Pesch, "Modelling robust flight-gate scheduling as a clique partitioning problem," *Transportation Science*, vol. 42, no. 3, pp. 292–301, Aug. 2008.
- [193] A. Lim, B. Rodrigues, and Y. Zhu, "Airport gate scheduling with time windows," *Artif. Intell. Rev.*, vol. 24, no. 1, pp. 5–31, Sep. 2005. [Online]. Available: <http://dx.doi.org/10.1007/s10462-004-7190-4>
- [194] S. H. Kim, E. Feron, and J.-P. Clarke, "Airport gate assignment that minimizes passenger flow in terminals and aircraft congestion on ramps," in *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 7693.
- [195] V. Prem Kumar and M. Bierlaire, "Multi-objective airport gate assignment problem in planning and operations," *Journal of advanced transportation*, vol. 48, no. 7, pp. 902–926, 2014.
- [196] Y. Nikulin and A. Drexl, "Theoretical aspects of multicriteria flight gate scheduling: deterministic and fuzzy models," *Journal of Scheduling*, vol. 13, no. 3, pp. 261–280, 2010.
- [197] S. Yan and C.-M. Huo, "Optimization of multiple objective gate assignments," *Transportation Research Part A: Policy and Practice*, vol. 35, no. 5, pp. 413 – 432, 2001.
- [198] A. Drexl and Y. Nikulin, "Multicriteria airport gate assignment and pareto simulated annealing," *IIE Transactions*, vol. 40, no. 4, pp. 385–397, 2008.
- [199] D. Teodorovic and S. Guberinic, "Optimal dispatching strategy on an airline network after a schedule perturbation," *European Journal of Operational Research*, vol. 15, no. 2, pp. 178 – 182, 1984. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0377221784902078>
- [200] C. Chang, *Flight Sequencing and Gate Assignments at Airport Hubs*. University of Maryland at College Park, 1994.
- [201] J. Xu and G. Bailey, "The airport gate assignment problem: Mathematical model and a tabu search algorithm," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3 - Volume 3*, ser. HICSS '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 3032–.
- [202] U. Dorndorf, F. Jaehn, and E. Pesch, "Flight gate scheduling with respect to a reference schedule," *Annals of Operations Research*, vol. 194, no. 1, pp. 177–187, 2012.
- [203] S. H. Kim, E. Feron, and J.-P. Clarke, "Assigning gates by resolving physical conflicts," in *AIAA Guidance, Navigation, and Control Conference*, 2009, p. 5648.

- [204] X.-B. Hu and E. Di Paolo, "An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem," in *Multi-objective memetic algorithms*. Springer, 2009, pp. 71–89.
- [205] H. M. Genç, O. K. Erol, I. Eksin, M. F. Berber, and B. O. Güleriyüz, "A stochastic neighborhood search approach for airport gate assignment problem," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 316–327, Jan. 2012.
- [206] S. G. Hamzawi, "Management and planning of airport gate capacity: a microcomputer-based gate assignment simulation model," *Transportation Planning and Technology*, vol. 11, no. 3, pp. 189–202, 1986.
- [207] H. P. Williams, *Model building in mathematical programming*. John Wiley & Sons, 2013.
- [208] M. Hassounah and G. N. Steuart, "Demand for aircraft gates," *ransportation Research Record*, n 1423, pp.26-33, 1993.
- [209] S. Yan and C.-M. Chang, "A network model for gate assignment," *Journal of Advanced Transportation*, vol. 32, no. 2, pp. 176–189, 1998.
- [210] B. Maharjan and T. I. Matis, "Multi-commodity flow network model of the flight gate assignment problem," *Computers & Industrial Engineering*, vol. 63, no. 4, pp. 1135 – 1144, 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0360835212001799>
- [211] S. Yan and C.-H. Tang, "A heuristic approach for airport gate assignments for stochastic flight delays," *European Journal of Operational Research*, vol. 180, no. 2, pp. 547 – 567, 2007.
- [212] M. Seker and N. Noyan, "Stochastic optimization models for the airport gate assignment problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 2, pp. 438 – 459, 2012.
- [213] K. Tanaka, "An introduction to fuzzy logic for practical applications," 1997.
- [214] D.-x. Wei and C.-y. Liu, "Fuzzy model and optimization for airport gate assignment problem," in *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, vol. 2. IEEE, 2009, pp. 828–832.
- [215] M. Dell’Orco, M. Marinelli, and M. G. Altieri, "Solving the gate assignment problem through the fuzzy bee colony optimization," *Transportation Research Part C: Emerging Technologies*, vol. 80, pp. 424–438, 2017.
- [216] G. M. Marakas, *Decision support systems in the 21st century*. Prentice Hall Upper Saddle River, NJ, 2003, vol. 134.
- [217] G.-S. Jo, J.-J. Jung, and C.-Y. Yang, "Expert system for scheduling in an airline gate allocation," *Expert systems with applications*, vol. 13, no. 4, pp. 275–282, 1997.
- [218] Y. Cheng, "A knowledge-based airport gate assignment system integrated with mathematical programming," *Computers & Industrial Engineering*, vol. 32, no. 4, pp. 837–852, 1997.
- [219] Y. Gu and C. A. Chung, "Genetic algorithm approach to aircraft gate reassignment problem," *Journal of Transportation Engineering*, vol. 125, no. 5, pp. 384–389, 1999.
- [220] L. Tang, S. Jiang, and J. Liu, "Rolling horizon approach for dynamic parallel machine scheduling problem with release times," *Industrial & Engineering Chemistry Research*, vol. 49, no. 1, pp. 381–389, 2009.
- [221] B. Maharjan and T. I. Matis, "An optimization model for gate reassignment in response to flight delays," *Journal of Air Transport Management*, vol. 17, no. 4, pp. 256–261, 2011.
- [222] M. Pternea and A. Haghani, "Mathematical models for flight-to-gate reassignment with passenger flows: State-of-the-art comparative analysis, formulation improvement, and a new multidimensional assignment model," *Computers & Industrial Engineering*, vol. 123, pp. 103–118, 2018.

