A PROPOSAL FOR AN IMPROVED VERSION OF EIGENANT ALGORITHM WITH PERFORMANCE EVALUATION ON COMBINATORIAL OPTIMIZATION PROBLEMS

Mahan Mahrueyan

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Amit Bhaya

Rio de Janeiro
Agosto de 2017

A PROPOSAL FOR AN IMPROVED VERSION OF EIGENANT ALGORITHM WITH PERFORMANCE EVALUATION ON COMBINATORIAL OPTIMIZATION PROBLEMS

Mahan Mahrueyan

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Amit Bhaya, Ph.D.

Prof. Eugenius Kaszkurewicz, D.Sc.

Prof. Marley Maria Bernardes Rebuzzi Vellasco, Ph.D.

Prof. Nelson Francisco Favilla Ebecken, D.Sc.

Prof. Oumar Diene, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2017

Mahrueyan, Mahan

A proposal for an improved version of EigenAnt algorithm with performance evaluation on combinatorial optimization problems/Mahan Mahrueyan. – Rio de Janeiro: UFRJ/COPPE, 2017.

XVII, 112 p.: il.; 29, 7cm.

Orientador: Amit Bhaya

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2017.

Referências Bibliográficas: p. 98 – 103.

1. Parameter Impact Analysis. 2. Improved EigenAnt Algorithm. 3. Routing Networks. 4. Multidimensional Knapsack Problem. 5. Constraint Handling. 6. Dynamic Optimization Problem. I. Bhaya, Amit. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

I dedicate this Thesis to my
mother Mitra and my father
Masoud who always are available
for helping me because of their
infinite love for me.
I also dedicate this thesis to the
soul of my grandfather Mahmoud
who was my first teacher in math
before entering the school.

ACKNOWLEDGMENTS

I would like to thank Professor Amit Bhaya for his supervision and support. I would also like to thank the staff of NACAD, especially Ms. Mara Prata, for their kind support. I would also like to thank Dr. Rolando Cuevas for his helps upon my arrival when I did not know Portuguese. Finally, I would like to thank CAPES for their financial support.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

UMA PROPOSTA PARA UMA EXTENSÃO DO ALGORITMO EIGENANT COM AVALIAÇÃO DE DESEMPENHO EM PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA

Mahan Mahrueyan

Agosto/2017

Orientador: Amit Bhaya

Programa: Engenharia Elétrica

O algoritmo denominado EigenAnt foi introduzido recentemente para a resolução do problema de encontrar o menor caminho entre dois nós de um grafo, utilizando uma dinâmica com evaporação local de feromônio. O referido algoritmo possui uma prova matemática de convergência ao menor caminho. Nesta tese, realiza-se a análise de estabilidade e sensibilidade paramêtrica do algoritmo EigenAnt aplicado a problemas de caminhos mínimos em cadeias binárias entre N nós. Motivado por esta análise, propõe-se uma extensão do algoritmo EigenAnt (denotado Improved EigenAnt), no qual a exploração de distintos equilíbrios estáveis e a velocidade de convergência a estes podem ser ajustada independentemente. Realiza-se também uma análise comparativa entre os algoritmos EigenAnt, Improved EigenAnt e outros algoritmos existentes do tipo Colônia de Formigas, no contexto de problemas de caminho mínimo em redes de roteamento. Adicionalmente, aplica-se o algoritmo novo proposto a problemas multidimensionais de mochileiro, por meio da modelagem destes como problemas de caminhos mínimos em cadeias binárias entre N nós, com restrições. Evaporação local de feromônio e convergência rápida são propriedades de algoritmos da classe EigenAnt que tornam esta classe vantajosa para rastreamento de soluções ótimas de problemas de otimização dinâmica, nos quais as instâncias do problema, a função objetivo e os parâmetros das restrições podem mudar ao longo do tempo. Uma investigação experimental da aplicação do algoritmo proposto (Improved EigenAnt) para rastrear a solução ótima em redes dinâmicas de roteamento e em problemas dinâmicos do mochileiro constituem uma outra contribuição desta tese.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

A PROPOSAL FOR AN IMPROVED VERSION OF EIGENANT ALGORITHM WITH PERFORMANCE EVALUATION ON COMBINATORIAL OPTIMIZATION PROBLEMS

Mahan Mahrueyan

August/2017

Advisor: Amit Bhaya

Department: Electrical Engineering

The EigenAnt algorithm has recently been introduced to solve the problem of finding the shortest path between two nodes by using dynamics involving local pheromone evaporation. This algorithm has a mathematical proof of convergence to the shortest path between two nodes. In this thesis, the stability and parameter impact analysis of EigenAnt algorithm applied to N-node Binary Chain Problems is carried out. Motivated by this analysis, an improved EigenAnt algorithm is proposed, in which the exploration of different stable equilibria and speed of convergence to them can be tuned separately. A comparative analysis of Improved EigenAnt algorithm with its predecessor EigenAnt and other Ant Colony Optimization algorithms is performed for combinatorial Routing Network shortest path problems. In addition, the application of the proposed Improved EigenAnt algorithm to Multidimensional Knapsack Problems is investigated, by modeling these problems as an N-node Binary Chain shortest path problems with constraints. Local pheromone evaporation and fast convergence features of the EigenAnt algorithm are advantageous for tracking the optimal solutions of dynamic optimization problems in which the problem instances, objective function and constraint parameters change over time. An experimental investigation of the application of the proposed Improved EigenAnt algorithm to track the optimal Dynamic Routing Networks and Dynamic Multidimensional Knapsack problems is another contribution of this thesis.

Contents

List of Figures

List of Tables

List of Symbols

B	Total number of cities in TSP, p. 4
C	Pheromone deposition coefficient, p. 12
D	The total amount of pheromone deposited on a path in N-node BCP, p. 21
Du_e	Change cycle duration, p. 62
H	The finite countable set of all the possible edges forming a path, p. 21
L_{gb}	The cost of best constructed solution, p. 6
L_m	The cost of constructed solution for the ant m, p. 5
L_{nn}	Rough approximation of the optimal cost, p. 7
L_r	The cost of constructed solution r, p. 17
M	Total number of ants, p. 4
N	Total number of layers, p. 10
O	Total number of outgoing nodes, p. 7
$P(\alpha_1)$	The probability transition function in IEigenAnt, p. 17
$P(\alpha_2)$	The normalized function in the pheromone update of IEigenAnt, p. 17
Q	Scaling parameter, p. 5
R_{jx}	The finite countable set of all the possible solutions containing the edge jx, p. 21
S	The location of an equilibrium points, p. 20
V	The profit of the chosen item in MKP, p. 44

W Capacity of a constraint in MKP, p. 44

Z Total number of constraints, p. 44

Ω Penalty value, p. 47

Φ The ensemble transition probability for a path in N-node BCP, p. 21

Υ Pheromone decay parameter, p. 6

α Pheromone amplification parameter, p. 4

α_1 Pheromone amplification parameter for the IEigenAnt in its construction solution phase, p. 13

α_2 Pheromone amplification parameter for the IEigenAnt in its pheromone update phase, p. 13

β Heuristic amplification parameter, p. 4

χ Normalized constraint value, p. 49

δ Scaling factor used in the pheromone evaporation of SIEigenAnt, p. 24

η Heuristic function, p. 4

κ Deposition parameter, p. 12

λ Penalty factor, p. 47

μ_z The violation amount of a constraint z, p. 47

ν Scaling parameter for penalty factor, p. 49

ϕ Expanded objective function, p. 47

ρ Pheromone evaporation parameter, p. 5

σ Standard Deviation, p. 33

τ Pheromone trail concentration, p. 4

ζ Penalty amplification parameter, p. 47

a The surrogate multiplied, p. 45

d The distance between two nodes, p. 4

e	Change cycle event, p. 60
f	Evaluated Cost, p. 46
i	Index that represent the current node(layer) of an ant, p. 4
j	Index that represents the next node (layer) in which an ant goes to, p. 4
l	Counter index, p. 4
m	Index that represents the ant number, p. 4
n	Index that represents the node number, p. 4
q_0	Tuning parameter to control biased exploration vs exploitation search, p. 6
r	The index represents different constructed solutions, p. 17
u	Pseudo-utility value, p. 46
w_{zj}	The consumption of the constraint z by the chosen object j, p. 44
x	The index that represents the two possible choices at each layer of a BCP, and O possible choices at each layer of RN, p. 17
y	Temporary variable, p. 48
z	The index that represents the constraint number in MKP, p. 44

List of Abbreviations

ACO	Ant Colony Optimization, p. 1
ACS	Ant Colony System, p. 3
AS	Ant System, p. 4
BCP	Binary Chain Problem, p. 2
COP	Combinatorial Optimization Problem, p. 3
CS	Convergence in Solution, p. 2
CV	Convergence in Value, p. 2
DCOP	Dynamic Combinatorial Optimization Problem, p. 3, 57
DMKP	Dynamic Multidimensional Knapsack Problems, p. 3
DOP	Dynamic Optimization Problem, p. 57
DRN	Dynamic Routing Network, p. 3
EA	Evolutionary Algorithm, p. 57
IEigenAnt	Improved EigenAnt, p. 2
IN	Iteration Number, p. 51
LP	linear Programming, p. 45
MKP	Multidimensional Knapsack Problems, p. 3
NA	Not Available, p. 52
PMLV	Pheromone Minimum Limit Value, p. 51
RN	Routing Networks, p. 2
RR	Randomized Rounding, p. 51

RWS	Roulette-Wheel Selection, p. 4
SACO	Simple Ant Colony Optimization, p. 2
SCP	Set Covering Problem, p. 2
SD	Standard Deviation, p. 33
SIEigenAnt	Sorting Improved EigenAnt, p. 24
SOP	Sequential Ordering Problem, p. 2
TMO	Tracking the Moving of Optimum, p. 57
TSP	Traveling Salesman Problem, p. 4

Chapter 1

Introduction

Biologically inspired (bio-inspired) computing is an area of research that borrows ideas from areas that are traditionally classified, in broad terms, as biology: connectionism, social behavior and emergence. These ideas are applied to problems in optimization and machine learning, in a mathematical framework. The first major examples of bio-inspired computing came from evolutionary theory, leading to evolutionary computation and genetic algorithms. Another important development was the use of social behavior leading to the emergent solution of a problem. Specifically, Dorigo was inspired by the optimal foraging behavior of ants to propose the so-called Ant Colony Optimization (ACO) paradigm, in which agents (ants) deposit pheromone as they travel and can also sense the concentration of pheromone that may already be present on a path. Amongst a set of alternative paths that lead from the source to the destination, the path that happens to be more heavily traveled on by ants, and therefore has a higher concentration of pheromone, tends to be chosen by the ants that follow (BONABEAU *et al.*, 1999; DENEUBOURG *et al.*, 1990; DORIGO *et al.*, 2006). The seemingly simple behavior of depositing and sensing pheromone can lead to the emergence of a foraging trail on the shortest path between the source and the destination.

In the most basic application of ACO, a set of artificial ants find the shortest path between a source and a destination. Ants deposit pheromone on paths they take, preferring paths that have more pheromone on them. Since shorter paths are traversed faster, more pheromone accumulates on them in a given time, attracting more ants and leading to reinforcement of the pheromone trail on shorter paths. This is a positive feedback process, that can also cause trails to persist on longer paths, even when a shorter path becomes available. To counteract this persistence on a longer path, ACO algorithms employ remedial measures, such as using negative feedback in the form of uniform evaporation on all paths. The paper (JAYADEVA *et al.*, 2013) proposed a new ACO algorithm, called EigenAnt, for finding the shortest path between a source and a destination, based on selective pheromone removal

that occurs only on the path that is actually chosen for each trip. In other words, EigenAnt algorithm has local pheromone evaporation unlike the global pheromone evaporation in the conventional ACO algorithms.

In an attempt to develop the theoretical aspects of ACO, it was proved that the shortest path is the only stable equilibrium for the EigenAnt algorithm to find the shortest path between two nodes. The proof of the EigenAnt algorithms shows that this property is maintained for arbitrary initial pheromone concentrations on paths, and even when path lengths change with time (JAYADEVA *et al.*, 2013). Furthermore, EigenAnt was applied successfully to Routing Networks (RN) (JAYADEVA *et al.*, 2013; SHAH, 2011), Sequential Ordering Problem (EZZAT *et al.*, 2014) and Set Covering Problem (KUMAR, 2016). However, none of the cited papers present a theoretical analysis approach to justify the success of EigenAnt dealing with larger problems than the basic problem of finding the shortest path between two nodes.

1.1 Brief Description of the Objectives of this Thesis

A stability analysis given in (IACOPINO and PALMER, 2012) for the application of Simple ACO (SACO) algorithm (DORIGO and STÜTZLE, 2001) to N-node Binary Chain Problems (BCP) motivates us to map the EigenAnt algorithm to the analytical model used for such analysis in order to get a theoretical insight for the application of EigenAnt to larger problems. Apart from the stability analysis, a parameter impact analysis is done for SACO application to N-node BCPs in (IACOPINO and PALMER, 2012). In order to follow the same direction for the EigenAnt, we propose the introduction of two additional parameters into the EigenAnt algorithm and refer to this as the Improved EigenAnt algorithm. The Improved EigenAnt demonstrates some promising features due to the fact that the introduction of the proposed parameters allows stability and speed of convergence issued to be treated separately and simultaneously. In contrast, in SACO (IACOPINO and PALMER, 2012), one can choose parameters to achieve either stability or speed of convergence, but not both simultaneously. Furthermore, we apply IEigenAnt to RNs in order to investigate the theoretical analysis done for the extended versions of BCP with number of edges more than two at each layer. We compare the results from two points of view: Convergence in Solution (CS) (i.e., the algorithm converges to a situation in which it generates the optimal solution over and over) and Convergence in Value (CV) (i.e., the algorithm will eventually find the optimal solution) (STÜTZLE and DORIGO, 2002). We also investigate

the influence of constraints on the stability analysis done for BCP by empirically analyzing the application of IEigenAnt to Multidimensional Knapsack Problems (MKP) modeled as an N-node BCPs.

Many real-world optimization problems are Combinatorial Optimization Problems (COP) subject to dynamic environments. In such Dynamic Combinatorial Optimization Problems (DCOPs), the objective, decision variables and/or constraints may change over time (YANG *et al.*, 2013). It is a challenge for an ACO solving a DCOP to track the optimal solutions because the algorithm might stagnate as the pheromone trails converge to an optimal solution that is no longer valid after the change takes place. EigenAnt demonstrated a promising performance dealing with Dynamic Routing Networks (DRN) due to its local pheromone evaporation (pheromone removal) (SHAH, 2011). In other words, EigenAnt keeps its convergence behavior while avoids stagnation problem which permits us to use the results based on CS point of view. Being able to use CS point of view when dealing with DCOP is an advantage for an algorithm because the CV point of view has the downside of missing the to date best optimal solution whenever the new optimal solution is greater than the previous one. For this reason, knowledge of the change time is essential for ACO algorithms that consider the CV point of view, in order that the algorithm be able to reset the best-to-date optimal value soon after the change occurs so as to not miss the new optimal value greater than the previous one. The fundamental drawback of the CV point of view is rarely discussed in the papers on the subject of solving DCOP with ACO algorithms since they assume, without further comment, that the change time is known. In this thesis, we compare the performance of IEigenAnt with EigenAnt dealing with challenging scenarios in DRN.

Very little work has been done in solving DCOPs with constraints that change over time, which makes the subject interesting. The Dynamic Multidimensional Knapsack Problem (DMKP) is solved by Ant Colony System (ACS) algorithms which is similar to the EigenAnt in the sense of using local pheromone evaporation (RANDALL, 2005). However, the paper used a correction procedure for avoiding violation of constraints which is difficult to implement in practice. We solve DMKP problem with IEigenAnt algorithm without any correction procedure due to the use of BCP modeling for DMKP that increases the adaptability of the algorithm as suggested in (BRANKE *et al.*, 2006).

1.2 Review of the Main Classes of Ant Colony Algorithms

In this section we give a brief review of the main classes of ACO algorithms, taking the opportunity to cite the main papers in the area.

1.2.1 ACO Algorithm

Ant System (AS)

The Ant colony optimization algorithm was first introduced in (DORIGO *et al.*, 1991) as a meta-heuristic method called Ant System (AS) to solve the well-known Traveling Salesman Problem (TSP). Since the application of ACO algorithms to TSP has been used as a benchmark to propose ACO algorithms in order to solve COPs, we briefly explain the application of ACO algorithms to solve TSP.

TSP can be stated as follows: given a list of B cities and the distances between each pair of cities i and j as d_{ij} , what is the shortest possible route that visits each city exactly once and returns to the original city?

In order to apply ACO algorithms to TSP, three phases are usually defined.

1- Solution Construction:

ACO algorithms are defined by the movement of M ants between B cities. Each ant m starts its travel from a random city n_1 and selects its next city through a random procedure called Roulette-Wheel Selection (RWS) until the last one is reached.

Roulette-Wheel Selection is a probabilistic rule to define the choice of the next city j for the ant located in the current city i. This could be imagined similar to a Roulette wheel in a casino. Usually a proportion of the wheel is assigned to each of the possible selections based on their fitness value. This could be achieved by dividing the fitness of a selection by the total fitness of all the selections, thereby normalizing them to 1. Then a random selection is made similar to how the roulette wheel is rotated. The transition probability function used in the RWS is defined for the ant m as follows:

$$p_{ij}^{m} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l\in\mathcal{N}_i^m}\tau_{il}^{\alpha}\eta_{il}^{\beta}} \tag{1.1}$$

where $\mathcal{N}_i^m$ is a set of available cities for the ant m situated in the city i, η_{ij} is a heuristic function, β is the heuristic amplification parameter, τ_{ij} is the pheromone trail concentration and α is the pheromone amplification parameter. Parameter

tuning of α and β is a challenge. The methods for parameter tuning in ACO algorithms are discussed in (WONG *et al.*, 2008). In TSP, η_{ij} is set to $\frac{1}{d_{ij}}$. In TSP, each pheromone trail τ_{ij} corresponds with each graph edge. Apart from the proper initialization of pheromone trail concentrations, finding the appropriate values for all parameters is crucial for an AS algorithm.

To exemplify, for the m^{th} ant starting from city n_1 the second city n_2 is selected by the ant m according to the transition probability p_{12}^m. Then, ant m selects the third city among the remaining $B-2$ cities via a transition probability function p_{23}^m. This continues until the last city in which the transition probability for selecting it from the last city n_B would be as $p_{B-1,1}^m = 1$. At the end, all the cities from n_1 to n_B are saved as the solution (path) constructed by the ant m.

2- Cost Evaluation:

Having constructed a solution, a function is defined to evaluate the cost of the constructed solution. The constructed solution cost for the ant m is denoted by L_m. In TSP, the constructed solution cost is the summation of all edge lengths d_{ij} starting from n_1 through all the other cities until it returns to n_1. Generally speaking, the number of cost function evaluations is a criterion for comparing the complexity between ACO algorithms.

3- Pheromone Update:

Pheromone update phase takes place in two steps at each iteration. First, a global process named pheromone evaporation reduces all the τ_{ij} globally for all the edges as follows:

$$\tau_{ij} = (1 - \rho)\, \tau_{ij} \tag{1.2}$$

where $\rho \in [0, 1]$ is the evaporation parameter. Second, the pheromone deposition process occurs only for the nodes included in the constructed solution by each ant m upon returning from the path it traversed as follows:

$$\tau_{ij}^m = \tau_{ij} + \triangle\tau_{ij}^m \tag{1.3}$$

where different values are defined for the $\triangle\tau_{ij}^m$ in the various versions of the AS. In (DORIGO *et al.*, 1991), three versions of AS are suggested based on different values for the $\triangle\tau_{ij}^m$ in Eq. (1.3): Ant-density in which $\triangle\tau_{ij}^m = Q$, Ant-quantity in which $\triangle\tau_{ij}^m = \frac{Q}{d_{ij}}$ and Ant-cycle in which $\triangle\tau_{ij}^m = \frac{1}{L_m}$. It should be mentioned that Q is a scaling parameter. In (DORIGO *et al.*, 1996), it was concluded that Ant-cycle in which uses the whole constructed solution cost L_m in pheromone update gives better results than the other versions.

These three phases are carried out in each global iteration for all the ants, until

a stopping criterion is satisfied.