- [223] T. Obata, "Quadratic assignment problem: evaluation of exact and heuristic algorithms," 1979.
- [224] J.-Y. Greff, L. Idoumghar, and R. Schott, "AnglaisApplication of markov decision processes to the frequency assignment problem," *AnglaisJournal on Applied Artificial Intelligence*, vol. 18, no. 8, pp. 761–773, 2004, article dans revue scientifique avec comité de lecture. internationale. A04-R-208 || greff04a A04-R-208 || greff04a.
- [225] H. Yanco and L. A. Stein, "An adaptive communication protocol for cooperating mobile robots," in *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. MIT Press, 1993, pp. 478–485.
- [226] H. W. Kuhn and A. W. Tucker, "John von neumann's work in the theory of games and mathematical economics," *Bulletin of the American Mathematical Society*, vol. 64, pp. 100–122, 05 1958.
- [227] J. v. d. Wal, "Stochastic dynamic programming: successive approximations and nearly optimal strategies for markov decision processes and markov games / j. van der wal," 1981, includes indexes.
- [228] O. Aoun and A. El Afia, "Application of multi-agent markov decision processes to gate assignment problem," in *Information Science and Technology (CIST), 2014 Third IEEE International Colloquium in*, Oct 2014, pp. 196–201.
- [229] J. Wang, S. Hou, Y. Su, J. Du, and W. Wang, "Markov decision process based multi-agent system applied to aeroengine maintenance policy optimization," in *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on*, vol. 3, Oct 2008, pp. 401–408.
- [230] G. S. Bastos, L. E. Souza, F. T. Ramos, and C. H. Ribeiro, "A single-dependent agent approach for stochastic time-dependent truck dispatching in open-pit mining," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*. IEEE, 2011, pp. 1057–1062.
- [231] D. Jacquemart and J. Morio, "Conflict probability estimation between aircraft with dynamic importance splitting," *Safety Science*, vol. 51, no. 1, pp. 94–100, 2013.
- [232] O. Ö. Özener, M. Ö. Matoglu, G. Erdogän, M. Haouari, and H. Sözer, "Solving a large-scale integrated fleet assignment and crew pairing problem," *Annals of Operations Research*, vol. 253, no. 1, pp. 477–500, 2017.
- [233] A. Kasirzadeh, M. Saddoune, and F. Soumis, "Airline crew scheduling: models, algorithms, and data sets," *EURO Journal on Transportation and Logistics*, vol. 6, no. 2, pp. 111–137, 2017.
- [234] A. Scholl *et al.*, "Robuste planung und optimierung: Grundlagen, konzepte und methoden; experimentelle untersuchungen," Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL), Tech. Rep., 2000.
- [235] V. Dück, "Increasing stability of aircraft and crew schedules," Ph.D. dissertation, PhD thesis, University Paderborn, 2010.
- [236] S. Yan and Y. Tu, "A network model for airline cabin crew scheduling," *European Journal of Operational Research*, vol. 140, no. 3, pp. 531–540, 2002-08-01T00:00:00.
- [237] E. Andersson, E. Housos, N. Kohl, and D. Wedelin, "Crew pairing optimization," in *Operations Research in the Airline Industry*, ser. International Series in Operations Research & Management Science, G. Yu, Ed. Springer US, 1998, vol. 9, pp. 228–258.
- [238] C. P. Medard and N. Sawhney, "Airline crew scheduling from planning to operations," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1013 – 1027, 2007.
- [239] J. Hu and E. L. Johnson, "Computational results with a primal–dual subproblem simplex method," *Operations Research Letters*, vol. 25, no. 4, pp. 149–157, 1999.
- [240] B. Gopalakrishnan and E. Johnson, "Airline crew scheduling: State-of-the-art," *Annals of Operations Research*, vol. 140, no. 1, pp. 305–337, 2005.

- [241] B. Crawford, C. Castro, and E. Monfroy, "A hybrid ant algorithm for the airline crew pairing problem," in *Proceedings of the 5th Mexican International Conference on Artificial Intelligence*, ser. MICAI'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 381–391.
- [242] C. Barnhart, E. L. Johnson, G. L. Nemhauser, and P. H. Vance, *Crew Scheduling*. Boston, MA: Springer US, 1999, pp. 493–521. [Online]. Available: [https://doi.org/10.1007/978-1-4615-5203-1\\_14](https://doi.org/10.1007/978-1-4615-5203-1_14)
- [243] H. D. Chu, E. Gelman, and E. L. Johnson, "Solving large scale crew scheduling problems," *European Journal of Operational Research*, vol. 97, no. 2, pp. 260–268, 1997.
- [244] P. H. Vance, A. Atamturk, C. Barnhart, E. Gelman, E. L. Johnson, A. Krishna, D. Mahidhara, G. L. Nemhauser, and R. Rebello, "A heuristic branch-and-price approach for the airline crew pairing problem," *preprint*, 1997.
- [245] G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon, and F. Soumis, "Crew pairing at air france," *European journal of operational research*, vol. 97, no. 2, pp. 245–259, 1997.
- [246] A. Makri and D. Klabjan, "A new pricing scheme for airline crew scheduling," *INFORMS Journal on Computing*, vol. 16, no. 1, pp. 56–67, 2004.
- [247] P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser, "Airline crew scheduling: A new formulation and decomposition algorithm," *Operations Research*, vol. 45, no. 2, pp. 188–200, 1997.
- [248] R. Borndörfer, U. Schelten, T. Schlechte, and S. Weider, "A column generation approach to airline crew scheduling," in *Operations Research Proceedings 2005*. Springer, 2006, pp. 343–348.
- [249] F. Zeghal and M. Minoux, "Modeling and solving a crew assignment problem in air transportation," *European Journal of Operational Research*, vol. 175, no. 1, pp. 187–209, 2006.
- [250] D. Teodorovic and G. Stojkovic, "Model to reduce airline schedule disturbances," *Journal of Transportation Engineering*, vol. 121, no. 4, pp. 324–331, 1995.
- [251] G. Wei, G. Yu, and M. Song, "Optimization model and algorithm for crew management during airline irregular operations," *Journal of Combinatorial Optimization*, vol. 1, no. 3, pp. 305–321, 1997.
- [252] L. Lettovsky, E. L. Johnson, and G. L. Nemhauser, "Airline crew recovery," *Transportation Science*, vol. 34, no. 4, pp. 337–348, 2000.
- [253] M. Stojkovic and F. Soumis, "An optimization model for the simultaneous operational flight and pilot scheduling problem," *Management Science*, vol. 47, no. 9, pp. 1290–1305, 2001.
- [254] A. J. Schaefer, E. L. Johnson, A. J. Kleywegt, and G. L. Nemhauser, "Airline crew scheduling under uncertainty," *Transportation Science*, vol. 39, no. 3, pp. 340–348, 2005.
- [255] M. Ehrgott and D. M. Ryan, "Constructing robust crew schedules with bicriteria optimization," *Journal of Multi-Criteria Decision Analysis*, vol. 11, no. 3, pp. 139–150, 2002.
- [256] O. Weide, D. Ryan, and M. Ehrgott, "An iterative approach to robust and integrated aircraft routing and crew scheduling," *Computers & Operations Research*, vol. 37, no. 5, pp. 833–844, 2010, *disruption Management*.
- [257] J. W. Yen and J. R. Birge, "A stochastic programming approach to the airline crew scheduling problem," *Transportation Science*, vol. 40, no. 1, pp. 3–14, 2006.
- [258] V. Dück, L. Ionescu, N. Kliewer, and L. Suhl, "Increasing stability of crew and aircraft schedules," *Transportation research part C: emerging technologies*, vol. 20, no. 1, pp. 47–61, 2012.
- [259] D. Lu and F. Gzara, "The robust crew pairing problem: model and solution methodology," *Journal of Global Optimization*, vol. 62, no. 1, pp. 29–54, 2015.
- [260] C. Barnhart, E. Johnson, G. Nemhauser, and P. Vance, "Crew scheduling," *European Journal of Operational Research*, pp. 493–521, 1997.
- [261] V. Duck, L. Ionescu, N. Kliewer, and L. Suhl, "Increasing stability of crew and aircraft schedules," *Transportation Research Part C: Emerging Technologies*, vol. 20, no. 1, pp. 47 – 61, 2012,