Another version of AS is SACO which does not have the heuristic function η_{ij} in (Eq. 1.1) (DORIGO and STÜTZLE, 2001):

$$p_{ij}^{m} = \frac{\tau_{ij}^{\alpha}}{\sum_{l \in \mathcal{N}_i^m} \tau_{il}^{\alpha}} \tag{1.4}$$

In (DORIGO *et al.*, 1996) the parameters for AS are suggested as follows:

$$Q = 10, \rho = 0.5$$
$$\alpha = 1, \beta = 5$$

Ant Colony System (ACS)

Another version of ACO algorithm is also proposed as ACS to solve TSP (DORIGO and GAMBARDELLA, 1997). Three phases of ACS are as follows:

1-Solution Construction:

In ACS, a parameter $0 \leq q_0 \leq 1$ is pre-defined. Then, a random number $0 \leq q \leq 1$ is generated and compared with the pre-defined q_0 whenever an ant wants to move from a city i to the city j. If $q > q_0$ the solution construction procedure takes place through RWS. However, if $q \leq q_0$, the ant at city i goes to the city j with the maximum value of $\tau_{ij}\eta_{ij}^{\beta}$. The first is entitled biased exploration search and the latter is entitled exploitation. In contrast with AS, α in Eq.(1.1) is not in the ACS solution construction phase.

2- Cost Evaluation:

This phase is exactly the same as in AS.

3-Pheromone Update

The pheromone update phase in ACS is done in two steps:

- **Global Update:** Takes place for the pheromone trails of the best constructed solution to date. The dynamic for the global pheromone update in ACS is as follows:

$$\tau_{ij} = (1 - \Upsilon)\,\tau_{ij} + \Upsilon \triangle_{ij} \tag{1.5}$$

  where $0 \leq \Upsilon \leq 1$ is the pheromone decay parameter and $\triangle_{ij}$ is $\frac{1}{L_{gb}}$ where L_{gb} denotes cost of the best constructed solution to date.

- **Local Update:** The local update has the objective of avoiding stagnation by decreasing the pheromone value on the previously used edges and making them

less attractive for other ants. In other words, the local pheromone evaporation takes places in this step. This pheromone update takes place for each ant that constructs a solution as follows:

$$\tau_{ij} = (1 - \rho)\,\tau_{ij} + \frac{\rho}{BL_{nn}} \tag{1.6}$$

where L_{nn} is a very rough approximation of the optimal cost. In TSP, rough approximation can be assumed as the cost of solution resulting from the greedy choice of city n_i to the neighboring city n_j until the construction of a complete solution.

In (DORIGO and GAMBARDELLA, 1997) the following parameter values are suggested for the ACS:

$$\rho = 0.1, \beta = 2$$
$$q_0 = 0.9, \Upsilon = 0.1$$

1.2.2 EigenAnt

The EigenAnt algorithm was first introduced in (JAYADEVA *et al.*, 2013) as an algorithm that finds the shortest path between two nodes, with a proven convergence to the optimal path as follows:

1-Solution Construction:

Unlike ACO that is a population based algorithm, EigenAnt only uses one ant at each iteration. In the problem of finding the shortest path between two nodes, the solution construction procedure consists of selecting one path among O outgoing paths through RWS. The transition probability function for the EigenAnt selection phase is as follows:

$$P_{ij} = \frac{(\tau_{ij})}{\sum_{l=1}^{O}(\tau_{il})} \tag{1.7}$$

Apart from the lack of heuristic function, the pheromone amplification parameter α is also not used in the EigenAnt transition probability function in Eq. (1.7).

2- Cost Evaluation:

Cost evaluation for the constructed solution for the problem of finding the shortest path between two nodes is easy: each edge length $L = d_{ij}$ is the solution cost.

3-Pheromone Update:

The major difference between EigenAnt and ACO is in this phase in which EigenAnt uses a specific dynamical model to update the pheromone concentration trails which leads to a proof of convergence to the shortest path between two nodes via nonlinear perturbation theory (JAYADEVA *et al.*, 2013).

The model used to update the pheromone trails for the problem of finding the shortest path between two nodes is as follows:

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + (Q/L_{ij})\,\frac{\tau_{ij}(t)}{\sum_{l=1}^{O}\tau_{ij}(t)} \tag{1.8}$$

EigenAnt is the same as ACS in the sense of using local pheromone evaporation ρ (pheromone removal).

1.3 Structure of the Thesis

In Chapter 2, we improve the EigenAnt algorithm by adding two pheromone amplification parameters to it: one in the transition probability function and the other in the pheromone update dynamic. We elaborate the implementation of EigenAnt on the analytical model, applied to N-node BCP, with stability and parameter impact analysis which results in the proposal of the IEigenAnt algorithm. We explain the advantages of IEigenAnt in tuning stability and speed of convergence independently, whereas in SACO, the choice must be made between, either stability or faster convergence. Moreover, an extension of IEigenAnt, with promising applications to K shortest paths problem or sorting networks, is proposed for the basic problem of finding the K shortest paths between two nodes.

In Chapter 3, we investigate the performance of IEigenAnt application to RN by doing a comparative analysis with the EigenAnt and ACO algorithms. Besides, an empirical parameter analysis is done for IEigenAnt in order to verify the conclusions of the Chapter 2 considering the fact that RN is an extended form of N-node BCP with number of edges more than two at each layer. Moreover, the performance of IEigenAnt is experimented on MKP modeled as an N-node BCP with constraints in order to investigate the extent to which the results of Chapter 2 remain relevant in this new setting.

In Chapter 4, the application of IEigenAnt to track the optimal solutions dealing DCOPs is investigated. The advantage of solving DRN with IEigenAnt is also discussed. Finally, DMKP problem in which its constraints, constraint capacity and problem instances change over time is solved by IEigenAnt.

In Chapter 5, the contributions and conclusions of the thesis are given. In closing,

a path to future research in the area of EigenAnt algorithms is suggested.

.

Chapter 2

Improvement and Extension of the EigenAnt Algorithm

In this chapter we Improve EigenAnt algorithm by adding two parameters to it in order to control the stability and convergence behavior of the algorithm separately. Then, we extend the IEigenAnt to a sorting algorithm that finds the K shortest path between two nodes.

2.1 Improved EigenAnt Algorithm

2.1.1 Motivations

Theoretical development accompanying the design of algorithms is desirable because the long-term behavior of an algorithm and the influence of certain parameters in the algorithm can be predicted (IACOPINO and PALMER, 2012). In the area of ACO algorithms, researchers have developed some theoretical approaches. In (BIRATTARI *et al.*, 2000), an analytical model called ant programming is proposed for ACO algorithms in order to analyze the convergence of ACO through optimal control theory. In (GUTJAHR, 2000, 2002; STÜTZLE and DORIGO, 2002), the probability of convergence to the optimal solution in ACO algorithm is analyzed. Later, in (GUTJAHR, 2006), a formal framework for theoretical investigation of ACO is developed through demonstrating a limit theorem for a system of differential equations which constitute its dynamical skeleton. In (IACOPINO and PALMER, 2012), a complete stability analysis is made for the application of SACO (DORIGO and STÜTZLE, 2001) algorithm to the N-node Binary Chain Problems (BCPs) through implementation of SACO on an analytical model. In addition, an analysis of the impact of pheromone amplification parameter α (Eq. (1.1)) is carried out. Such a numerical parameter impact analysis for the TSP is carried out in (MEYER, 2004).

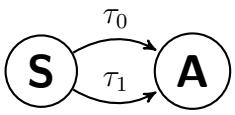

Figure 2.1: The 1-node BCP of finding the shorter of the two paths from the start node (S) to the arrival node (A). $\tau_x, x = \{0, 1\}$ denote the pheromone concentration on the path.

The paper (JAYADEVA *et al.*, 2013) proposed the EigenAnt dynamics for the shortest path problems (for paths of different lengths between a source and a destination node). In (IACOPINO and PALMER, 2012), an analysis for the SACO algorithm applied to a 2-node 2-path problem, was given and then extended to a N-node 2-path BCP. Their proposal involves an exploration phase, also known as a path selection phase, which uses a transition probability to choose a path to move from one node to the next. This probability is calculated using the pheromone concentration ratio as well as a pheromone amplification parameter. The path selection phase is followed by a pheromone update phase, for which the SACO model is used. In this Chapter, we extend the stability analysis of EigenAnt algorithm to the larger problems modeled as the N-node BCP. In contrast with the probability transition function of the original EigenAnt in (Eq. 1.7), which lacks the pheromone amplification parameter α, we consider an improved version of EigenAnt with the parameter α in order to apply the parameter impact analysis used in (IACOPINO and PALMER, 2012).

2.1.2 1-node BCP

We begin the analysis of EigenAnt application to the BCPs for the smallest version of 1-node BCP (Fig. 2.1). In fact, the 1-node BCP is the simplest limited case of the general problem with only two edges that searches for the shortest path between two nodes, and to which we can apply the general convergence proof to the shortest path was given in (JAYADEVA *et al.*, 2013). We consider this simplest problem of 1-node binary chain in order to perceive the correspondence between the results of (IACOPINO and PALMER, 2012) and those of (JAYADEVA *et al.*, 2013) in order to use the former to elucidate and extend the results of the latter paper to our applications.

Formation of the Analytical Model for SACO

In the following, we elaborate the procedure of implementing SACO on the analytical model done in (IACOPINO and PALMER, 2012). As explained in Chapter 1, SACO algorithm is the AS algorithm without any heuristics (Eq. (1.4)).

The global behavior of the system can be described using the statistical physics idea of ensemble averaging over a large number of copies or instances of the system (ensemble), each of which represents a possible system state. This procedure applied to Eqs. (1.2, 1.3 and 1.4) leads to the deterministic difference equation below:

$$\begin{aligned} \tau_x(t+1) = (1-\rho)\,\tau_x(t) + \triangle\tau\left(\frac{\tau_x^\alpha(t)}{\tau_0^\alpha(t)+\tau_1^\alpha(t)}\right) \\ \left(\frac{\tau_x^\alpha(t)}{\tau_0^\alpha(t)+\tau_1^\alpha(t)}\right) \in [0 \quad 1] \\ x = \{0,1\} \end{aligned} \tag{2.1}$$

Denoting $P_x = \left(\frac{\tau_x^\alpha(t)}{\tau_0^\alpha(t)+\tau_1^\alpha(t)}\right), x \in \{0,1\}$, we pass from a discrete representation to a continuous one:

$$\begin{cases} \frac{\mathrm{d}\tau_0}{\mathrm{dt}} = -\rho\tau_0 + C_0 P_0 \\ \frac{\mathrm{d}\tau_1}{\mathrm{dt}} = -\rho\tau_1 + C_1 P_1 \end{cases} \tag{2.2}$$

The pheromone deposition coefficients C_0 and $C_1 = \kappa C_0$ represent the $\triangle\tau$ in Eq. (2.1) associated with the related paths. It should be mentioned that $\kappa = \frac{C_1}{C_0} = \frac{L_0}{L_1}$ is referred to as the deposition parameter. In (IACOPINO and PALMER, 2012), Eq. (2.2) is referred to as the analytical model for a 1-node BCP.

EigenAnt Algorithm Mapped to the Analytical Model of (IACOPINO and PALMER, 2012)

The EigenAnt algorithm is similar with SACO in not using any heuristic at its probability transition function (Eq. 1.7). In contrast with SACO, the evaporation function in EigenAnt is done locally- in other words, it is a pheromone removal of the selected edge. Moreover, EigenAnt does not use pheromone amplification parameter α in its transition probability function. Hence, we assume an improved version of EigenAnt that includes the parameter α in its probability transition function in order to be the same as SACO in this sense. Similar to SACO, the selection phase of IEigenAnt is merged with its pheromone update phase through the ensemble hypothesis. However, the evaporation function in the IEigenAnt is done locally for the selected edge x that causes the merge of the whole pheromone update phase, consisting both the evaporation and deposition functions, with the selection phase. Therefore, the following analytical model is generated for the IEigenAnt:

$$\tau_x(t+1) = \left((1-\rho)\,\tau_x(t) + \frac{Q}{L_x}\left(\frac{\tau_x(t)}{\tau_0(t)+\tau_1(t)}\right)\right)\left(\frac{\tau_x^{\alpha}(t)}{\tau_0^{\alpha}(t)+\tau_1^{\alpha}(t)}\right) \tag{2.3}$$

It can be noticed in (Eq. 2.3) that the left term of the equation still lacks the parameter α pertains to the pheromone update dynamic of the EigenAnt (Eq. 1.8). Thus, IEigenAnt algorithm is further improved defining the following pheromone update dynamic:

$$\tau_x(t+1) = (1-\rho)\,\tau_x(t) + (Q/L_x)\,\frac{\tau_x^{\alpha}(t)}{\tau_0^{\alpha}(t)+\tau_1^{\alpha}(t)} \tag{2.4}$$

In order to distinguish between α in the transition probability function of the IEigenAnt, and the deposition function of the pheromone update dynamic, we denote the first α_1 and the latter α_2 . As a result, the analytical model for IEigenAnt is as follows:

$$\tau_x(t+1) = \left((1-\rho)\,\tau_x(t) + \frac{Q}{L_x}\left(\frac{\tau_x^{\alpha_2}(t)}{\tau_0^{\alpha_2}(t)+\tau_1^{\alpha_2}(t)}\right)\right)\left(\frac{\tau_x^{\alpha_1}(t)}{\tau_0^{\alpha_1}(t)+\tau_1^{\alpha_1}(t)}\right) \tag{2.5}$$

It can be noticed that the left term in Eq. (2.5) is the same as the discrete version of the analytical model in Eq. (2.1). Passing from the discrete representation to the continuous one, gives the following analytical model for EigenAnt:

$$\begin{cases} f: \frac{\mathrm{d}\tau_0}{\mathrm{dt}} = \left(-\rho\tau_0 + C_0 P_0(\alpha_2)\right) P_0(\alpha_1) \\ g: \frac{\mathrm{d}\tau_1}{\mathrm{dt}} = \left(-\rho\tau_1 + C_1 P_1(\alpha_2)\right) P_1(\alpha_1) \end{cases} \tag{2.6}$$

Discussion

In (IACOPINO and PALMER, 2012), it is concluded from the stability analysis of the equilibrium points of the dynamic in Eq. (2.2) that with $\alpha = 1$ the system converges to the shortest path. However, it is also concluded that the speed of convergence is the slowest with $\alpha = 1$. Therefore, choosing $\alpha = 1$ in SACO application to the 1-node BCPs guarantees the convergence to the shortest path at the expense of increased convergence time. It is also concluded that with $\alpha > 1$ the system might converge to a non-optimal path, although the speed of convergence is faster than in the case $\alpha = 1$. Finally, it is concluded that the system does not demonstrate convergence behavior with $\alpha < 1$.

The analytical model of IEigenAnt (Eq. (2.6)) has the same equilibrium points as the SACO (Eq. (2.2)). The only distinction between the two analytical models

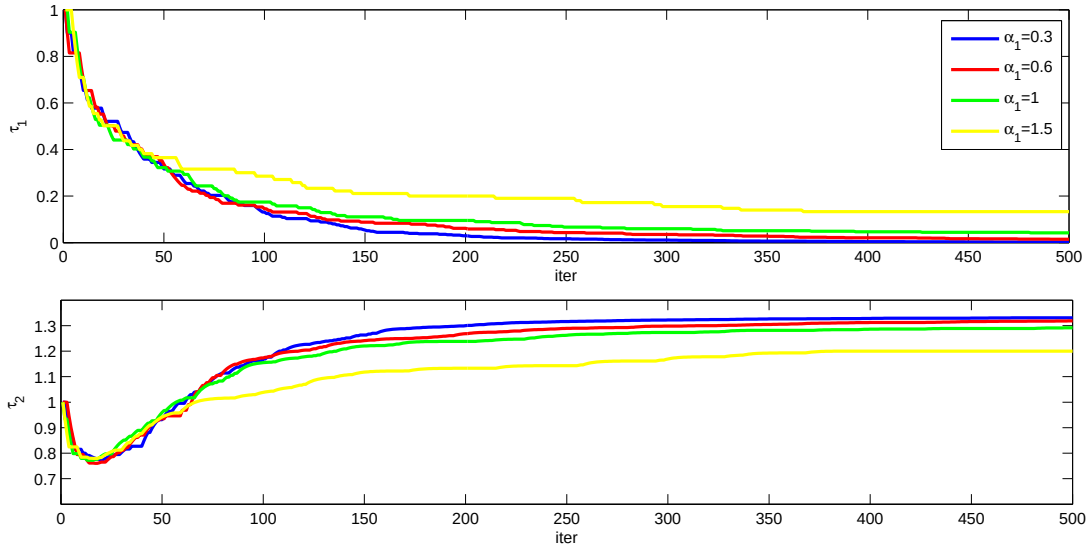

Figure 2.2: Influence of parameter α_1 on the speed of convergence in IEigenAnt

is in the additional factor $P(\alpha_1)$ in Eq. (2.6) that affects the speed of convergence without affecting the nature of the stability of the equilibrium point. A larger value of $P(\alpha_1)$ causes faster convergence. The parameter α_1 in $P(\alpha_1)$, pertains to the transition probability function, has an inverse relation with the value of $0 \leq P \leq 1$. Since α_2 can be chosen independently of α_1, we have an additional freedom in the design of the dynamics. Specifically, choosing $\alpha_2 = 1$ causes the pheromone trails converge to the shortest path in 1-node BCPs which has already been proved in (JAYADEVA *et al.*, 2013). Similar to SACO, $\alpha_2 > 1$ causes the pheromone trails converge to the non-optimal path and $\alpha_2 < 1$ does not demonstrate convergence behavior in the pheromone trails.

In order to demonstrate such a result about the speed of convergence we apply IEigenAnt with the pheromone removal parameter of $\rho = 0.2$ to a 1-node binary chain problem with the two edges of length 5 and 3.75. The pheromone amplification parameter in the pheromone update phase of IEigenAnt is $\alpha_2 = 1$ that guarantees the convergence to the shortest path. Fig. 2.2 illustrates that the speed of convergence has direct relation with value of α_1 as the pheromone trail corresponding with the non-optimal edge converges to zero with the smallest value of $\alpha_1 = 0.3$ after 300 iterations while the slowest convergence with $\alpha_1 = 1.5$ is depicted as it has not yet converged to zero until 500 number of iterations.

We now apply IEigenAnt to the general problem of finding the shortest path between two nodes with four edges of length $\{5, 3, 7, 2\}$ with $\alpha_2 = 1$ and $\rho = 0.2$. Fig. 2.3 demonstrates the inverse relation of α_1 with the speed of convergence.

In order to emphasize the benefit of using IEigenAnt in having the freedom of choice for the values of parameter α_1 pertaining to the probability transition function

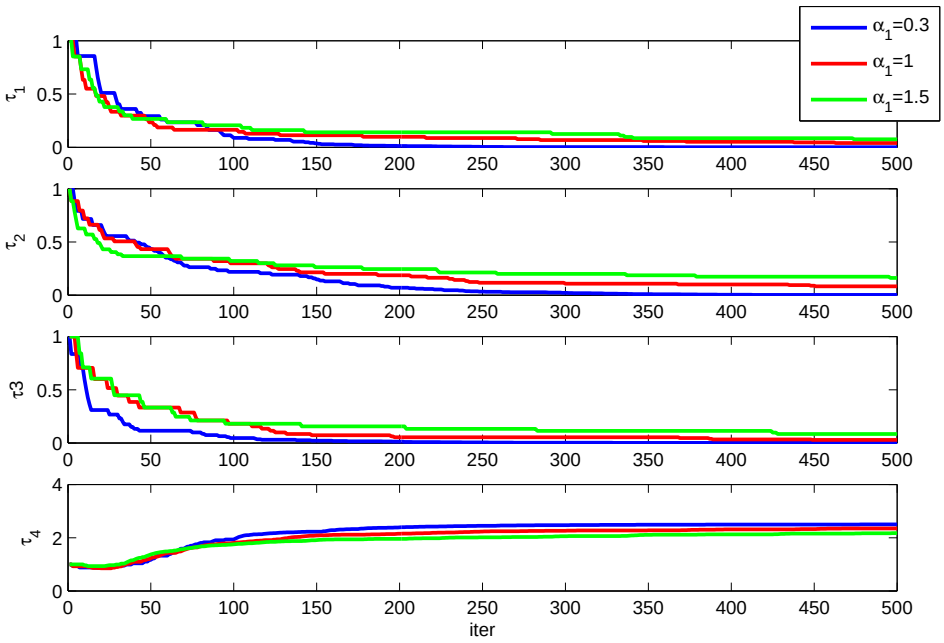

Figure 2.3: Speed of convergence analysis in finding the shortest path between two nodes by IEigenAnt

and α_2 pertaining to the pheromone update dynamic, we apply IEigenAnt to the previous problem with $\alpha_1 = \alpha_2 = \alpha = 0.3$. Fig. 2.4 demonstrates that pheromone concentrations converge to values that do not suggest a clear choice of a specific edge, emphasizing that in order to maintain stability of the desired equilibrium and also attain a fast speed of convergence it is necessary to make the choices $\alpha_1 < 1$ and $\alpha_2 = 1$.

2.1.3 2-Node BCP

Fig. 2.5 depicts a 2-node BCP model. Fig. 2.5 shows that every edge (in the 2-node BCP graph) belongs to two distinct paths out of four possible paths from node S to node B, passing through node A. Pheromone trails correspond with each edge in Fig.2.5.

SACO Implementation on the Analytical Model

The analytical model proposed in (IACOPINO and PALMER, 2012) through the implementation of SACO algorithm applied to 2-node BCP is as follows:

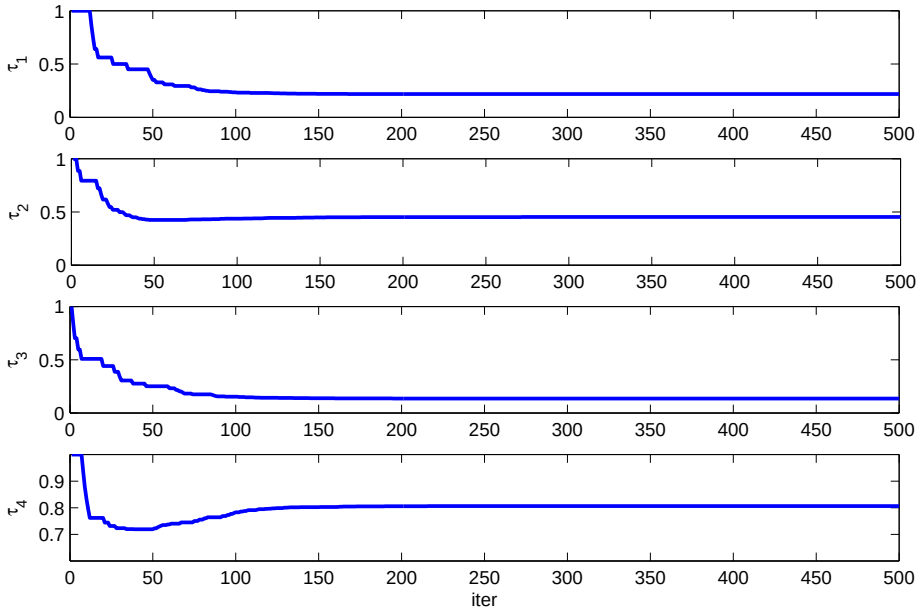

Figure 2.4: IEigenAnt with $\alpha_1 = \alpha_2 = \alpha = 0.3$ showing that the choice of $\alpha_2 < 1$ might result in pheromone concentrations converging to values such that no edge emerges as a clear choice (in terms of having a larger pheromone concentration than the other edges)

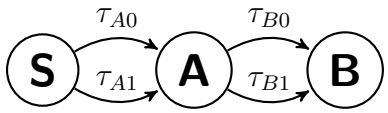

Figure 2.5: The 2-node BCP

$$\begin{cases} \frac{\mathrm{d}\tau_{A0}}{\mathrm{dt}} = -\rho\tau_{A_o} + (C_{00}P_{B0} + C_{01}P_{B1})\,P_{A0} \\ \frac{\mathrm{d}\tau_{A1}}{\mathrm{dt}} = -\rho\tau_{A1} + (C_{10}P_{B0} + C_{11}P_{B1})\,P_{A1} \\ \frac{\mathrm{d}\tau_{B0}}{\mathrm{dt}} = -\rho\tau_{B0} + (C_{00}P_{A0} + C_{10}P_{A1})\,P_{B0} \\ \frac{\mathrm{d}\tau_{B1}}{\mathrm{dt}} = -\rho\tau_{B1} + (C_{01}P_{A0} + C_{11}P_{A1})\,P_{B1} \end{cases} \tag{2.7}$$

Similar to the 1-node BCP, the selection phase is only merged with the deposition function of each pheromone trail. For example, the pheromone trail τ_{Ao} in the first line of Eq. (2.7) can be a member of a path including $\tau_{A0}\tau_{B0}$ or $\tau_{A0}\tau_{B1}$ where two different pheromone deposition coefficients of C_{00} and C_{01} (depending on the selected total path length) are considered for them, respectively.

The probability of choosing each path relating to the deposition parameters C_{00} and C_{01} ($P_{A0}P_{B0}$ and $P_{A0}P_{B1}$) is multiplied by its respective deposition parameter.

IEigenAnt Algorithm Applied to 2-node BCPs

The selection phase of IEigenAnt applied to the 2-node BCP consists of two steps. First, selection between edges relating to pheromone trails of τ_{A0} and τ_{A1}. Second, the selection between τ_{B0} and τ_{B1}. It should be mentioned that the transition probability function is dependent on the parameter α_1 ($P(\alpha_1)$) .

Furthermore, two distinct pheromone updates occur at each layer in Fig. 2.5. The pheromone update of IEigenAnt applied to the 2-node binary chain at each layer has the following dynamic:

$$\begin{cases} \tau_{Ax}(t+1) = (1-\rho)\,\tau_{Ax}(t) + \frac{Q}{L_r} P_{Ax}(\alpha_2) \\ \\ \tau_{Bx}(t+1) = (1-\rho)\,\tau_{Bx}(t) + \frac{Q}{L_r} P_{Bx}(\alpha_2) \end{cases} \tag{2.8}$$

where L_r is the total length of the path r selected out of possible four paths and $x \in \{0,1\}$. Moreover, the local evaporation is used in Eq. (2.8). Solution convergence analysis considers whether the pheromone trail equations τ_{Ax} and τ_{Bx} (Eq. (2.8)) converge to the edges of a path with the minimum value of L_s.

Deriving the Analytical Model of IEigenAnt Applied to the 2-node BCPs

Since the evaporation function takes place locally in IEigenAnt, the probability of selecting each pheromone trail at its layer is multiplied by the corresponding evaporation term. Therefore, through the ensemble hypothesis, we obtain the following analytical continuous-time model for the IEigenAnt:

$$\begin{cases} \frac{\mathrm{d}\tau_{A0}}{\mathrm{dt}} = P_{A0}(\alpha_1)\left(-\rho\tau_{A_0} + \left(C_{00}P_{B0}(\alpha_1) + C_{01}P_{B1}(\alpha_1)\right)P_{A0}(\alpha_2)\right) \\ \frac{\mathrm{d}\tau_{A1}}{\mathrm{dt}} = P_{A1}(\alpha_1)\left(-\rho\tau_{A1} + \left(C_{10}P_{B0}(\alpha_1) + C_{11}P_{B1}(\alpha_1)\right)P_{A1}(\alpha_2)\right) \\ \frac{\mathrm{d}\tau_{B0}}{\mathrm{dt}} = P_{B0}(\alpha_1)\left(-\rho\tau_{B0} + \left(C_{00}P_{A0}(\alpha_1) + C_{10}P_{A1}(\alpha_1)\right)P_{B0}(\alpha_2)\right) \\ \frac{\mathrm{d}\tau_{B1}}{\mathrm{dt}} = P_{B1}(\alpha_1)\left(-\rho\tau_{B1} + \left(C_{01}P_{A0}(\alpha_1) + C_{11}P_{A1}(\alpha_1)\right)P_{B1}(\alpha_2)\right) \end{cases} \tag{2.9}$$

Similar to the 1-node BCP case, the factorized terms $P(\alpha_1)$ demonstrate the effect of the parameter α_1 from the transition probability function on the speed of convergence. However, there are differences in the stability analysis of the analytical model in Eq. (2.9) and the analytical model of SACO in Eq. (2.7) due to the P functions dependent on both α_1 and α_2 ($P(\alpha_1)$ and $P(\alpha_2)$). It remains to show that only the parameter α_2 has effect on the equilibrium points of the system so that parameter selection of α_1 can be made with a view only to speed of convergence. In order to observe the effect of each parameter α_1 and α_2 in Eq. (2.9), we modify the mathematical procedure used in (IACOPINO and PALMER, 2012) for finding the

equilibrium points of the corresponding analytical model.

At the equilibrium points of Eq. (2.9), the following holds:

$$\begin{cases} \rho\tau_{A_0} = \left(C_{00}P_{B0}\left(\alpha_1\right) + C_{01}P_{B1}\left(\alpha_1\right)\right)P_{A0}\left(\alpha_2\right) \\ \rho\tau_{A1} = \left(C_{10}P_{B0}\left(\alpha_1\right) + C_{11}P_{B1}\left(\alpha_1\right)\right)P_{A1}\left(\alpha_2\right) \\ \rho\tau_{B0} = \left(C_{00}P_{A0}\left(\alpha_1\right) + C_{10}P_{A1}\left(\alpha_1\right)\right)P_{B0}\left(\alpha_2\right) \\ \rho\tau_{B1} = \left(C_{01}P_{A0}\left(\alpha_1\right) + C_{11}P_{A1}\left(\alpha_1\right)\right)P_{B1}\left(\alpha_2\right) \end{cases} \tag{2.10}$$

Since $P_{A0}\left(\alpha_2\right) + P_{A1}\left(\alpha_2\right) = 1$ and $P_{B0}\left(\alpha_2\right) + P_{B1}\left(\alpha_2\right) = 1$,

$$\begin{cases} \rho\left(\frac{\tau_{A0}}{C_{00}P_{B0}(\alpha_1)+C_{01}P_{B1}(\alpha_1)} + \frac{\tau_{A1}}{C_{10}P_{B0}(\alpha_1)+C_{11}P_{B1}(\alpha_1)}\right) = 1 \\ \rho\left(\frac{\tau_{B0}}{C_{00}P_{A0}(\alpha_1)+C_{10}P_{A1}(\alpha_1)} + \frac{\tau_{B1}}{C_{01}P_{A0}(\alpha_1)+C_{11}P_{A1}(\alpha_1)}\right) = 1 \end{cases} \tag{2.11}$$

Starting from the first equations of the system in Eq. (2.10), relative to the node A, since $P_{Ax}\left(\alpha_2\right) = \frac{\tau_{Ax}^{\alpha_2}}{\tau_{A0}^{\alpha_2}+\tau_{A1}^{\alpha_2}}$:

$$\begin{cases} \rho\left(\tau_{A0}^{\alpha_2} + \tau_{A1}^{\alpha_2}\right) = \tau_{A0}^{\alpha_2-1}\left(C_{00}P_{B0}\left(\alpha_1\right) + C_{01}P_{B1}\left(\alpha_1\right)\right) \\ \rho\left(\tau_{A0}^{\alpha_2} + \tau_{A1}^{\alpha_2}\right) = \tau_{A1}^{\alpha_2-1}\left(C_{10}P_{B0}\left(\alpha_1\right) + C_{11}P_{B1}\left(\alpha_1\right)\right) \end{cases} \tag{2.12}$$

Dividing the second formula by the first, we would have:

$$1 = \left(\frac{\tau_{A1}}{\tau_{A0}}\right)^{\alpha_2-1}\left(\frac{C_{10}P_{B0}\left(\alpha_1\right) + C_{11}P_{B1}\left(\alpha_1\right)}{C_{00}P_{B0}\left(\alpha_1\right) + C_{01}P_{B1}\left(\alpha_1\right)}\right) \tag{2.13}$$

$$\Rightarrow \tau_{A0} = \tau_{A1}\left(\frac{C_{10}P_{B0}\left(\alpha_1\right) + C_{11}P_{B1}\left(\alpha_1\right)}{C_{00}P_{B0}\left(\alpha_1\right) + C_{01}P_{B1}\left(\alpha_1\right)}\right)^{\frac{1}{\alpha_2-1}} \tag{2.14}$$

Considering $m_A = \left(\frac{C_{10}P_{B0}(\alpha_1)+C_{11}P_{B1}(\alpha_1)}{C_{00}P_{B0}(\alpha_1)+C_{01}P_{B1}(\alpha_1)}\right)$ and $\gamma = \frac{\alpha_2}{\alpha_2-1}$, we would have a straight line through the origin in the (τ_{A0}, τ_{A1}) plane as $\tau_{A0} = \tau_{A1}m_A^{\gamma-1}$. Substituting m_A in the first equation of Eq. (2.11), we would have:

$$\begin{aligned} \frac{\tau_{A0}}{C_{00}P_{B0}\left(\alpha_1\right) + C_{01}P_{B1}\left(\alpha_1\right)} + \frac{\tau_{A1}}{m_A\left(C_{00}P_{B0}\left(\alpha_1\right) + C_{01}P_{B1}\left(\alpha_1\right)\right)} &= \frac{1}{\rho} \\ \Rightarrow \tau_{A0} = \frac{C_{00}P_{B0}\left(\alpha_1\right) + C_{01}P_{B1}\left(\alpha_1\right)}{\rho} - \frac{\tau_{A1}}{m_A} \end{aligned} \tag{2.15}$$

This is a straight line of slope $-\frac{1}{m_A}$ in the (τ_{A0}, τ_{A1}) plane and it is valid when both $\tau \neq 0$. The intersection of these two lines is a unique point in this plane, but it depends on its location in the (τ_{B0}, τ_{B1}) plane.

Similarly, for the node B:

$$\begin{cases} \tau_{B0} = \frac{C_{00}P_{A0}(\alpha_1)+C_{10}P_{A1}(\alpha_1)}{\rho} - \frac{\tau_{B1}}{m_B} \\ \tau_{B0} = \tau_{B1} m_B^{\gamma-1} \text{ Let } m_B = \left(\frac{C_{01}P_{A0}(\alpha_1)+C_{11}P_{A1}(\alpha_1)}{C_{00}P_{A0}(\alpha_1)+C_{10}P_{A1}(\alpha_1)}\right) \end{cases} \tag{2.16}$$

Imposing one of the $\tau = 0$ (Eq. (2.11)) for each couple of equations gives directly four equilibrium points, one for each combination:

$$S_{00} = \begin{cases} \tau_{A0} = \frac{C_{00}}{\rho} \\ \tau_{A1} = 0 \\ \tau_{B0} = \frac{C_{00}}{\rho} \\ \tau_{B1} = 0 \end{cases} \quad S_{01} = \begin{cases} \tau_{A0} = \frac{C_{01}}{\rho} \\ \tau_{A1} = 0 \\ \tau_{B0} = 0 \\ \tau_{B1} = \frac{C_{01}}{\rho} \end{cases} \quad S_{10} = \begin{cases} \tau_{A0} = 0 \\ \tau_{A1} = \frac{C_{10}}{\rho} \\ \tau_{B0} = \frac{C_{10}}{\rho} \\ \tau_{B1} = 0 \end{cases} \quad S_{11} = \begin{cases} \tau_{A0} = 0 \\ \tau_{A1} = \frac{C_{11}}{\rho} \\ \tau_{B0} = 0 \\ \tau_{B1} = \frac{C_{11}}{\rho} \end{cases} \tag{2.17}$$

If we impose one $\tau = 0$ for only one couple of equations, the other couple of equations can be simplified; e.g, for $\tau_{A0} = 0$, the system can be written as:

$$\begin{cases} \frac{\mathrm{d}\tau_{B0}}{\mathrm{dt}} = -\rho\tau_{B0} + C_{10}P_{B0}(\alpha_2) \\ \frac{\mathrm{d}\tau_{B1}}{\mathrm{dt}} = -\rho\tau_{B1} + C_{11}P_{B1}(\alpha_2) \end{cases} \tag{2.18}$$

This system has an equilibrium point lying on the side connecting two vertices solutions presented above (Eq. (2.17)), in this case it lies on the side between S_{11} and S_{10}. Imposing the condition $\tau = 0$ for all the τ gives the following four equilibrium points:

$$S_{20} = \begin{cases} \tau_{A0} = \frac{C_{00}}{\rho}\frac{\left(\frac{C_{10}}{C_{00}}\right)^\gamma}{1+\left(\frac{C_{10}}{C_{00}}\right)^\gamma} \\ \tau_{A1} = \frac{C_{00}}{\rho}\frac{\left(\frac{C_{10}}{C_{00}}\right)}{1+\left(\frac{C_{10}}{C_{00}}\right)^\gamma} \\ \tau_{B0} = \frac{C_{00}}{\rho}\frac{\left(\frac{C_{10}}{C_{00}}\right)\left(\frac{C_{10}}{C_{00}}\right)^\gamma}{1+\left(\frac{C_{10}}{C_{00}}\right)^\gamma} \\ \tau_{B1} = 0 \end{cases} \quad S_{21} = \begin{cases} \tau_{A0} = \frac{C_{01}}{\rho}\frac{\left(\frac{C_{11}}{C_{01}}\right)^\gamma}{1+\left(\frac{C_{11}}{C_{01}}\right)^\gamma} \\ \tau_{A1} = \frac{C_{01}}{\rho}\frac{\left(\frac{C_{11}}{C_{01}}\right)}{1+\left(\frac{C_{11}}{C_{01}}\right)^\gamma} \\ \tau_{B0} = 0 \\ \tau_{B1} = \frac{C_{01}}{\rho}\frac{\left(\frac{C_{11}}{C_{01}}\right)\left(\frac{C_{11}}{C_{01}}\right)^\gamma}{1+\left(\frac{C_{11}}{C_{01}}\right)^\gamma} \end{cases} \tag{2.19}$$

$$S_{02} = \begin{cases} \tau_{A0} = \frac{C_{00}}{\rho}\frac{\left(\frac{C_{01}}{C_{00}}\right)\left(\frac{C_{01}}{C_{00}}\right)^\gamma}{1+\left(\frac{C_{01}}{C_{00}}\right)^\gamma} \\ \tau_{A1} = 0 \\ \tau_{B0} = \frac{C_{00}}{\rho}\frac{\left(\frac{C_{01}}{C_{00}}\right)^\gamma}{1+\left(\frac{C_{01}}{C_{00}}\right)^\gamma} \\ \tau_{B1} = \frac{C_{00}}{\rho}\frac{\left(\frac{C_{01}}{C_{00}}\right)}{1+\left(\frac{C_{01}}{C_{00}}\right)^\gamma} \end{cases} \quad S_{12} = \begin{cases} \tau_{A0} = 0 \\ \tau_{A1} = \frac{C_{10}}{\rho}\frac{\left(\frac{C_{11}}{C_{10}}\right)\left(\frac{C_{11}}{C_{10}}\right)^\gamma}{1+\left(\frac{C_{11}}{C_{10}}\right)^\gamma} \\ \tau_{B0} = \frac{C_{10}}{\rho}\frac{\left(\frac{C_{11}}{C_{10}}\right)^\gamma}{1+\left(\frac{C_{11}}{C_{10}}\right)^\gamma} \\ \tau_{B1} = \frac{C_{10}}{\rho}\frac{\left(\frac{C_{11}}{C_{10}}\right)}{1+\left(\frac{C_{11}}{C_{10}}\right)^\gamma} \end{cases}$$

Finally, imposing that all the $\tau \neq 0$, we are interested in finding the point given

by the interception of the four lines described by the Eq. (2.14), Eq. (2.15) and Eq. (2.16):

$$S_{22} = \begin{cases} \tau_{A0} = \frac{C_{01}+C_{00}m_B^\gamma}{\rho} \frac{m_A^\gamma}{\left(1+m_B^\gamma\right)\left(1+m_A^\gamma\right)} \\ \tau_{B0} = \frac{C_{10}+C_{00}m_A^\gamma}{\rho} \frac{m_B^\gamma}{\left(1+m_B^\gamma\right)\left(1+m_A^\gamma\right)} \end{cases} \tag{2.20}$$

Once the slopes m_A and m_B are determined, we can determine a unique location given by the interception of these lines. As mentioned before the equilibrium points should all be dependent on the parameter α_2 in order to be able to implement EigenAnt on the analytical model (Eq. (2.7)) analyzed in (IACOPINO and PALMER, 2012). However, the equilibrium point S_{22} is dependent on m_A and m_B which are dependent on α_1. In (IACOPINO and PALMER, 2012), the relation between m_A and m_B is found as follows:

$$\begin{cases} m_A^\gamma = \frac{C_{11}-C_{10}m_B}{C_{00}m_B+C_{01}} \\ m_B^\gamma = \frac{C_{11}-C_{01}m_A}{C_{00}m_A+C_{10}} \end{cases} \tag{2.21}$$

In (IACOPINO and PALMER, 2012), it is concluded from the above equations that there is a unique solution m_A for a given m_B and vice versa. Moreover, it is concluded that m_A and m_B have the following range by plotting the equation pair in Eq. (2.21):

$$m_A \in \left[\frac{C_{10}}{C_{00}}, \frac{C_{11}}{C_{01}}\right] \qquad m_B \in \left[\frac{C_{01}}{C_{00}}, \frac{C_{11}}{C_{10}}\right] \tag{2.22}$$

The aforementioned conclusions imply the lumping of m_A and m_B to a deposition parameter κ which makes the analysis independent of α_1 possible.