- special issue on Optimization in Public Transport, Special issue on Optimization in Public Transport, International Symposium on Transportation and Traffic Theory (ISTTT), Berkeley, California, July 18-20, 2011.
- [262] J. M. Rosenberger, A. J. Schaefer, D. Goldsman, E. L. Johnson, A. J. Kleywegt, and G. L. Nemhauser, "A stochastic model of airline operations," *Transportation Science*, vol. 36, no. 4, pp. 357-377, Nov. 2002.
- [263] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm optimization," *Proceeding of the conference on System, Man, and Cybernetics*, vol. 4, pp. 104-109, 1997.
- [264] A. Eiben and S. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, pp. 19 - 31, 2011.
- [265] A. Ratnweera, S. Halgamuge, and H. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 240 - 255, 2004.
- [266] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. IEEE, 2007, pp. 120-127.
- [267] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stuetzle, "Paramils: an automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, vol. 36, no. 1, pp. 267-306, 2009.
- [268] A. J. Schaefer, E. L. Johnson, A. J. Kleywegt, and G. L. Nemhauser, "Airline crew scheduling under uncertainty," *Transportation Science*, Tech. Rep., 2001.
- [269] R. Givan, S. Leach, and T. Dean, "Bounded-parameter markov decision process," *Artif. Intell.*, vol. 122, no. 1-2, pp. 71-109, Sep. 2000.
- [270] O. Candell, R. Karim, and P. Söderholm, "eMaintenance-information logistics for maintenance support," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 6, pp. 937-944, 2009.
- [271] G. Radnoti and S. Carr, *Profit strategies for air transportation*. McGraw-Hill New York, NY, 2002.
- [272] A. Al-Garni, M. Tozan, and W. Abdelrahman, "Graphical techniques for managing field failures of aircraft systems and components," *Journal of Aircraft*, vol. 46, no. 2, pp. 608-616, 2009.
- [273] S. Duffuaa and A. Andijani, "An integrated simulation model for effective planning of maintenance operations for saudi arabian airlines (saudia)," *Production Planning & Control*, vol. 10, no. 6, pp. 579-584, 1999.
- [274] K. A. Latorella and C. G. Drury, "A framework for human reliability in aircraft inspection," in *Proceedings of the Seventh FAA Meeting on Human Factors Issues in Aircraft Maintenance and Inspection*. Federal Aviation Administration, Washington, DC, 1992.
- [275] K. A. Latorella and P. V. Prabhu, "A review of human error in aviation maintenance and inspection," *International Journal of industrial ergonomics*, vol. 26, no. 2, pp. 133-161, 2000.
- [276] M. Masmoudi and A. Hat, "Fuzzy uncertainty modelling for project planning: application to helicopter maintenance," *International Journal of Production Research*, vol. 50, no. 13, pp. 3594-3611, 2012.
- [277] C. Friend, *Aircraft maintenance management*. Longman Publishing Group, 1992.
- [278] P. Samaranayake and S. Kiridena, "Aircraft maintenance planning and scheduling: an integrated framework," *Journal of Quality in Maintenance Engineering*, vol. 18, no. 4, pp. 432-453, 2012.
- [279] C. Barnhart, N. L. Boland, L. W. Clarke, E. L. Johnson, G. L. Nemhauser, and R. G. Sheno, "Flight string models for aircraft fleet and routing," *Transportation Science*, vol. 32, no. 3, pp. 208-220, 1998. [Online]. Available: <http://dx.doi.org/10.1287/trsc.32.3.208>
- [280] S. Lan, J.-P. Clarke, and C. Barnhart, "Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions," *Transportation Science*,

- vol. 40, no. 1, pp. 15–28, 2006. [Online]. Available: <http://pubsonline.informs.org/doi/abs/10.1287/trsc.1050.0134>
- [281] L. Clarke, E. Johnson, G. Nemhauser, and Z. Zhu, “The aircraft rotation problem,” *Annals of Operations Research*, vol. 69, no. 0, pp. 33–46, 1997. [Online]. Available: <http://dx.doi.org/10.1023/A:1018945415148>
- [282] Z. Liang and W. A. Chaovalitwongse, *The Aircraft Maintenance Routing Problem*. Boston, MA: Springer US, 2009, pp. 327–348. [Online]. Available: [http://dx.doi.org/10.1007/978-0-387-88617-6\\_12](http://dx.doi.org/10.1007/978-0-387-88617-6_12)
- [283] M. Junger, M. Elf, and V. Kaibel, “Rotation planning for the continental service of a european airline,” in *Mathematics-Key Technology for the Future*. Springer, 2003, pp. 675–689.
- [284] “A network-based model for the integrated weekly aircraft maintenance routing and fleet assignment problem,” *Transportation Science*, vol. 47, no. 4, pp. 493–507, Nov. 2013. [Online]. Available: <http://dx.doi.org/10.1287/trsc.1120.0434>
- [285] R. Gopalan and K. T. Talluri, “The aircraft maintenance routing problem,” *Operations Research*, vol. 46, no. 2, pp. 260–271, 1998. [Online]. Available: <http://pubsonline.informs.org/doi/abs/10.1287/opre.46.2.260>
- [286] K. T. Talluri, “The four-day aircraft maintenance routing problem,” *Transportation Science*, vol. 32, no. 1, pp. 43–53, 1998. [Online]. Available: <http://dx.doi.org/10.1287/trsc.32.1.43>
- [287] A. Sarac, R. Batta, and C. M. Rump, “A branch-and-price approach for operational aircraft maintenance routing,” *European Journal of Operational Research*, vol. 175, no. 3, pp. 1850 – 1869, 2006. [Online]. Available: [//www.sciencedirect.com/science/article/pii/S0377221705004807](http://www.sciencedirect.com/science/article/pii/S0377221705004807)
- [288] C. Sriram and A. Haghani, “An optimization model for aircraft maintenance scheduling and re-assignment,” *Transportation Research Part A: Policy and Practice*, vol. 37, no. 1, pp. 29–48, 2003.
- [289] C. A. Hane, C. Barnhart, E. L. Johnson, R. E. Marsten, G. L. Nemhauser, and G. Sigismondi, “The fleet assignment problem: Solving a large-scale integer program,” *Mathematical Programming*, vol. 70, no. 1, pp. 211–232, 1995.
- [290] D. Klabjan, E. L. Johnson, G. L. Nemhauser, E. Gelman, and S. Ramaswamy, “Airline crew scheduling with time windows and plane-count constraints,” *Transportation Science*, vol. 36, no. 3, pp. 337–348, 2002. [Online]. Available: <http://dx.doi.org/10.1287/trsc.36.3.337.7831>
- [291] J.-F. Cordeau, G. Stojkovic, F. Soumis, and J. Desrosiers, “Benders decomposition for simultaneous aircraft routing and crew scheduling,” *Transportation Science*, vol. 35, no. 4, pp. 375–388, 2001. [Online]. Available: <http://dx.doi.org/10.1287/trsc.35.4.375.10432>
- [292] A. M. Cohn and C. Barnhart, “Improving crew scheduling by incorporating key maintenance routing decisions,” *Operations Research*, vol. 51, no. 3, pp. 387–396, 2003. [Online]. Available: <http://pubsonline.informs.org/doi/abs/10.1287/opre.51.3.387.14759>
- [293] O. Aoun, M. Sarhani, and A. El Afia, “Investigation of hidden markov model for the tuning of metaheuristics in airline scheduling problems,” *IFAC-PapersOnLine*, vol. 49, no. 3, pp. 347–352, 2016.
- [294] O. Aoun and A. El Afia, “Using markov decision processes to solve stochastic gate assignment problem,” in *Logistics and Operations Management (GOL), 2014 International Conference on*, June 2014, pp. 42–47.
- [295] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.
- [296] J. Patokallio. Database for flight logging, mapping, stats, and sharing. <http://openflights.org>. OpenFlights. <http://OpenFlights.org>.
- [297] I. Chadès, M.-J. Cros, F. Garcia, and R. Sabbadin, “Markov decision processes (mdp) toolbox,” URL <http://www.inra.fr/mia/T/MDPtoolbox/42>, 2009.