2.1.4 N-node BCPs

SACO Implementation on the Analytical Model

The analytical model that is used to prove the convergence of SACO algorithm applied to the N-node BCPs is proposed as the following set of N pairs of equation (IACOPINO and PALMER, 2012):

$$\begin{cases} \frac{\mathrm{d}\tau_{j0}}{\mathrm{dt}} = -\rho\tau_{j0} + D_{j0} \\ \frac{\mathrm{d}\tau_{j1}}{\mathrm{dt}} = -\rho\tau_{j1} + D_{j1} \end{cases} \tag{2.23}$$

where D_{jx} describes the amount of pheromone deposited. The structure for this term is rather complex; considering the equation for the edge jx, D_{jx} can be expressed as:

$$D_{jx} = \sum_{r \in R_{jx}} C_{jx}^{r} \Phi_{jx}^{r} \tag{2.24}$$

$$\Phi_{jx}^{r} = P_{jx} \prod_{l \in H, l \neq j0, j1} P_l \tag{2.25}$$

where R_{jx} is the finite countable set of all the possible solutions containing the edge jx. The size of this set is 2^{N-1}, C_{jx}^{r} is the pheromone update coefficient for a specific path r , and Φ_{jx}^{r} indicates the transition probability for such a path. H is the finite countable set of all the possible edges forming a path, its size is $2N$, and P_{jx} is the generic transition probability:

$$P_{jx} = \frac{\tau_{jx}^{\alpha}}{\tau_{j0}^{\alpha} + \tau_{j1}^{\alpha}} \tag{2.26}$$

IEigenAnt Implementation on the Analytical Model for the N-node BCPs

Considering the local pheromone evaporation feature in IEigenAnt, we assume a SACO algorithm with local evaporation in order to implement IEigenAnt on the analytical model of the N-node BCPs:

$$\begin{cases} \frac{\mathrm{d}\tau_{j0}}{\mathrm{dt}} = (P_{j0}) \left(-\rho \tau_{j0} + \left(\sum_{r \in R_{j0}} C_{j0}^{r} \prod_{l \in H, l \neq j0, j1} P_l \right) \right) \\ \frac{\mathrm{d}\tau_{j1}}{\mathrm{dt}} = (P_{j1}) \left(-\rho \tau_{j1} + \left(\sum_{r \in R_{j1}} C_{j1}^{r} \prod_{l \in H, l \neq j0, j1} P_l \right) \right) \end{cases} \tag{2.27}$$

As can be noticed from Eq. (2.27) and Eq. (2.25), the analytical model formed for the SACO with local pheromone evaporation lacks the factored term P_{jx} in the parenthesis of Eq. (2.27). Thus, the term P_{jx} should be augmented in the parenthesis in order to have a successful implementation on the model. The term P_{jx} can be augmented through the pheromone update dynamic which results in the transformation of SACO algorithm with local evaporation to the EigenAnt algorithm. As a result, we would have the following pheromone update dynamic for an ant that goes to the layer j in the IEigenAnt algorithm applied to the N-node BCP:

$$\tau_{jx}(t+1) = (1-\rho)\,\tau_{jx}(t) + \left(\frac{Q}{L_r} \right) P_{jx}(\alpha_2) \tag{2.28}$$

Merging the above dynamic with the selection phase, the analytical model relating to the IEigenAnt algorithm is formed as follows:

$$\begin{cases} \frac{\mathrm{d}\tau_{j0}}{\mathrm{dt}} = \left(P_{j0}\left(\alpha_1\right)\right)\left(-\rho\tau_{j0} + \left(\sum_{r\in R_{j0}} C_{j0}^{r} P_{j0}\left(\alpha_2\right) \prod_{l\in H, l\neq j0,j1} P_l\left(\alpha_1\right)\right)\right) \\ \frac{\mathrm{d}\tau_{j1}}{\mathrm{dt}} = \left(P_{j1}\left(\alpha_1\right)\right)\left(-\rho\tau_{j1} + \left(\sum_{r\in R_{j1}} C_{j1}^{r} P_{j1}\left(\alpha_2\right) \prod_{l\in H, l\neq j0,j1} P_l\left(\alpha_1\right)\right)\right) \end{cases} \tag{2.29}$$

Considering $\alpha_1 = \alpha_2 = \alpha$ the same as the original EigenAnt algorithm, the analytical model in Eq. (2.29) and Eq. (2.23) have the same equilibrium points which means that stability analysis done in (IACOPINO and PALMER, 2012) for the application of SACO algorithm to the N-node BCPs is satisfied for the EigenAnt algorithm. However, there are differences between Eq. (2.29) and Eq. (2.23) when $\alpha_1 \neq \alpha_2$ for the IEigenAnt similarly to the discussion above for the 2-node binary chain. Our goal is to show that the equilibrium points of Eq. (2.29) are only dependent on α_2 so that the difference of α_1 with α_2 can be overlooked in the stability analysis which makes the stability analysis done for the implementation of SACO on the analytical model Eq. (2.23) the same as the one for the IEigenAnt in Eq. (2.29).

In (IACOPINO and PALMER, 2012), it is demonstrated that the equilibrium points of Eq. (2.23) have the same format of the equilibrium points in the analytical model (Eq. (2.7)) relating to the application of SACO to the 2-node BCP. For the IEigenAnt, there is an N-cube space where at each surface Su defined for the related pair x there are equilibrium points at the vertices with the format of Eq. (2.17), equilibrium points connecting the vertices on the sides of the surface with the format of Eq. (2.19) and the midpoint equilibrium point with the format of Eq. (2.20). As mentioned before, the mid-point equilibrium point depends on the $m_l\left(\alpha_1\right)$. A theorem is proved in (IACOPINO and PALMER, 2012) for midpoint equilibrium points for N-node BCP. We rewrite the theorem as follows:

Theorem 1. *Given a generic mid-point and the surface Su where it lies, the coordinates of this mid-point along any pheromone variable pair jx defining the surface Su are given by the following equations:*

$$Pair \quad jx : \begin{cases} \tau_{j0} = \frac{C_{j0}}{\rho}\left(\frac{m_j^{\gamma}}{1+m_j^{\gamma}}\right) \\ \tau_{j1} = \frac{C_{j1}}{\rho}\left(\frac{m_j}{1+m_j^{\gamma}}\right) \end{cases} \quad m_j = \frac{C_{j1}}{C_{j0}} \qquad \gamma = \frac{\alpha_2}{\alpha_2 - 1} \tag{2.30}$$

As it can be noticed from Theorem. 1, m is lumped to a pheromone deposition parameter in (IACOPINO and PALMER, 2012); hence, the position of mid-point equilibrium is only dependent on α_2. Therefore, the stability analysis that used in (IACOPINO and PALMER, 2012) for the application of SACO to the N-node BCPs is also true for the application of IEigenAnt algorithm. The impact of parameter α_1

on the speed of convergence is the same as the 1-node binary problem case as for the IEigenAnt application due to the $P_{jx}(\alpha_1)$ factor in Eq. (2.29).

Conclusions

The stability analysis done in (IACOPINO and PALMER, 2012) for the application of SACO algorithm to the N-node BCPs with different values of α are hold for different values of α_2 in the application of IEigenAnt to N-node binary chain problems. On the other hand, there is a direct effect of parameter α_1 on the speed of convergence of the analytical model of the IEigenAnt applied to N-node binary chain problems. We summarize our conclusions as follows for the application of IEigenAnt to N-node BCPs with different values of α_1 and α_2:

1. For $\alpha_1 = 1$ and $\alpha_2 = 1$, the algorithm is the original EigenAnt algorithm. The system is driven towards the local optimal solution, represented by vertex, but its velocity is low.

2. For $\alpha_1 < 1$ and $\alpha_2 = 1$, the system converges to the local optimal solutions with a faster speed than the original EigenAnt.

3. For $\alpha_1 > 1$ and $\alpha_2 = 1$, the system converges to the local optimal solution with a lower speed than the original EigenAnt.

4. For $\alpha_1 < 1$ and $\alpha_2 < 1$, the system shows only one stable point, lying inside the N-cube, representing uniform distribution of pheromones on the BCP. In the terms of problem solutions, the system does not converge, but fluctuates around this stable point performing continuous explorations for new vertices. The speed of explorations is also high.

5. For $\alpha_1 = 1$ and $\alpha_2 < 1$, the system shows an exploratory (non-convergent) behavior with a lower speed than the previous case.

6. For $\alpha_1 > 1$ and $\alpha_2 < 1$, the system shows an exploratory behavior with the lowest speed.

7. For $\alpha_1 < 1$ and $\alpha_2 > 1$, all the possible problem solutions are stable points. The speed of convergence is also very high which makes the possibility of premature convergence large due to the greedy behavior of the algorithm.

8. For $\alpha_1 = 1$ and $\alpha_2 > 1$, the system still has the risk of premature convergence but with a lower possibility than the previous case due the the reduction in the speed of convergence.

9. For $\alpha_1 > 1$ and $\alpha_2 > 1$, all the possible solutions are stable points; however, the convergence speed is at the lowest speed which might render some exploration behavior.

2.2 Extended IEigenAnt

In this section, we propose an extension of IEigenAnt entitled Sorting IEigenAnt. SIEigenAnt demonstrates a convergence behavior in which each pheromone trail converges to a value proportional the corresponding path length. Moreover, we illustrate the capability of SIEigenAnt in finding K shortest paths between two nodes in both the stationary and dynamic problem cases. It should be mentioned that an extensions of EigenAnt entitled M-unit EigenAnt is also proposed for finding the K shortest paths between two nodes in (SHAH, 2011; SHAH *et al.*, 2011). However, they used K pheromone trails for each edge which means that M-unit EigenAnt uses a great amount of time and memory attributed to $K \times O$ number of pheromone trails for a simple problem of optimization between two nodes, whereas SIEigenAnt only requires O number of pheromone trails.

2.2.1 SIEigenAnt Algorithm

The solution construction phase of SIEigenAnt algorithm is the same of IEigenAnt. The difference is in pheromone update phase where the pheromone evaporation parameter is as follows:

$$\rho\left(P\left(\alpha_2\right)\right) = P\left(\alpha_2\right)^2 \delta \tag{2.31}$$

As a result, the following pheromone update dynamic is assumed for the selected path j:

$$\tau_j\left(t+1\right) = \left(1 - P_j\left(\alpha_2\right)^2 \delta\right) \tau_j\left(t\right) + \frac{\beta\delta}{L_j} P_j\left(\alpha_2\right) \tag{2.32}$$

where δ is a scaling factor which is also added to the deposition term together with the evaporation term. The following constraint for the parameter δ in Eq. (2.32) should be maintained:

$$\left(1 - P_j\left(\alpha_2\right)^2 \delta\right) \geq 0 \mapsto P_j\left(\alpha_2\right)^2 \delta \leq 1$$
$$\Rightarrow \delta \leq \frac{1}{P_j\left(\alpha_2\right)^2} \tag{2.33}$$

Maintaining the condition Eq. (2.33) is a challenge due to the variable value of $0 \leq P_j\left(\alpha_2\right) \leq 1$ over time. We assume a constant value for δ in which $\delta > 1$ renders the risk of negative pheromone trail value while $\delta \leq 1$ is free of such a risk but with a very slow speed of convergence. In order to have an algorithm that has the fast speed of convergence and does not result in negative pheromone trails, we propose the use of minimum procedure in SIEigenAnt algorithm. The minimum procedure is similar to the Max-Min algorithm proposed in (STÜTZLE and HOOS, 2000). However, the procedure only checks the violation of minimum pheromone trail value. Whenever the minimum trail value is violated the corresponding pheromone trail value is set to the minimum limit value and δ is decreased by a constant value.

2.2.2 Experimental Results

In this subsection, we illustrate the performance of SIEigentAnt algorithm dealing with four types of problem to which it is applicable. First, we illustrate the performance of SIEigenAnt algorithm in finding the K shortest paths between two nodes. Second, we illustrate the capability of SIEigenAnt in finding the K shortest paths between two nodes in the so-called dynamic case, in which these paths change over time. Third, we illustrate the performance of SIEigenAnt algorithm in solving the K shortest path problem when it has paths with equal length and lengths with small differences. Finally, we illustrate the application of SIEigenAnt to the dynamic version of the previous problem. In all the experiments, we set $\delta = 10$, $\alpha_1 = 0.3$, $\alpha_2 = 1$, $\beta = 1$ and we decrease $\delta = \delta - 0.5$ whenever it violates the minimum pheromone value $\tau_{min} = 10^{-3}$. Initial pheromone trail values are set to their related path lengths, as suggested in (SHAH *et al.*, 2011), in order to make the problem more difficult to solve.

We apply SIEigenAnt algorithm to the problem with the following ten edges:

$$L = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Fig. 2.6 demonstrates the convergence of the pheromone trails. As it can be noticed, the trail initiating from the smallest value terminates at the highest value for the edge $L = 1$. It continues with the same inverse relation between initial value to the final value for other paths from $L = 2$ to $L = 10$ that illustrates the successful

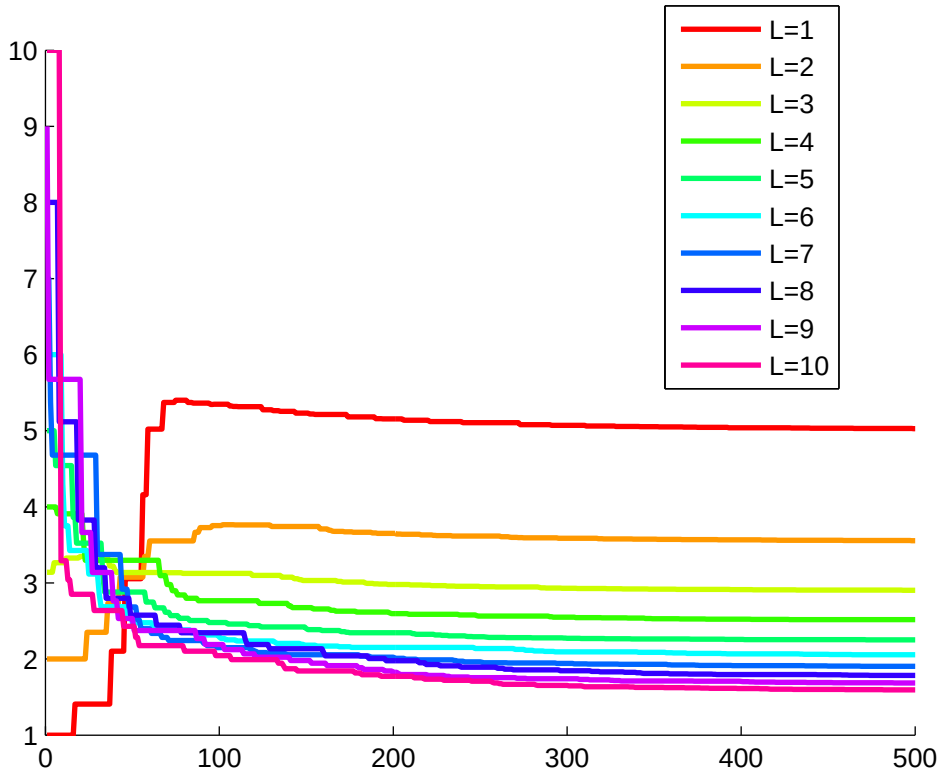

Figure 2.6: Finding K shortest paths between two nodes

finding of 10 shortest paths. The speed of convergence is fast enough considering the convergence occurred before the 100^{th} iteration. The following final pheromone trail values also verify the success of SIEigenAnt algorithm:

$$\tau_{500} = \{5.0274, 3.5556, 2.9038, 2.5164, 2.2507, 2.0547, 1.9028, 1.7844, 1.6850, 1.5941\}$$

It should be mentioned that no violation of minimum pheromone trails occurred in this experiment.

Then, we apply SIEigenAnt algorithm to the dynamic version of the previous problem where the edge lengths change, at the 200^{th} iteration, from the previous values 1 through 10 to the following:

$$L = \{1, 2, 12, 4, 0.5, 6, 7, 8, 1.5, 10\}$$

Fig. 2.7 demonstrates that the new optimal path $L = 0.5$ is found by SIEigenAnt right after the change. Besides, the new third optimal path $L = 1.5$ which previously had length $L = 9$ is found relatively quickly. The SIEigenAnt algorithm demonstrates slow but successful reaction in finding the new worst path $L = 12$

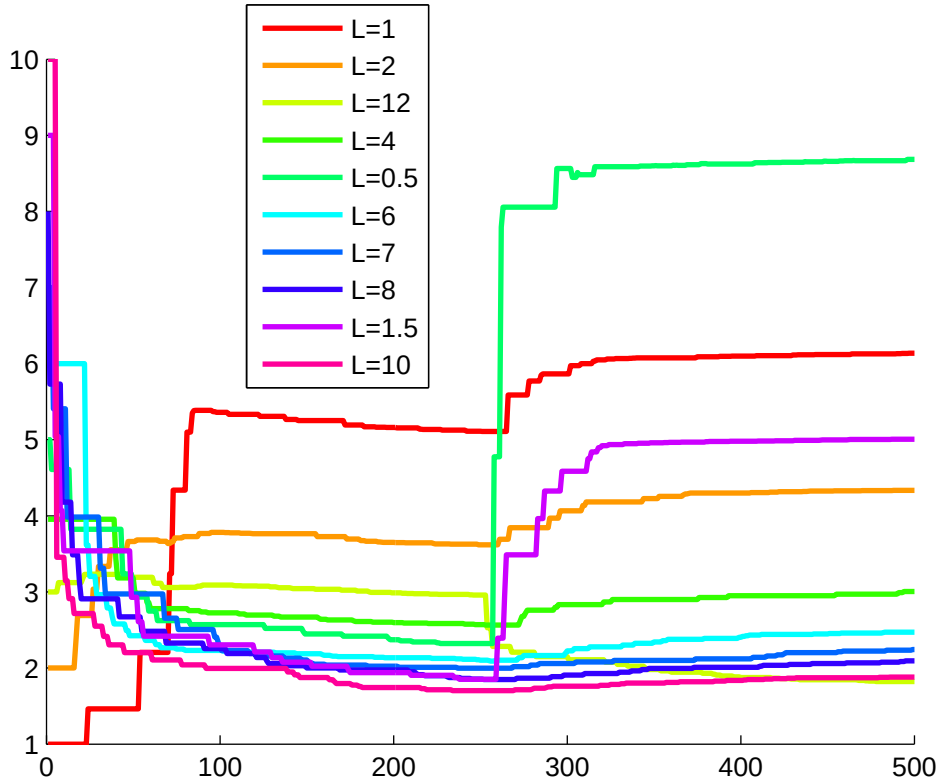

Figure 2.7: Finding dynamic K shortest paths between two nodes. The change that took place at the 200^{th} iteration for the corresponding edge is denoted by $\mapsto$ as: $L = \{1, 2, 3 \mapsto 12, 4, 5 \mapsto 0.5, 6, 7, 8, 9 \mapsto 1.5, 10\}$

which used to be the third optimal path $L = 3$. The following final pheromone trail values also verify the success of SIEigenAnt algorithm:

$$\tau_{500} = \{6.1379, 4.3342, 1.8232, 3.0042, 8.6838, 2.4706, 2.2430, 2.0933, 5.0051, 1.8816\}$$

It should be mentioned that no violations of minimum pheromone trail occurred in this experiment.

We now apply SIEigenAnt to a version of problems that are more challenging to solve in which the paths length are equal or slightly different. The following path length is chosen for experiment:

$$L = \{1, 1.5, 0.2, 4.9, 0.22, 5, 0.25, 1.5, 9, 5\}$$

Fig. 2.8 illustrates that the algorithm detects the two almost optimal paths of $L\{0.2, 0.22\}$. Two equal paths of length 1.5 and the three nearly equal paths of $L\{4.9, 5, 5\}$. Apart from that, K shortest paths are successfully found. It should be mentioned that in this experiment minimum pheromone trail limit is violated and

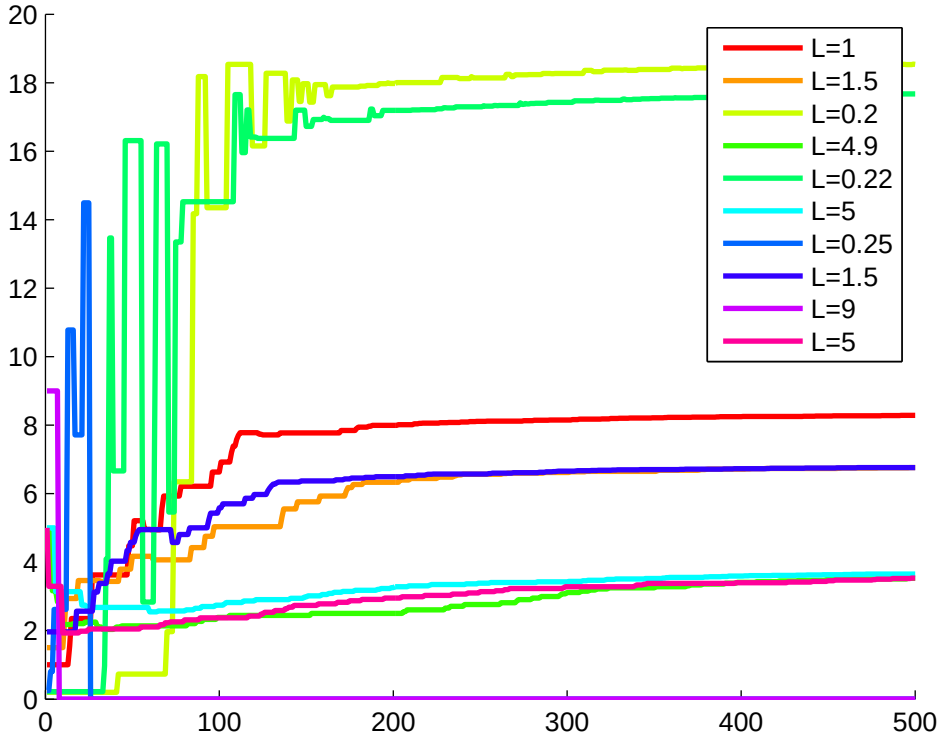

Figure 2.8: Finding K shortest path with equal or nearly equal lengths. The pheromone trails corresponding with three paths with lengths $\{4.9, 5, 5\}$ are converged equally near the value 2. The pheromone trails corresponding with two equal paths of length 1.5 are converged to the value 6.

therefore δ is decreased to 13.2.

Finally, we apply SIEigenAnt to the dynamic version of this type of problem where the path' lengths change at the 200^{th} iteration as follows:

$$L = [15.24.9.195.251.515]$$

Fig. 2.9 illustrates that after the 250^{th} iteration the orange plot relating to $1.5 \mapsto 5$ path length change decreases in reaction to the change. Moreover, the new optimal value of 0.19 which is slightly shorter than the previous of 0.2 is distinguished. However, the algorithm is incapable of finding the sudden change of the previously worst path 9 to the length 1. It should be mentioned that in this experiment minimum pheromone trail limit is violated and therefore δ is decreased to 12.3.

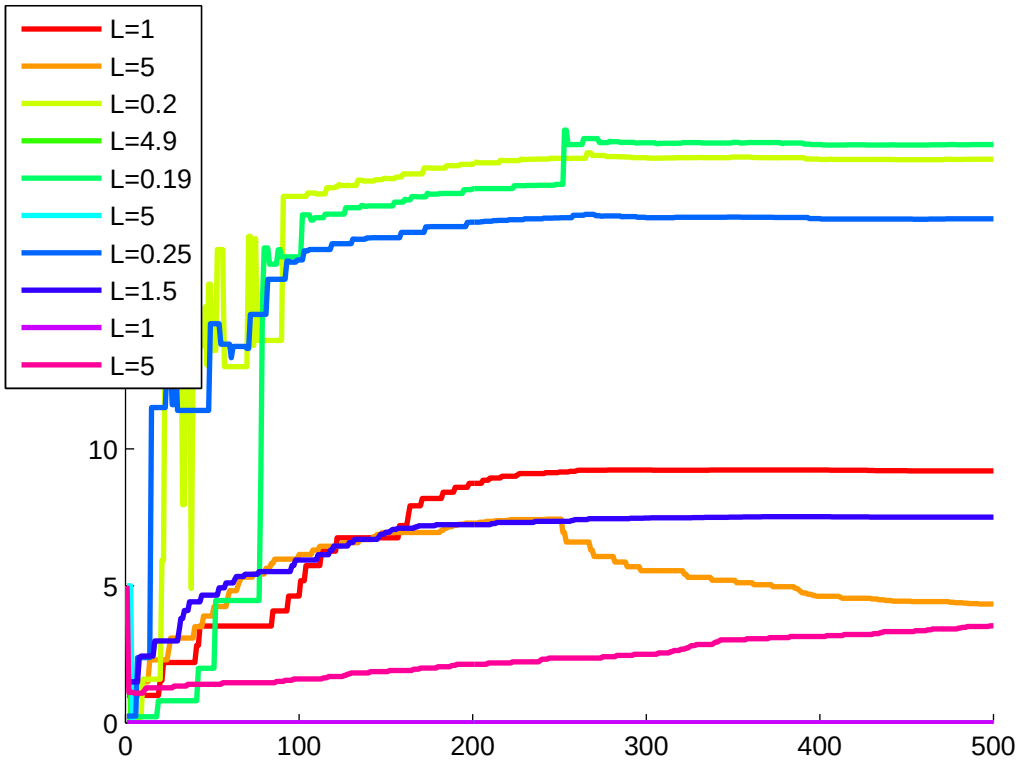

Figure 2.9: Finding dynamic K shortest path with equal or nearly equal lengths. After the 250^{th} iteration the orange plot relating to $1.5 \mapsto 5$ path length change decreases in reaction to the change and converges to a pheromone value equal to the path with length 5. The new optimal value of 0.19 which is slightly shorter than the previous optimal of 0.2 is distinguished. The algorithm is incapable of finding the sudden change of the previously worst path 9 to the length 1 (red plot).

2.3 Summary

In this chapter, we investigated the properties of EigenAnt using an analytical model originally developed for the related SACO algorithm in (IACOPINO and PALMER, 2012) for analysis of the stability and the impact of pheromone amplification parameter α. As a result, we improved EigenAnt algorithm by introducing two independent pheromone amplification parameters in its transition probability function as well as in the pheromone update dynamic.

We recalled the results of (IACOPINO and PALMER, 2012) regarding the effects of the parameter α on the SACO algorithm: it either accelerates the speed of the algorithm or influences the stability of the local optimal solution. In contrast, we illustrated that the two independent pheromone amplification parameters in IEigenAnt isolate these two effects so that IEigenAnt is able to maintain stability of the local optimal solution together with a fast speed of convergence to it. In a nutshell, IEigenAnt demonstrates a behavior which enables the designer of the algorithm to tune the convergence versus exploration and the speed of the algorithm separately. Subsequently, we investigate the conclusions about IEigenAnt in the application to RN which is an extended model for BCP and MKP modeled as an N-node BCP which involves constraints both in the stationary and dynamic environments.

Finally, we proposed a Sorting Improved EigenAnt algorithm with the pheromone removal parameter dependent on the corresponding pheromone trail distribution. We showed examples that suggest that SIEigenAnt is capable of finding K shortest paths between two nodes in both the stationary and dynamic cases.

Chapter 3

IEigenAnt for Combinatorial Optimization Problems

3.1 Introduction

In this Chapter, we investigate the performance of IEigenAnt algorithm applied to RN and MKP. The first is an investigation for the conclusions about the application of IEigenAnt to BCP when it is modeled as a graph with number of edges more than 2 at each node in each layer. The latter checks the compatibility of the conclusions with the BCP with constraint. In addition, we compare the IEigenAnt with its predecessor EigenAnt and common ACO algorithms applied to the problems. It should be mentioned that we check the results of each experiment based on both of the CV and CS points of view.

3.2 Routing Networks

A three stage multi-hop network as in Fig.3.1 is solved by EigenAnt in (JAYADEVA *et al.*, 2013). Note that, without loss of generality, all edges in the last layer can be chosen to have length one. Generally speaking, the multi-hop network in Fig.3.1 can be modeled as an $N \times O$ matrix.

The goal of an algorithm that solves a 3×3 RN problem in Fig. 3.1 is to find the shortest path that starts from the node (1) and ends in the node (11), and takes three hops from the starting node (1) through the intermediate nodes in layers $\{1, 2, 3\}$ until it gets to the destination node (11).

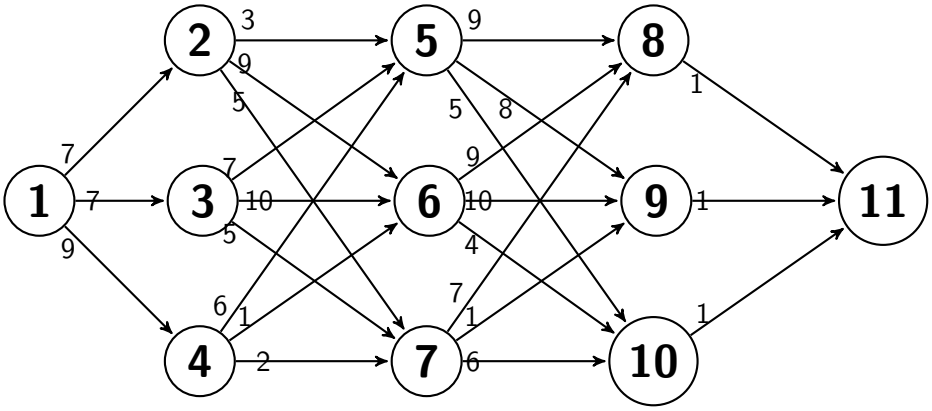

Figure 3.1: A Routing Network of size 3×3

3.2.1 IEigenAnt Application to RN

Solution Construction

One ant at each iteration constructs a solution from the source node (1) to the final node (11) (Fig.3.1).

As in the simple two node case, each pheromone trail corresponds to an edge of the graph in Fig.3.1. Initial pheromone values for each edge are chosen to be equal to the length of the corresponding edge as suggested in (SHAH, 2011) in order to initially make bad paths preferable to the ants and, thus to make convergence more difficult. At each node of the layer i, an ant chooses the outgoing edge j from the O possible choices with the following general transition probability function:

$$P_{ij}(\alpha_1) = \frac{\tau_{ij}^{\alpha 1}}{\sum_{l=1}^{O} \tau_{ij}^{\alpha_1}} \tag{3.1}$$

Generally speaking, each ant in the $N \times O$ RN has O edges to choose via roulette-wheel selection, from the first layer to the N^{th} layer.

Cost Evaluation

The cost of the r^{th} constructed solution is evaluated by summing the length of the edges that form the solution from the first to the N^{th} layer ($L_r = \sum_{l=1}^{N} d_l$), where d_l is the length of the edge selected at the l^{th} layer.

Pheromone Update

At each iteration, the pheromone trails relating to the edges of the constructed solution are updated successively. The following dynamic is for IEigenAnt

pheromone update of each edge $x \in \{1, \cdots O\}$ at the layer j for the r^{th} path:

$$\tau_{jx}(t+1) = (1-\rho)\,\tau_{jx}(t) + \left(\frac{Q}{L_r}\right) P_{jx}(\alpha_2) \tag{3.2}$$

where $P_{jx}(\alpha_2) = \frac{\tau_{jx}^{\alpha 2}}{\sum_{l=1}^{O} \tau_{jl}^{\alpha_2}}$.

Finding the Final Result

In the CS point of view, the final path is constructed by selecting each edge based on the maximum pheromone trail at each layer from the source node to the destination node. In the CV point of view, the path with the smallest evaluated cost to date is saved.

3.2.2 Experimental Results

We generate a 10×10 RN randomly with edges lengths in the range of [1 100]. First, we compare EigenAnt algorithm results with different pheromone removal parameter values of $\rho = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. It should be mentioned that all the experiments in this Subsection are done 30 times. We run each algorithm for the 10×10 RN model until 20000 cost evaluations take place. Also, we set the scaling parameter $Q = N = 10$. Fig. 3.2 and Fig. 3.3 illustrate the boxplots of the experiments in the CV and CS points of views, respectively. The mean of each experiment is marked using the symbol $\Diamond$. Standard Deviation σ of each experiment is also shown in the figures. It can be seen from the figures that the results from the CV point of view are better, as expected, due to the weaker convergence condition. Fig. 3.2 depicts the complete dominance of parameter $\rho = 0.1$ for the CV point of view with the mean cost of 67.13. Fig. 3.3 depicts the dominance of $\rho = 0.1$ for the CS point of view with the mean cost of 77.23 while the use of $\rho = 0.2$ illustrates a risky behavior that has three failures which causes its mean increase to 79.26 even though it has the best median. Besides, the SD of EigenAnt with $\rho = 0.1$ through the CS point of view ($\sigma = 7.35$) emphasizes the dominance of this type of EigenAnt algorithm.

Then, we apply the IEigenAnt algorithm to 10×10 RN problem. We test all the 8 cases of IEigenAnt in a way that whenever the parameters α_1 or α_2 are bigger or smaller than one, it is half less or half greater than one(i.e. $\alpha_1 \quad or \quad \alpha_2 < 1 \mapsto \alpha_1 \quad or \quad \alpha_2 = 0.5$ and $\alpha_1 \quad or \quad \alpha_2 > 1 \mapsto \alpha_1 \quad or \quad \alpha_2 = 1.5$). Fig. 3.4 and Fig. 3.5 demonstrate the boxplots of the experiments related to all the 9 conclusions drawn about IEigenAnt in the CV and CS points of view, respectively. It should be mentioned that we used $\rho = 0.1$ due to the best result achieved for the pheromone

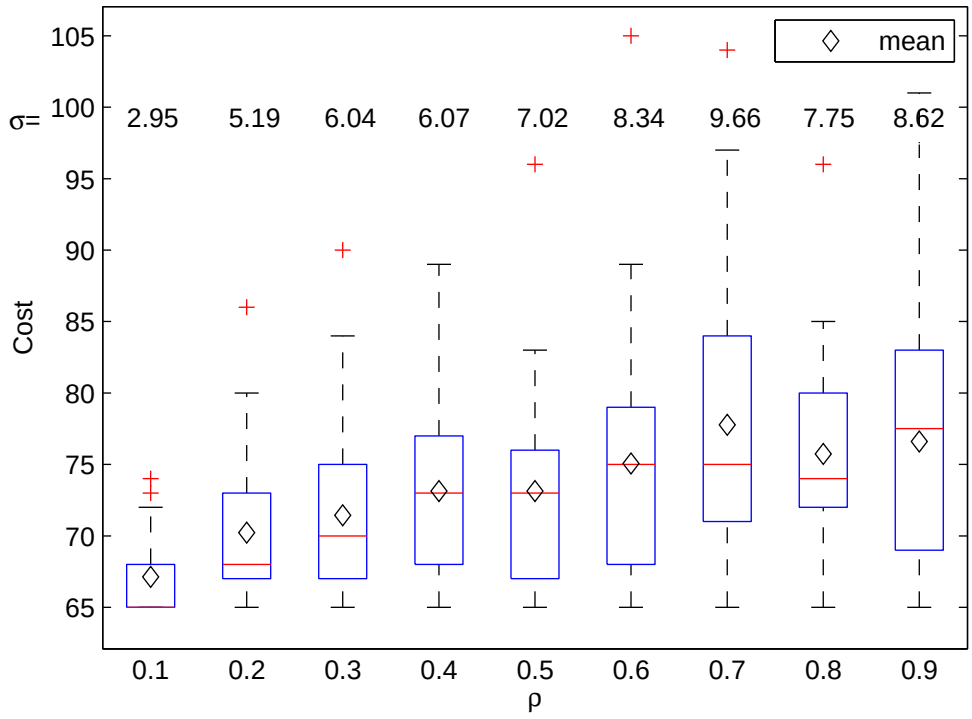

Figure 3.2: EigenAnt comparison for CV as the pheromone removal parameter varies from 0.1 to 0.9. As is usual for box plots, the whiskers represent the quartiles, and the red crosses the outliers, while we have added the means (shown as diamonds) and the numerical values of the SD across thirty runs for each parameter settings (row of numbers at the top of the plot).

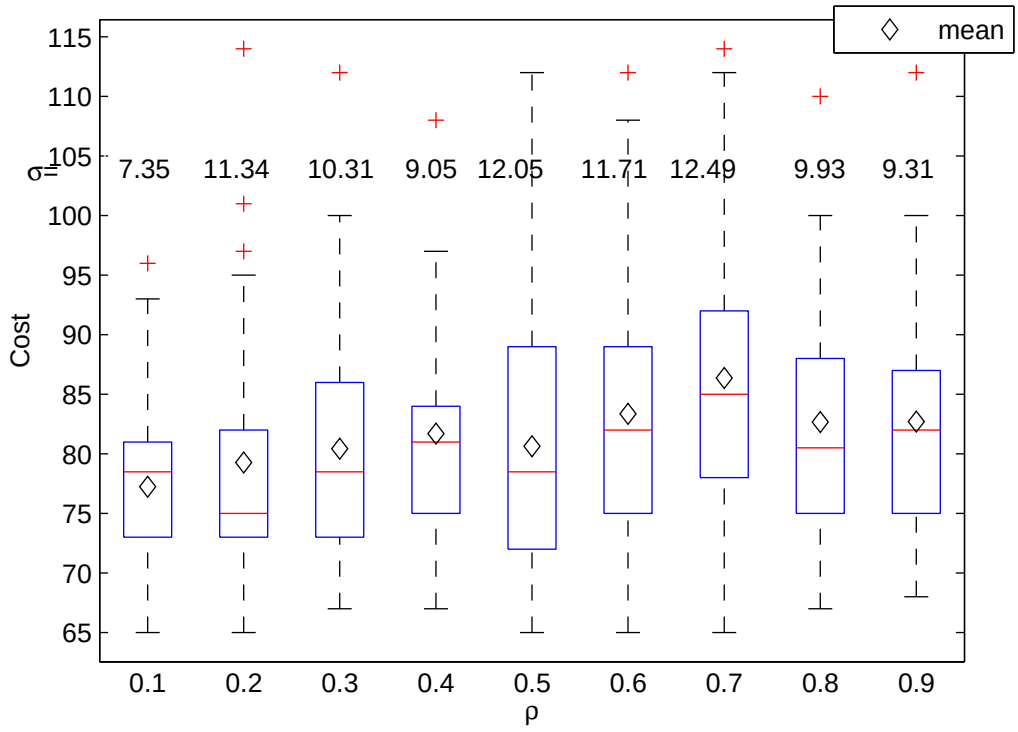

Figure 3.3: EigenAnt comparison for CS as the pheromone removal parameter varies from 0.1 to 0.9. As is usual for box plots, the whiskers represent the quartiles, and the red crosses the outliers, while we have added the means (shown as diamonds) and the numerical values of the SD across thirty runs for each parameter settings (row of numbers at the top of the plot).