---



# Appendix:

## Thesis publications

---

### Journals

1. Aoun, Oussama, Malek Sarhani, and Abdellatif El Afia (2018). "Hidden markov model classifier for the adaptive particle swarm optimization", In Recent Developments in Metaheuristics, pp. 1-15. Springer.
2. Abdellatif El Afia, Malek Sarhani and Aoun Oussama (2019). "A Probabilistic Finite State Machine Design of Particle Swarm Optimization", In Bioinspired Heuristics for Optimization, pp. 185-201. Springer.
3. Aoun, Oussama and Abdellatif El Afia (2018). "Time-Dependence in Multi-Agent MDP Applied to Gate Assignment Problem", INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS. Vol. 9(2), pp. 331-340. SCIENCE & INFORMATION SAI ORGANIZATION LTD 19 BOLLING RD, BRADFORD, WEST YORKSHIRE, 00000, ENGLAND
4. Aoun, Oussama and Abdellatif El Afia (2018). "Particle swarm optimisation with population size and acceleration coefficients adaptation using hidden Markov model state classification", International Journal of Metaheuristics. Vol. 7(1), pp. 1-29. Inderscience Publishers (IEL). .
5. Oussama Aoun, Malek Sarhani, and Abdellatif El Afia (2016). "Investigation of hidden markov model for the tuning of metaheuristics in airline scheduling problems". IFAC-PapersOnLine, 49(3), 347-352
6. Abdellatif El Afia, Malek Sarhani and Aoun Oussama (2017). "Hidden markov model control of inertia weight adaptation for Particle swarm optimization", IFAC-PapersOnLine. Vol. 50(1), pp. 9997-10002. Elsevier

### Conferences

1. Aoun Oussama and Abdellatif El Afia (2018). "Self Inertia Weight Adaptation for the Particle Swarm Optimization", In Proceedings of the International

- Conference on Learning and Optimization Algorithms: Theory and Applications, LOPAL 2018, Rabat, Morocco, May 2-5, 2018, pp. 8:1-8:6.
2. Abdellatif El Afia and Aoun Oussama (2017). "Data-driven based aircraft maintenance routing by markov decision process model", In Proceedings of the 2nd international Conference on Big Data, Cloud and Applications, BDCA 2017, Tetouan, Morocco, March 29-30, 2017. , pp. 74:1-74:6
  3. Aoun Oussama and Abdellatif El Afia (2014). "Using Markov decision processes to solve stochastic gate assignment problem", In International Conference on Logistics and Operations Management (GOL), June, 2014 , pp. 42-47.
  4. Aoun Oussama and Abdellatif El Afia (2014). "Application of multi-agent Markov decision processes to gate assignment problem", In Information Science and Technology (CIST), 2014 Third IEEE International Colloquium in., Oct, 2014. , pp. 196-201
  5. Aoun Oussama and Abdellatif El Afia (2014). "A robust crew pairing based on Multi-agent Markov Decision Processes", In Second World Conference on Complex Systems (WCCS), Nov, 2014. , pp. 762-768