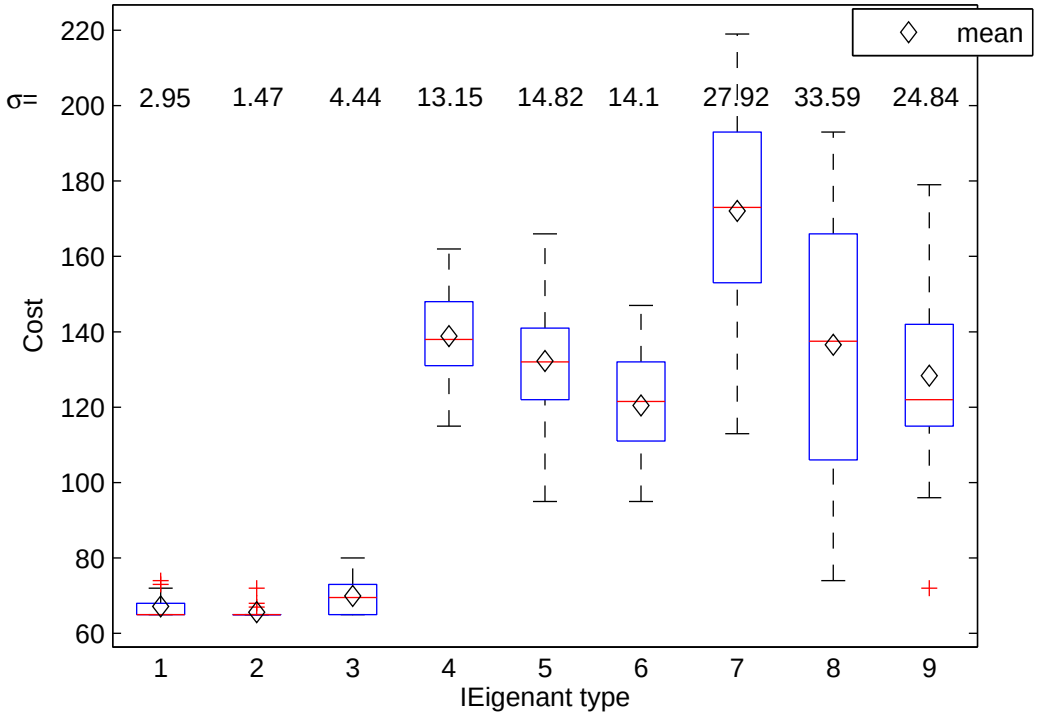

Figure 3.4: EigenAnt (the first box plot) compared with IEigenAnt (the second box plot to the ninth) for CV with pheromone removal parameter $\rho = 0.1$. IEigenAnt types are numbered associated with the conclusions from the Chapter 2 as follows:
1 $(\alpha_1 = 1, \alpha_2 = 1)$,2 $(\alpha_1 < 1, \alpha_2 = 1)$,3 $(\alpha_1 > 1, \alpha_2 = 1)$,
4 $(\alpha_1 < 1, \alpha_2 < 1)$,5 $(\alpha_1 = 1, \alpha_2 < 1)$,6 $(\alpha_1 > 1, \alpha_2 < 1)$,
7 $(\alpha_1 < 1, \alpha_2 > 1)$,8 $(\alpha_1 = 1, \alpha_2 > 1)$,9 $(\alpha_1 > 1, \alpha_2 > 1)$
As is usual for box plots, the whiskers represent the quartiles, and the red crosses the outliers, while we have added the means (shown as diamonds) and the numerical values of the SD across thirty runs for each parameter settings (row of numbers at the top of the plot).

removal value for the EigenAnt application in the previous experiments. The first box plot in each of the figures is related to the first conclusion which is about the original EigenAnt. It can be seen that EigenAnt outperforms all other IEigenAnt types except for the second one which is with $\alpha_1 = 0.5$ and $\alpha_2 = 1$. The theoretical conclusions in Chapter 2 explained that the second type of IEigenAnt converges to the local optimal solution with a faster speed than the EigenAnt. Thus, this type of IEigenAnt outperforms EigenAnt verifying the theoretical expectations.

Fig. 3.4 and Fig. 3.5 also depict the major dominance of IEigenAnt algorithms with $\alpha_2 = 1$ over those with $\alpha \neq 0$ is noticeable. We perform an empirical analysis of pheromone removal parameters $\rho = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ for the IEigenAnt algorithms with $\alpha_2 = 1$ and different values of parameter $\alpha_1 = \{0.1, 0.2 \ldots 1.5\}$. Considering 15 cases of IEigenAnt algorithm and 9 pheromone

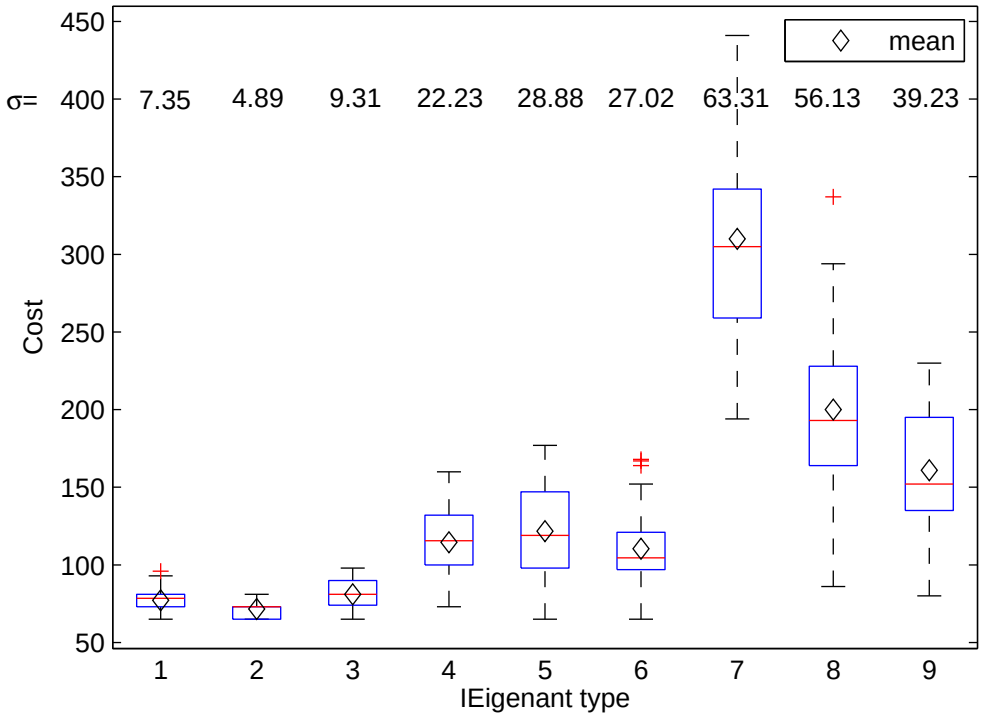

Figure 3.5: EigenAnt (the first box plot) compared with IEigenAnt (the second box plot to the ninth) for CS with pheromone removal parameter $\rho = 0.1$. IEigenAnt types are numbered associated with the conclusions from the Chapter 2 as follows:
1 $(\alpha_1 = 1, \alpha_2 = 1)$,2 $(\alpha_1 < 1, \alpha_2 = 1)$,3 $(\alpha_1 > 1, \alpha_2 = 1)$,
4 $(\alpha_1 < 1, \alpha_2 < 1)$,5 $(\alpha_1 = 1, \alpha_2 < 1)$,6 $(\alpha_1 > 1, \alpha_2 < 1)$,
7 $(\alpha_1 < 1, \alpha_2 > 1)$,8 $(\alpha_1 = 1, \alpha_2 > 1)$,9 $(\alpha_1 > 1, \alpha_2 > 1)$
As is usual for box plots, the whiskers represent the quartiles, and the red crosses the outliers, while we have added the means (shown as diamonds) and the numerical values of the SD across thirty runs for each parameter settings (row of numbers at the top of the plot).

removal parameters for each case, we have the total $15 * 9 = 135$ algorithms to experiment. We experiment each algorithm 30 times and all other parameters and initial values are set the same as the previous experiments. Table 3.1 demonstrates the mean and SD of the best results and the associated pheromone removal parameter ρ for each of the IEigenAnts with different parameters α_1 from CV and CS points of view.

α_1	ρ of Best CV	Best CV Result	ρ of Best CS	Best CS Result
0.1	0.9	Mean= 66 SD=2.02	0.9	Mean=74.43 SD=5.73
0.2	0.5	Mean= 65.87 SD=1.81	0.8	Mean=73.87 SD=5.27
0.3	0.3	Mean= 66 SD=2.16	0.5	Mean=73.8 SD=7.72
0.4	0.2	Mean= 65.93 SD=1.57	0.2	Mean=73.2 SD=6.99
0.5	0.2	Mean= 66.2 SD=1.83	0.4	Mean=71.37 SD=5.5
0.6	0.3	Mean= 66.47 SD=2.19	0.3	Mean=74.73 SD=6.17
0.7	0.1	Mean= 66.47 SD=2.91	0.2	Mean=75.6 SD=5.86
0.8	0.1	Mean= 67.03 SD=3.19	0.1	Mean=74.1 SD=7.69
0.9	0.1	Mean= 68.1 SD=4.91	0.2	Mean=75.8 SD=9.51
1	0.1	Mean= 67.13 SD=2.96	0.1	Mean=77.23 SD=7.35
1.1	0.1	Mean= 68.7 SD=3.85	0.2	Mean=77.9 SD=10.06
1.2	0.1	Mean= 69.33 SD=4.44	0.2	Mean=78.27 SD=6.26
1.3	0.1	Mean= 69.33 SD=4.4	0.2	Mean=79.57 SD=9.69
1.4	0.1	Mean= 69.87 SD=4.57	0.1	Mean=79.53 SD=10.35
1.5	0.1	Mean= 72.93 SD=5.21	0.1	Mean=83.93 SD=7.41

Table 3.1: Statistical analysis of IEigenAnt with respect to $0 < \alpha_1 \leq 1.5$ and ρ

From Table 3.1, we notice that IEigenAnt with $\alpha_1 = 0.2$ and $\rho = 0.5$ demonstrates the best performance from the CV point of view. Moreover, the IEigenAnt with $\alpha_1 = 0.5$ and $\rho = 0.4$ demonstrates the best performance from the CS point of view. It can be noticed that the CV point of view has better results than the CS which makes it preferable unless the application of the problem requires

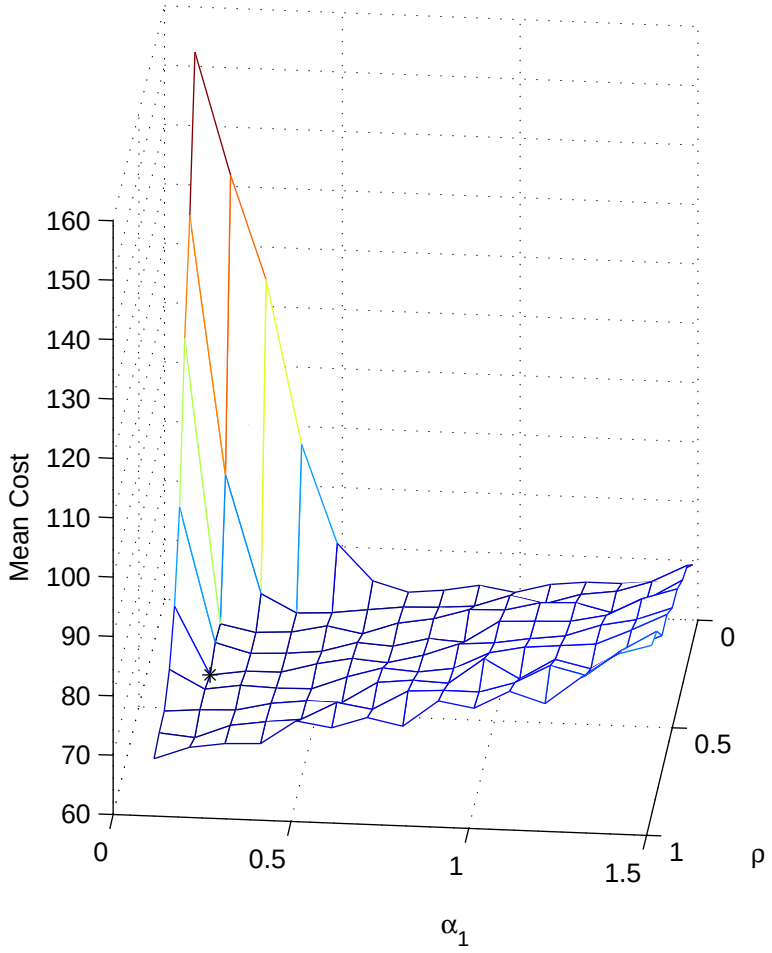

Figure 3.6: Illustration of IEigenAnt parameter analysis for CV. The parameter α_2 is set to 1 in order to guarantee the convergence of system to local optima while α_1, associated with the y axis, varies from 0.1 to 1.5 and ρ, associated with the x axis, is tested with the value varying from 0.1 to 0.9 for each choice of α_1. The z axis depicts the mean value of 30 experiments for each case while the star point depicts the optimal parameter tuning.

the CS point of view. Furthermore, the advantage of decoupling the stability and convergence tuning can be noticed from Table 3.1 because generally the smaller the α_1 is the better performance can be expected. Besides, the worse result of the algorithm in case of $\alpha_1 > 1$ than the $\alpha_1 < 1$ is also noticeable. EigenAnt has performance inferior to IEigenAnt2 ($\alpha_1 < 1$) except for the case with $\alpha_1 = 0.9$ from the CV point of view. The worsening performance of the IEigenAnt algorithm with the increase of α_1 verifies that the theoretical analysis given in Chapter 2. Fig. 3.6 and Fig. 3.7 illustrate the 3D plot of mean value empirical parameter analysis of IEigenAnt with respect to α_1 and ρ from the CV and CS points of view, respectively. The marked points at each figure depict the optimal parameter values.

Next, we compare IEigenAnt and EigenAnt with ACO algorithms. The

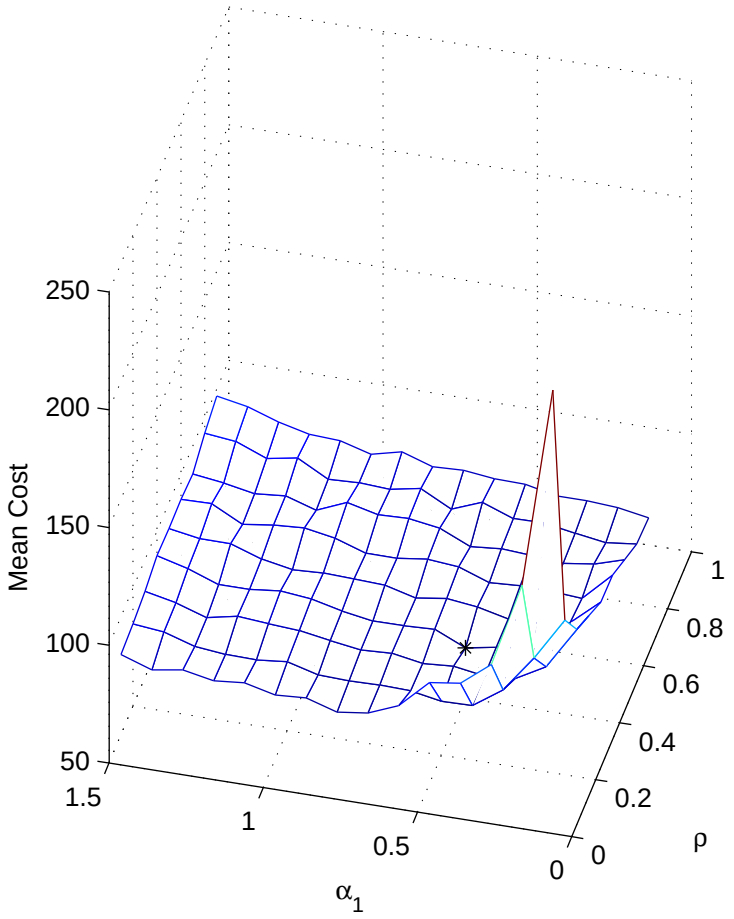

Figure 3.7: Illustration of IEigenAnt parameter analysis for CS. The parameter α_2 is set to 1 in order to guarantee the convergence of system to local optima while α_1, associated with the y axis, varies from 0.1 to 1.5 and ρ, associated with the x axis, is tested with the value varying from 0.1 to 0.9 for each choice of α_1. The z axis depicts the mean value of 30 experiments for each case while the star point depicts the optimal parameter tuning.

parameter settings and number of cost evaluations are the same as previous cases for IEigenAnt; however, in order to make an allowance for ACO algorithms we run them until 40000 cost evaluations take place. We use the optimal parameters chosen from the previous experiments for EigenAnt and IEigenAnt namely, $\alpha_1 = 0.2$, $\alpha_2 = 1$ and $\rho = 0.5$ for IEigenAnt, from CV point of view. For the CS case, we use $\alpha_1 = 0.5$, $\alpha_2 = 1$ and $\rho = 0.4$. Also, $\rho = 0.1$ is used for the EigenAnt experiments from both the CV and CS points of view. The ACO algorithms are as follows:

1. **ACSh:** We choose ACS with the parameters suggested in (DORIGO and GAMBARDELLA, 1997) (i.e. $\rho = 0.1$, $\beta = 2$, $q_0 = 0.9$, $\Upsilon = 0.1$ in (Eq. (1.5))). Also, L_{nn} in (Eq. (1.6)) is the cost of a path constructed through the greedy choice of each edge from the source to the destination node. We set the heuristic η_{ij} the same as Eq. (1.1) inversely proportional to each edge length. Finally, 10 ants construct the solutions at each iteration.

2. **ACS1:** As suggested in (MONTEMANNI *et al.*, 2005), we use a transition probability function similar to IEigenAnt; without heuristics. We use the q_0 parameter as of ACS algorithms, all other parameters are set as in the previous case and $\alpha = 1$.

3. **ACS0.5:** We use the ACS algorithm as the previous case with $\alpha = 0.5$ in order to check the performance of the algorithm with a smaller α.

4. **ASh:** We apply AS algorithm with a heuristic similar to the one used for ACS. The parameters in AS are those suggested in (DORIGO *et al.*, 1996) ($Q = 10$, $\rho = 0.5$, $\alpha = 1$, $\beta = 5$). Moreover, 10 ants construct solutions at each iteration.

5. **SACO:** We apply SACO algorithm with $\rho = 0.1$ and $\alpha = 1$.

We ran the above ACO algorithms 30 times until 40000 cost evaluations take place. Fig. 3.8 demonstrates the comparison of IEigenAnt with ACO algorithms from the CV point of view in which the superiority of IEigenAnt algorithm is noticeable. It should be mentioned that the ACS with heuristic and SACO had a poor performance to such an extent that we exclude them from Fig. 3.8. Fig. 3.9 illustrates the comparison of the algorithms from the CS point of view. It can be seen from the Fig. 3.9 that ACS algorithms without heuristic demonstrate a competitive performance with the IEigenAnt from the CS point of view.

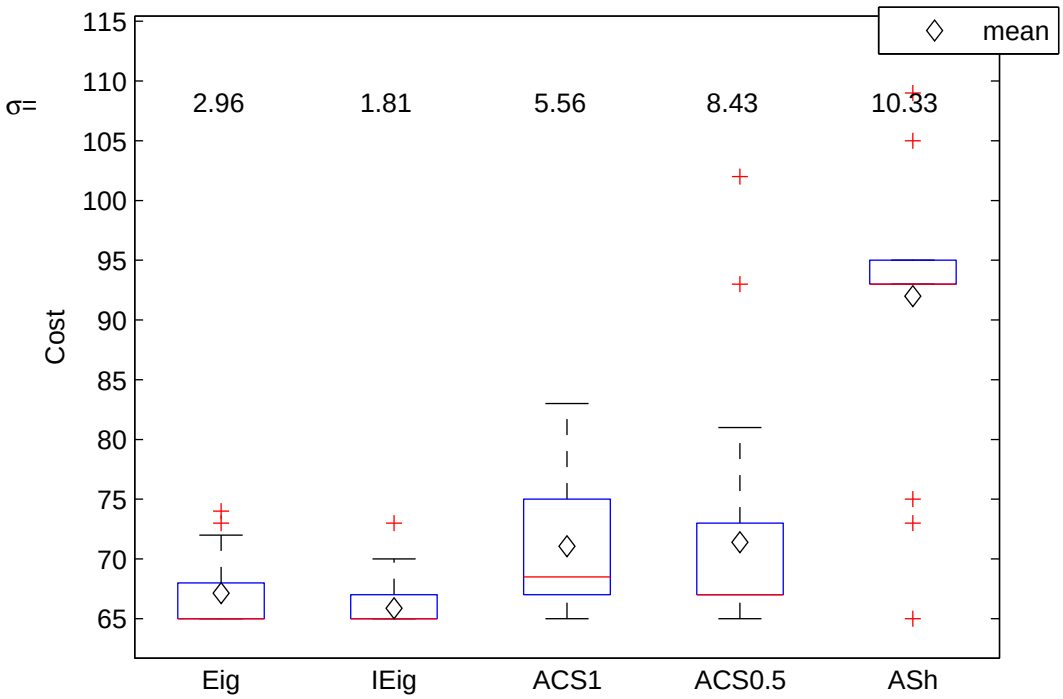

Figure 3.8: EigenAnt with $\rho = 0.1$ (the first box plot), IEigenAnt with the optimal parameter tuning (the second box plot) are compared with other ACO algorithms including ACS without heuristic that has the parameter choices of $\alpha_1 = 1$ and $\alpha_1 = 0.5$ (the third and the fourth box plots, respectively) and AS algorithm with heuristic (the last box plot) for CV. As is usual for box plots, the whiskers represent the quartiles, and the red crosses the outliers, while we have added the means (shown as diamonds) and the numerical values of the SD across thirty runs for each parameter settings (row of numbers at the top of the plot).

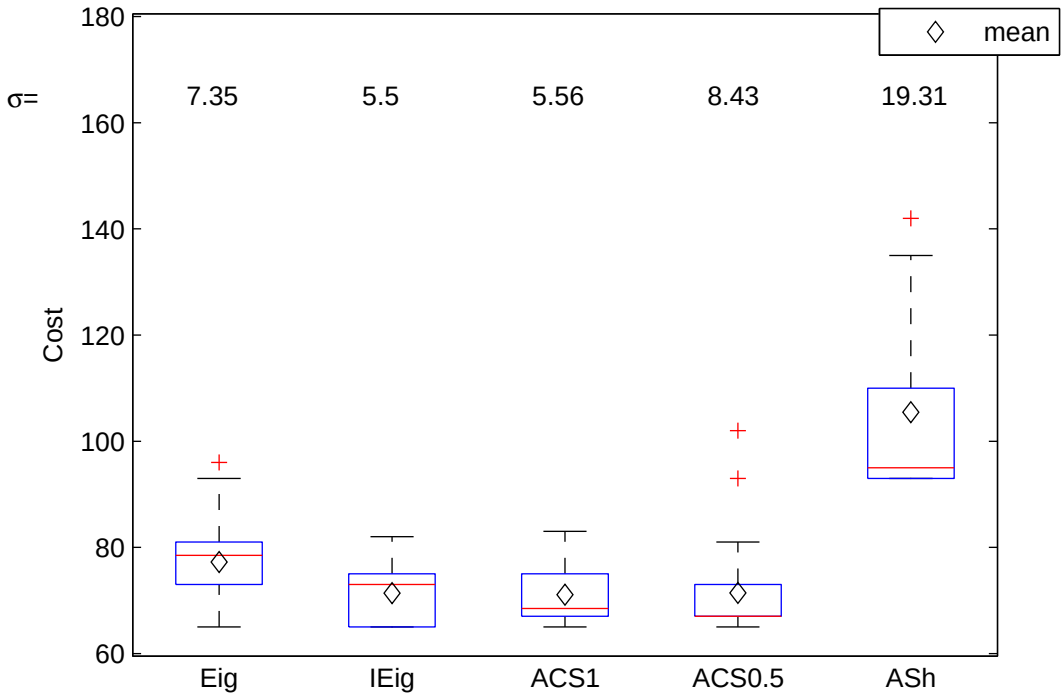

Figure 3.9: EigenAnt with $\rho = 0.1$ (the first box plot), IEigenAnt with the optimal parameter tuning (the second box plot) are compared with other ACO algorithms including ACS without heuristic that has the parameter choices of $\alpha_1 = 1$ and $\alpha_1 = 0.5$ (the third and the fourth box plots, respectively) and AS algorithm with heuristic (the last box plot) for CS. As is usual for box plots, the whiskers represent the quartiles, and the red crosses the outliers, while we have added the means (shown as diamonds) and the numerical values of the SD across thirty runs for each parameter settings (row of numbers at the top of the plot).

3.3 Multidimensional Knapsack Problem

The multidimensional knapsack problem is an NP-hard subset problem which has been studied for its wide range of applications relating to distributed systems (LAHAMI *et al.*, 2012), allocation problems (DUENAS *et al.*, 2014) and cognitive radio networks (SONG *et al.*, 2008). A set of N items with profits $v_j > 0$ and Z resources with capacities $W_z > 0$ are given. Each item j consumes an amount of $w_{zj} \geq 0$ from each resource z. The $0-1$ decision variables x_j indicate which item are selected. The goal is to choose a subset of items with maximum total profit. Selected items must, however, not exceed resource capacities; this is expressed by the knapsack constraints. The MKP is formulated as follows:

$$max \left(\sum_{j=1}^{N} x_j.v_j \right) \tag{3.3}$$

$$\sum_{j=1}^{N} x_j.w_{zj} < W_z \tag{3.4}$$

$$x \in \{0, 1\}$$

$$z \in \{1, 2, \ldots, Z\}$$

$$j \in \{1, 2, \ldots, N\}$$

In (KE *et al.*, 2010; LEGUIZAMON and MICHALEWICZ, 1999), MKP is modeled as a graph in which each ant k drops pheromone on the nodes corresponding each item j. In (FIDANOVA, 2002), the MKP is modeled as a complete graph in which two edges connect each pair of nodes in order to illustrate the consecutive order of the selected items. Each ant drops pheromone on the edge corresponding the order of selection between the items i and j. In (ALAYA *et al.*, 2004), one edge connects each item i to the item j so that the order of selection is not important in the process of dropping pheromones by the ants. The mentioned types of modeling have a graph-based format in which all the items are considered by each ant from every step of the solution construction phase. Hence, the violation of constraints is checked before each step of the solution construction phase.

In order to use the IEigenAnt algorithm, which proved promising for the RN problem, we choose to model MKP as a BCP. Among the works done in the area of ACO application to MKP, the paper (KONG *et al.*, 2008) modeled MKP as a BCP depicted in Fig. 3.10. Each pheromone trail τ_{jx} corresponds to choosing the item j at the related layer number $j = \{1 \cdots N\}$ through the edge with $x = 1$, or not choosing the item j through the edge with $x = 0$. In contrast with graph-based modelings, the ants do not consider all the items at each step of the solution construction phase in BCP. Since the order of checking each item is predefined in

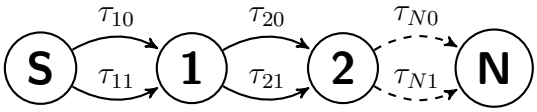

Figure 3.10: A BCP model for an MKP with N items

BCP, the violation of constraints can not be checked before each step of the solution construction phase. In other words, in a graph-based modeling, you have N items from which you pick item j and decides to select or not select the item, while in BCP modeling you have all the N items already sorted and based on this order you check which item to choose or not. The former has much freedom of exploration and complexity while the latter has less freedom of exploration and is simpler. Thus, the use of constraint handling techniques is necessary in modeling MKP as BCP (COELLO, 2002; MEZURA-MONTES and COELLO, 2011).

3.3.1 Constraint Handling

In (KONG *et al.*, 2008), a repair algorithm that modifies item choices until constraints are satisfied suggested in (CHU and BEASLEY, 1998) is used as a constraint handling technique. The other possible constraint handling technique for the application of IEigenAnt algorithm applied to MKP modeled as BCP is penalty function (COELLO, 2002; MEZURA-MONTES and COELLO, 2011). First, we explain the application of IEigenAnt to MKP with the repair algorithm used in (KONG *et al.*, 2008). Then, we explain the application of IEigenAnt to the MKP with penalty function.

Repair Algorithm

In order to initiate the repair algorithm, the items should be resorted and renumbered according to the decreasing order of their pseudo-utility value. The pseudo-utility value is calculated through surrogate relaxation method introduced in (PIRKUL, 1987). The surrogate relaxation method is the simplification of the problem into a single constraint problem as follows:

$$\sum_{j=1}^{N} \left(\sum_{z=1}^{Z} a_z w_{zj} \right) x_j \leq \sum_{z=1}^{Z} W_z \tag{3.5}$$

where a_z is the surrogate multiplier.

Surrogate multipliers are determined by solving the relaxed MKP (i.e. variable x can take any value $\in [0, 1]$) by linear Programming and using the values of the

dual variable as surrogate multipliers. Then, the pseudo-utility value is calculated as follows:

$$u_j = \frac{v_j}{\sum_{z=1}^{Z} w_z a_{zj}} \tag{3.6}$$

The items are then sorted in decreasing order based on their pseudo-utility values u_j. It should be mentioned that the above procedure is used in (BRANKE *et al.*, 2006) with the title of **real valued representation with weight coding**.

An ant constructs its solution at each iteration by defining the value of $x = \{0, 1\}$ at each layer j from 1^{st} to the N^{th}(i.e. selecting or deselecting the 1^{st} to the N^{th} corresponding item). The transition probability function is calculated at each layer j separately as follows:

$$P_{jx} = \frac{\tau_{jx}^{\alpha_1}}{\tau_{j0}^{\alpha_1} + \tau_{j1}^{\alpha_1}} \tag{3.7}$$

After construction of each solution, the feasibility of the solution is checked. In case of an infeasible solution, the repair algorithm takes charge. The repair algorithm in (KONG *et al.*, 2008) is inspired from (CHU and BEASLEY, 1998) in which it has two phases: DROP and ADD. In the first phase, each bit of the solution string is examined in increasing order of pseudo-utility u_j and each bit with the value of one is changed to zero until the solution becomes feasible. In the second phase, the reversion process of examining each bit in decreasing order of u_j takes charge. A bit changes from zero to one as long as feasibility is not violated. Having executed the repair algorithm, the cost of the solution r is evaluated. The shortcoming of repair method is its time complexity because the procedures of ADD and DROP are time consuming attributed to the increase in the number of function evaluation they impose. Moreover, the repair method has a weak exploration behavior because the infeasible space would not be searched.

Before evaluating the solution cost, we should observe that MKP is a maximum problem while IEigenAnt is designed to find minimum values. Thus, MKP is recast as a minimum value problem. Since in (3.3), the goal is to maximize the profit of selected items x_j; inversely minimizing the profits of unselected items $1 - x_j$ leads to the following minimization problem:

$$minf\,(x) = \left(\sum_{j=1}^{N} v_j . \,(1 - x_j) \right) \tag{3.8}$$

Thus, the evaluated cost is the summation profits of the unselected items ($L_r = f(x)$). Then, the pheromone update phase takes charge in which at each layer j the following pheromone update takes place for the edge x associated with the solution r:

$$\tau_{jx}(t+1) = (1-\rho)\,\tau_{jx}(t) + (Q/L_r)\,\frac{\tau_{jx}^{\alpha_2}(t)}{\tau_{j0}^{\alpha_2}(t) + \tau_{j1}^{\alpha_2}(t)} \tag{3.9}$$

Finally, the result associated with the CV point of view is the solution with the best-to-date evaluated cost. Furthermore, the result associated with CS point of view is the solution constructed from the greedy choice of each edge x at the layer j based on the value of corresponding pheromone trail τ_{jx}.

Penalty Function

The solution construction is the same as the previous constraint handling technique except for the repair algorithm that is not used in the penalty function method. In other words, the constructed solution might be infeasible so that the exploration also takes place in the infeasible region. However, the cost evaluation of IEigenAnt application to MKP with penalty function constraint handling technique is different from the previous repair method. In the penalty function method the constraint problem is transformed into an unconstrained problem by augmenting the objective function $f(x)$ as follows:

$$\phi(x) = f(x) + \Omega(x) \tag{3.10}$$

where $\phi(x)$ is the expanded objective function to be optimized, and $\Omega(x)$ is the penalty value (MEZURA-MONTES and COELLO, 2011). The evaluated cost is also the expanded objective $\phi(x)$.

We propose to calculate the penalty $\Omega(x)$ as follows:

$$\Omega(x) = \sum_{z=1}^{Z} \lambda_z \max(0, \mu_z)^{\zeta} \tag{3.11}$$

where λ is a penalty factor, ζ is a penalty amplification parameter and μ_z is the violation amount of a constraint z.

Various methods for defining penalty factor λ have been suggested. The penalty factor should be selected in such a way that it generates feasible solutions while maintaining a sufficient exploration behavior. Choosing a penalty factor that is too

large always leads to feasible solutions; however, the search of the algorithm could be limited to the interior of the feasible region so that the boundaries of the region could be missed. On the other hand, choosing a penalty factor that is too small could lead to an infeasible solution. An ideal penalty factor λ generates the feasible solutions frequently while it sometimes generates infeasible solutions so that the algorithm can search the boundaries more accurately.

Generally speaking, three approaches for defining the penalty factor are suggested(COELLO, 2002; MEZURA-MONTES and COELLO, 2011):

1. **Static Penalty**: In this approach, the penalty factors remain constant during the algorithm. The main drawback of this approach is poor generalization (i.e the values suitable for one problem might be unsuitable for another one).

2. **Dynamic Penalty**: In this approach, the penalty factors are dependent on time t. Normally, they increase over time. In spite of claims about the out-performance of dynamic penalty approach over the static, in practice such advantages have not been observed (COELLO, 2002).

3. **Adaptive Penalties**: In this approach, the penalty factors change by using a feedback from the search process. However, some additional parameters are added to the problem in order to apply the feedback. Those additional parameters introduce the tuning challenges.

Discussion

Repair method is not suitable for the MKP problems with large number of items because the number of cost evaluations would soar up through the ADD and DROP procedures in the repair algorithm; thus, in (KONG *et al.*, 2008) the experiments are only done for problems with 100 items. Adaptive and dynamic penalty methods are experimented with and found to have similar performance with static method in (ALI *et al.*, 2014; COELLO, 2002; MOHAMMADI *et al.*, 2015). As a result the static penalty function is found to be suitable for solving MKP with IEigenAnt.

We propose a novel static penalty approach for the IEigenAnt application to MKP in which it performs competitively for different benchmarks of MKP. First, we normalize the constraint violation amount μ_z as follows:

$$y_z = \frac{\sum_{j=1}^{N} w_{zj} x_j - W_z}{W_z} \tag{3.12}$$

$$\mu_z = max\,(0, y_z) \tag{3.13}$$

As a result, the expanded objective function ϕ is rewritten as follows:

$$\phi = \sum_{j=1}^{N} v_j.(1 - x_j) \left(1 + \sum_{z=1}^{Z} \lambda_z \mu_z\right) \tag{3.14}$$

The recipe for defining penalty factors λ_z consists of normalization of each constraint based on its maximum limit. The equation for normalization is as follows:

$$y = \sum_{z=1}^{Z} W_z \tag{3.15}$$

$$\chi_z = \frac{W_z}{y} \tag{3.16}$$

The idea is to set higher penalty factors λ_z to the constraints with smaller constraint capacity W_z. In order to do so, constraints are sorted based on their capacity W_z in an ascending order while χ_z is sorted in a descending order. As a result, the higher value of χ is set to the constraint with lower capacity. Finally, a scaling parameter ν is multiplied by each χ from which the penalty factor λ is formed. The scaling parameter value depends on the number of constraints and the constraint tightness(CHU and BEASLEY, 1998). It should be mentioned that the parameter ζ in (3.11) is set to one.

The pheromone update phase is the same as the previous repairing method except for the parameter L_r in (3.9) is usually chosen to be equal to evaluated cost ϕ. However, we use the following value for L_r which is suggested in (ALAYA *et al.*, 2004):

$$L_r = \frac{1}{1 + \phi - L_{gb}} \tag{3.17}$$

where L_{gb} is the best-to-date evaluated cost.

The final results in CV and CS points of view are calculated in the same way as the previous method. The preliminary experiments demonstrate that the final feasible solution for CS point of view is not always guaranteed; hence, we only consider CV point of view in subsequent experiments. Subsequently, we will apply IEigenAnt to the MKP benchmarks with 500 items and 5 constraints using the novel static penalty function as the constraint handling method.

3.3.2 Avoiding the Stagnation Problem

Pheromone trails on a path not included in the current solution tends to converge to zero (JAYADEVA *et al.*, 2013). As soon as a pheromone trail on a given path becomes zero, there is no chance for the RWS to choose that path as a component of future solutions. Unfortunately, such paths might belong to a better solution which has not been discovered yet. Therefore, the algorithm could stagnate in a sub-optimal solution. Three methods have been suggested in the literature for escaping from this type of stagnation. The first one is an ad-hoc local search changing a certain number of random indices of global best solution tour for a certain number of times in each iteration (KONG *et al.*, 2008). This method is completely based on chance and increases the time complexity too much. Another method is to reinitialize the pheromone concentrations before they reach zero (KONG *et al.*, 2008). This is also problematic as the evolution of concentration is forgotten, the program continues as if it had been completely reinitialized.

The third alternative is the use of a Max-Min method for controlling the pheromone concentrations which means to check each pheromone concentration so that it neither becomes larger than a maximum limit nor smaller than a minimum one. Whenever any pheromone trail violates the minimum or maximum limit, its value should be replaced by the relative minimum or maximum limit value. This method is introduced in (STÜTZLE and HOOS, 2000) as Max-Min ACO. In order to escape the stagnation better, the minimum pheromone value is suggested to change dynamically (KE *et al.*, 2010). In the subsequent experiments we check the violations of pheromone limits before the dynamical update. Besides, the minimum pheromone limit changes after passing predefined numbers of iterations.

3.3.3 Experimental Results

In this Subsection, the results of the algorithm are compared with the previously proposed ACO algorithms applied to the benchmarks extracted from OR library (BEASLEY, 1990).

In all the simulations, parameters are chosen as follows:

$$\rho = 0.001$$
$$\beta = 520 \times N$$
$$\alpha_1 = 0.3$$
$$\alpha_2$$

The simulations are done 30 times for each of the OR benchmark problems.

Each maximum capacity W_z is equal to a percentage of the total object weight in the constraint z. This percentage is called tightness ratio (CHU and BEASLEY, 1998). The first 10 problems are with the tightness ratio of 25%, the second 10 problems are with the tightness ratio of 50% and the last 10 problems are with 75% of the tightness ratio.

Maximum pheromone limit value is set to 3000, 4000 and 5000 for benchmarks with tightness ratios of 25%,50% and 75%, respectively. Furthermore, the minimum limit is initially set to 2×10^{-10}. The order in which minimum pheromone limit value is changed after every 1800^{th} of iterations is demonstrated in table 3.2. In the table 3.2, the order of minimum pheromone limit increase up to 24^{th} ($24 \times 1800 = 43200$) multiplication is demonstrated in the Table.

Table 3.2: Minimum Pheromone limit change order

IN	1	2	3	4	5	6
PMLV	2×10^{-9}	2×10^{-10}	2×10^{-9}	2×10^{-8}	2×10^{-9}	2×10^{-8}
IN	7	8	9	10	11	12
PMLV	2×10^{-7}	2×10^{-8}	2×10^{-7}	2×10^{-6}	2×10^{-7}	2×10^{-6}
IN	13	14	15	16	17	18
PMLV	2×10^{-5}	2×10^{-6}	2×10^{-5}	2×10^{-4}	2×10^{-5}	2×10^{-4}
IN	19	20	21	22	23	24
PMLV	2×10^{-3}	2×10^{-4}	2×10^{-3}	2×10^{-2}	2×10^{-3}	2×10^{-2}

In table 3.2, IN stands for Iteration Number (IN) after which Pheromone Minimum Limit Value (PMLV) changes.

Inspired by (KUMAR, 2016), we initialize the pheromone trails to the corresponding LP relaxation solution multiplied by 100. We solve the LP relaxation problem through CVX, a package for specifying and solving convex programs (CVX RESEARCH, 2012; GRANT and BOYD, 2008). In addition, we perform Randomized Rounding algorithm (RAGHAVAN and TOMPSON, 1987) before the IEigenAnt in order to have an acceptable solution at the beginning of the algorithm.

Randomized rounding is a technique applicable to solve a class of 0-1 integer programming problems. The technique consists of solving the problem through linear programming relaxation. The fractional solution obtained through linear programming, is rounded to integer value using the probabilistic method called randomized rounding. The procedure is to choose the value of integer 0 or 1, with a probability dependent on the calculated fractional value x.

Repeating such a random selection several times is proved to give a result arbitrarily near the optimal integer programming solution (RAGHAVAN and TOMPSON, 1987). The procedure is illustrated in the algorithm 1 where Π_I is a $0-1$ linear program, with variable $x_i \in \{0,1\}$, and Π_R is its relaxation, with

$\widehat{x_i} \in [0, 1]$.

Algorithm 1 Randomized Rounding algorithm

1: Solve Π_R; Let the solutions $\widehat{x} = \{\widehat{x1}, \widehat{x_2}, \ldots, \widehat{x_n}\}$ take the values $\widehat{x} \in [0, 1]^n$.
2: Set the variables x_i randomly to one or zero according to the following rule:

$$Prob.[x = 1] = \widehat{x_i}$$

Reviewing the articles that have worked on OR MKP benchmarks, we can see that only in (KE *et al.*, 2010) the authors attacked all 500 items benchmarks with ACO algorithm. In (ALAYA *et al.*, 2004) only 5 benchmarks are tested with ACO algorithm. Also, in (KONG *et al.*, 2008) the benchmarks with 100 items are already solved optimally and there is no challenge left in this type of benchmark. However, they did not try benchmarks of 500 items that are more difficult due to the fact that they use the repair method with a high time-complexity. Therefore, as mentioned before, we only concentrate on the type of benchmarks with 500 items using our novel static penalty function method.

In Table 3.3, hybrid RR-IEigenAnt performance on $500 - 5$ MKP benchmark of 25% tightness ratio is illustrated. Best result, average and SD of EigentAnt are compared with the dynamical Max-Min ACO used in (KE *et al.*, 2010) with and without local search and the method in (ALAYA *et al.*, 2004). The dynamical Max-Min ACO with local search is called as DMMAS+ls and the other as DMMAS. Also, the result of randomized rounding is showed in Table 3.3 as RR. In all of the tables, NA stands for not available. As can be seen in Table 3.3, the real competition is between Hybrid RR-IEigenAnt and dynamical Max-Min ACO. As DMMAS+ls includes local search, it has better results except in three cases that Hybrid RR-IEigenAnt algorithm wins over it even though it does not use any local search. In other cases, IEigenAnt method gives results better than Alaya and a little bit worse than the Dynamical Max-Min without local search. It should be mentioned that the penalty function parameter λ is set to $\frac{50}{5}$ for this category of benchmark problems.

In Table 3.4 the results of Hybrid RR-IEigenAnt algorithm implementation on $500 - 5$ benchmarks using 50% tightness ratio with penalty function parameter of $\lambda = \frac{50}{3}$ are depicted. As can be seen, the results are reasonably close to the DMMAS method without local search. In one case of benchmark No.11, the best result of Hybrid RR-IEigenAnt is better than the DMMAS algorithm without local search.

In table 3.5, the results of Hybrid RR-IEigenAnt algorithm implementation on $500 - 5$ benchmarks using 75% tightness ratio with the penalty function parameter of $\lambda = 50$ are depicted.

Table 3.3: Hybrid RR-IEigenAnt on 500-5 MKP with 25% tightness ratio

No		Hybrid RR-IEigenAnt	DMMAS+ls	DMMAS	Alaya	RR
00	Best	119926	**120148**	120116	119893	119504
	Mean	119830	**120111**	120056	119658	119504
	SD	25.2	17.3	25.5	NA	0.0
01	Best	117660	**117879**	117857	117604	117604
	Mean	117610	**117841**	117786	117423	117604
	SD	16.8	13.7	27.9	NA	0.0
02	Best	121061	**121131**	121109	120846	120556
	Mean	120970	**121097**	121043	120622	120556
	SD	37.4	17.7	27.2	NA	0.0
03	Best	120695	**120804**	120785	120534	120298
	Mean	120600	**120776**	120715	120279	120298
	SD	102.7	11.3	24.1	NA	0.0
04	Best	122123	**122319**	**122319**	122126	121600
	Mean	122040	**122303**	122254	121829	85158
	SD	43.9	16.3	29.8	NA	56702
05	Best	121910	**122024**	121992	NA	121479
	Mean	121800	**121991**	121936	NA	121479
	SD	52.8	14.8	23.6	NA	0.0
06	Best	**119266**	119127	119096	NA	119076
	Mean	**119170**	119093	119043	NA	43445
	SD	34.9	12.6	26.4	NA	58074
07	Best	**120634**	120568	120536	NA	120078
	Mean	120520	**120525**	120472	NA	120078
	SD	47.1	20.0	27.6	NA	0.0
08	Best	121436	**121575**	121551	NA	121003
	Mean	121290	**121537**	121479	NA	121003
	SD	54.2	14.4	31.9	NA	0.0
09	Best	120579	**120717**	120692	NA	120328
	Mean	120500	**120678**	120627	NA	120328
	SD	33.7	17.9	25.7	NA	0.0

Table 3.4: Hybrid RR-IEigenAnt on 500-5 MKP with 50% tightness ratio

No		Hybrid RR-IEigenAnt	DMMAS+ls	DMMAS	RR
10	Best	218289	**218428**	218400	218201
	Mean	218240	**218397**	218344	218201
	SD	35.4	12.9	28.1	0.0
11	Best	221114	**221202**	221191	220450
	Mean	220980	**221168**	221117	220160
	SD	60.06	47.7	30.9	67177
12	Best	217390	**217534**	217528	217304
	Mean	217320	**217513**	217459	217304
	SD	25.5	13.4	30.9	0.0
13	Best	223534	**223560**	**223560**	222979
	Mean	223360	**223547**	223499	148460
	SD	44.8	11.0	24.6	106770
14	Best	218894	**218966**	218962	218722
	Mean	218790	**218956**	218905	218722
	SD	37.2	11.1	25.9	0.0
15	Best	220425	**220530**	220496	220078
	Mean	220310	**220497**	220455	220078
	SD	45.0	14.1	17.2	0.0
16	Best	219914	**219989**	219987	219247
	Mean	219850	**219974**	219924	109520
	SD	73.6	15.8	31.5	111390
17	Best	218057	**218194**	218180	217888
	Mean	218040	**218171**	218124	217888
	SD	3.5	10.9	25.8	0.0
18	Best	216827	**216963**	216958	216402
	Mean	216780	**216948**	216904	129660
	SD	25.6	111.2	28.6	107670
19	Best	219601	**219719**	219704	219281
	Mean	219570	**219694**	219657	219281
	SD	26.1	8.0	20.6	0.0

Table 3.5: Hybrid RR-IEigenAnt on 500-5 MKP with 75% tightness ratio

No		Hybrid RR-IEigenAnt	DMMAS+ls	DMMAS	RR
20	Best	295763	**295828**	**295828**	295271
	Mean	295720	**295809**	295764	167290
	SD	40.5	13.9	20.9	148790
21	Best	308010	**308086**	308077	307582
	Mean	307930	**308069**	308023	307582
	SD	35.1	9.7	25.6	0.0
22	Best	299720	**299796**	**299796**	299352
	Mean	299680	**299781**	299738	199500
	SD	34.7	13.0	16.5	143480
23	Best	306453	**306480**	**306480**	306258
	Mean	306320	**306467**	306427	306150
	SD	52.6	9.0	27.5	19.7
24	Best	300262	**300342**	300334	299914
	Mean	300200	**300334**	300280	299914
	SD	38.7	11.2	22.2	0.0
25	Best	302491	**302571**	302560	302433
	Mean	302440	**302556**	302525	302433
	SD	11.3	7.6	19.7	0.0
26	Best	301239	**301329**	301325	300889
	Mean	301180	**301317**	301278	300889
	SD	27.5	7.9	26.7	0.0
27	Best	306349	**306454**	306422	306157
	Mean	306290	**306426**	306388	306157
	SD	25.1	8.5	20.4	0.0
28	Best	302751	**302828**	302809	302197
	Mean	302680	**302810**	302765	271980
	SD	43.2	13.5	22.1	92209
29	Best	299859	**299906**	299902	299406
	Mean	299790	**299894**	299845	299406
	SD	68.3	9.1	23.8	0.0

3.4 Summary

In this chapter, we verified the theoretical analysis done for the IEigenAnt algorithm application to the RN problem as a generalized case of BCP. Besides, we performed a statistical comparison of IEigenAnt with EigenAnt and ACO algorithm applied to the RN problem in which IEigenAnt algorithm outperformed the other algorithms.

MKP as a BCP problem with constraint is also checked for the benchmarks with 500 items and 5 constraints. We introduced a novel static penalty function for the constraint handling technique that is fast enough to cope with 500 nodes in BCP. Moreover, we hybridized IEigenAnt with RR in order to make the algorithm faster and used an iteration based dynamic mix-min method for stagnation avoidance. The results of the IEigenAnt algorithm with fast speed of convergence $\alpha_1 = 0.3 < 1$ and the convergence to local optimal solution $\alpha_2 = 1$ tuning were competitive with the best ACO algorithm that applied to these benchmarks to date while even in the two of benchmarks the algorithm found unknown optimal solutions.

Since EigenAnt was not capable of finding the feasible solutions, the theoretical analysis for IEigenAnt is to some extent verified for BCPs with constraint handling. Yet, it has not completely verified because IEigenAnt competitive performance is only achieved with CV point of view while the CS point of view is incapable of achieving the feasible solutions.

Chapter 4

IEigenAnt for Dynamic Combinatorial Optimization Problems

A Combinatorial Optimization Problem (COP) might change over time in a way that its optimal solution changes. This type of COP is entitled Dynamic COP (DCOP) which is simply defined as a sequence of static COPs linked up by some dynamic rules. (YANG *et al.*, 2013). Generally speaking, DCOP is in the family of Dynamic Optimization Problems (DOP) which is an optimization problem within a dynamic environment. In other words, a DOP is a problem with time dependent parameters (JIN and BRANKE, 2005).

In this thesis, we focus on DCOPs in which the problem instances change over time. The frequency and the magnitude of the change are related to the dynamics. After each change, the optimal solution might change. Tracking the Moving of Optimum (TMO) is the main goal of an algorithm dealing with DCOPs (MAVROVOUNIOTIS *et al.*, 2017).

The main challenge in TMO for algorithms applied to DCOPs is that the algorithms converge to the optimal solution before the change takes place so that the algorithm stagnates to the old solution (MAVROVOUNIOTIS *et al.*, 2017). Traditionally, Evolutionary Algorithms (EA) have been applied to solve DOPs because of their natural adaptive behavior (JIN and BRANKE, 2005). Recently, Swarm Intelligence (SI) algorithms that also have adaptive features have also been applied to DOPs. The adaptive feature in EA and SI promotes the knowledge transfer from the previously optimized environments (MAVROVOUNIOTIS *et al.*, 2017). Since IEigenAnt is inspired by ACO algorithms, we focus on the works done in the area of ACO algorithms in solving DCOPs. Two types of ACO frameworks are used to solve DCOPs: evaporation based and population based

(MAVROVOUNIOTIS *et al.*, 2017). We are interested in the former framework because the EigenAnt uses one ant at each iteration.

In order to tackle the premature stagnation problem, four strategies are adopted from EA algorithms (JIN and BRANKE, 2005) of which the two following strategies are applicable to IEigenAnt algorithm [1].

1. **Generate diversity after change:**

   In this strategy, the diversity is generated whenever a change is detected. Re-initializing the pheromone trails is the basic method to generate diversity in ACO algorithms; however, all the previous knowledge stored in the pheromone trails is destroyed in this way (EYCKELHOF and SNOEK, 2002). Nevertheless, complete resetting of the algorithm is necessary whenever the magnitude of change is too large (JIN and BRANKE, 2005). Another alternative is partial resetting whenever a change is detected near the place that the dynamic change occurs (GUNTSCH *et al.*, 2001) and (EYCKELHOF and SNOEK, 2002). The downside of this method is that the instant of the change should also be detected. In (ANGUS and HENDTLASS, 2002), the pheromone trails in ACS algorithm are normalized whenever the change is detected. In (EYCKELHOF and SNOEK, 2002), a so-called shaking action takes place whenever a change is detected by which the pheromone trails τ_{ij} are decreased toward a small pheromone value τ_0 ($\tau_{ij} > \tau_0$) through the following logarithmic formula:

$$\tau_{ij} = \tau_0 \left(1 + log\frac{\tau_{ij}}{\tau_0}\right) \tag{4.1}$$

   The shaking action is applicable near the change location in big problems.

2. **Maintain diversity throughout the run:**

   The advantage of this strategy is that it does not necessarily require the detection of the change in order to tackle the stagnation problem. Local pheromone evaporation is considered as a method to maintain diversity; hence, EigenAnt is successful in finding the shortest path between two nodes when the paths' lengths change over time (JAYADEVA *et al.*, 2013) and in solving time-linkage [2] dynamic RN (SHAH, 2011). For such a reason, ACS algorithm

[1] Memory and population based strategies are also suggested; however, they require more than one ant at each iteration; which is not applicable to EigenAnt algorithm.

[2] Time-linkage DOP is a type of DOP in which the optimal solutions affect future changes in the dynamic environment.

is also successful dealing with DCOP in (LORPUNMANEE *et al.*, 2007) and (RANDALL, 2005).

It can be noticed that maintaining diversity is preferable to generating the diversity because detection of the change is not required. However, one important factor apparently overlooked by the researchers in the area of DCOPs is that the results should be stored in order to use the CS point of view. Storing the results to use in the CV point of view is problematic because the value of the optimal solution to date might be smaller than that the value of the new optimal solution. Such a scenario causes the algorithm to miss the new optimal solution in CV point of view so that change detection is required in order to reset the best-to-date value for the CV point of view which is in contrast with the motivation of using maintaining diversity strategy.

The aim of TMO is to find the best solution of the problem at any time. Different performance measurement methods are suggested to TMO in (MAVROVOUNIOTIS *et al.*, 2017). We use a performance measurement method that pays more attention to extreme behavior of the system, in particular, the best that the system can do in which the final solution before change, either from the CV or CS points of view, is compared with the global optimum. The global optimum can be measured by running the algorithm for each state of the problem separately as a stationary problem.

Some researchers suggested benchmark generators in order to compare different algorithms on DCOPs. For example, an XOR benchmark generator is suggested for BCPs in which the problem is not actually changing but flips the values of constructed solution from 0 to 1 and vice versa by filtering the constructed solution to different XOR maps over time (YANG, 2003). Hence, the optimal cost is constant over time. The generality of the benchmark generator is the motivation of using such a generator; however, the real-world scenario is sacrificed by using such a benchmark generator. Moreover, the constant optimal cost over time of such a benchmark generator is incapable of producing some challenging scenarios for an algorithm dealing with DCOPs. On the other hand, some researchers prefer to concentrate on benchmarks specified for single problems by defining the dynamics through switching between static instances. In this method, we can generate change instances in a controlled manner, as suggested in (UYAR and UYAR, 2009), so that the algorithm can be tested with different change severities. We use the latter method of generating dynamics in order to be able to verify the IEigenAnt performance dealing with different change severity. In the sequel, we test the performance of IEigenAnt algorithm dealing with DRN when the change time is not required to be detected. Such a test illustrates the application of CS point of view in IEigenAnt. Next, we assume that the change time can be detected and

perform a comparative analysis between EigenAnt, IEigenAnt and ACO algorithms designed for such problems. Furthermore, we verify the performance of IEigenAnt dealing with a more challenging problem of DMKP. Such a problem has rarely been taken into account by researchers due to the challenges that the change in constraints imposes on the problem.

4.1 IEigenAnt for DRN

As mentioned before, in (SHAH, 2011) EigenAnt is applied to 10×10 dynamic multi-hop network. The scenario used in (SHAH, 2011) is that the traffic on the optimal path after some time makes it non optimal and a second path becomes optimal. Then, after some additional time a third path becomes optimal due to the accumulated traffic on the second path. Later, the second path becomes optimal again since its traffic is decreased. Finally, a new optimal path emerges.

The dynamic scenario used in (SHAH, 2011), does not consider the challenging environmental changes that might occur in a DCOP. Moreover, the experiment done in (SHAH, 2011) lacks a performance index to evaluate the power of EigenAnt in adapting to change. Before generating a scenario for DRN, we should review some concepts in the topic of DOP. A **change** in a DOP occurs when the domain of feasible solutions and/or the evaluated cost f is modified during the optimization process. A **change cycle event** e is a series of iterations of an algorithm between two consecutive changes. The first change cycle event begins in the first iteration of the optimization process and ends one iteration before the first change, while the last change cycle event begins at the iteration after the last change and ends at the last iteration of the optimization processes(TINÓS and YANG, 2014). We propose the following change cycle events for our DRN problem:

1. **Original problem:** An algorithm faces a problem in its original state.

2. **Increased optimal cost:** This is a challenging situation for CV point of view. The problem changes in a way that the new optimal solution has a cost greater than the older one. For CV point of view, the best-to-date cost should be reset when a change takes place in order to be able to tackle such a problem change. In contrast, there would be no requirement of detecting the changing time from the CS point of view. In (SHAH, 2011), such a case of problem change is used whenever the optimal path is no longer available due to the heavy traffic.

3. **Emergence of a new optimal solution**: This case is also considered in SHAH (2011) where a new optimal path emerges.

Table 4.1: The stationary results of DRN change cycle events from the CV point of view

		Change cycle event number				
		1	2	3	4	5
IEigenAnt	Mean	65.87	70.37	**48**	65.87	137.7
	Best	**65**	**67**	**48**	**65**	**132**
EigenAnt	Mean	67.16	71.87	50.4	67.16	137.23
	Best	**65**	**67**	**48**	**65**	**132**
ACS	Mean	72.5	73.83	65.06	72.5	143.5
	Best	**65**	**67**	**48**	**65**	**132**

Table 4.2: The stationary results of DRN change cycle events from the CS point of view

		change cycle event number				
		1	2	3	4	5
IEigenAnt	Mean	71.37	79.73	50.87	71.37	144.5
	Best	**65**	69	**48**	**65**	**132**
EigenAnt	Mean	78.43	79.77	59.93	78.43	145.23
	Best	**65**	70	**48**	**65**	**132**
ACS	Mean	72.5	84.13	65.06	72.5	143.5
	Best	**65**	**67**	**48**	**65**	**132**

4. **Return of the optimal solution:** This case is also once used in (SHAH, 2011). This type of problem change is challenging and often memory-based strategies are suggested to solve such cases (JIN and BRANKE, 2005).

5. **Radical change**: This type of change is not considered in (SHAH, 2011). We generate a high magnitude of change in optimal value. The optimal solution is of higher cost than the one before change, so that this case also includes the second type.

We tested each of the change cycle events above for 30 times in a stationary RN problem with the best parameters of algorithms in algorithm in Chapter 3: IEigenAnt from CV point of view (i.e. $\alpha_1 = 0.2$ and $\rho = 0.5$), IEigenAnt from CS point of view (i.e. $\alpha_1 = 0.5$ and $\rho = 0.4$), EigenAnt (i.e. $\rho = 0.1$) and ACS without heuristic (i.e. $\rho = 0.1$, $\beta = 2$, $q_0 = 0.9$, $\Upsilon = 0.1$ and $\alpha = 0.5$). The other settings of the algorithms are the same as in Chapter 3. The best cost and mean cost of each algorithm from the CV and CS points of view are shown in Table 4.1 and Table 4.2, respectively.

It should be mentioned that Dijkstra's algorithm (DIJKSTRA, 1959) also found the values equal to the best found values in Table 4.1 for each stationary problem.

We tried to include the typical challenges that might occur in a DOP problem in the defined change cycle events. We create change dynamics by implementing the

mentioned change cycle events in such a way that the problem starts with the first change cycle event e_1 and changes until the 5^{th} change cycle event e_5. Furthermore, we repeat the 5 change cycle events once during the optimization.

The number of consecutive iterations of the algorithm in a change cycle event e is defined as **change cycle duration** Du_e in (TINÓS and YANG, 2014). Other definitions given in (TINÓS and YANG, 2014) are also valid for the DRN we have generated:

1. **Single time-dependent:** The optimal point in change cycle event e is dependent on optimal point in change cycle event g. Therefore, our DRN is a single time-dependent DOP because each of the change cycle events is generated based on the previous ones.

2. **Periodic:** The optimal point in change cycle event e is equal to the optimal point in the change cycle event $e - g$. We generate such a periodic DRN by repeating all the 5 change cycle events as mentioned before.

3. **Last environment dependent:** The optimal point in the change cycle event e is only dependent on the optimal point in the change cycle event $e - 1$. All the change cycle event except for the return of the optimal solution are last environment dependent change cycle events.

Change cycle duration (Du_e) is an important factor for an algorithm dealing with a DOP because a large Du_e can cause the algorithm converge to the optimal path of the older change cycle event. On the other hand, small Du_e requires the algorithm to have a fast reaction to change. Thus, we consider two types of change cycle events duration in our test. First, we test the algorithms on the DRN problem with $Du_e = 2400$ cost evaluations. Then, we test the algorithms on the DRN problem with $Du_e = 7200$ cost evaluations. Shorter change cycle duration means a lower frequency of dynamics in (JIN and BRANKE, 2005).

4.1.1 Solving DRN with Undetectable Change Time

We assume that the change time is not detectable; hence, we use CS point of view in our algorithms. We apply EigenAnt (with $\rho = 0.1$), IEigenAnt (with $\alpha_1 = 0.5$ and $\rho = 0.4$) and ACS without heuristic. We test ACS on DRN because it illustrated a competitive performance with IEigenAnt in Chapter 3. Moreover, ACS also has a local evaporation which is promising for DOPs according to (LORPUNMANEE *et al.*, 2007) and (RANDALL, 2005).

Short Change Cycle Duration Test

Fig. 4.1 demonstrates the comparison of EigenAnt, IEigenAnt and ACS algorithms in the first change cycle event (original problem) of DOP with short change cycle duration. The second box plots in Fig. 4.1 end with the suffix 6 after the name of each algorithm relates to the repeat of the first change cycle event. The mean of each experiment is depicted through $\Diamond$. SD (σ) of each experiment is also shown in this figure. We maintain these features in all the box plot figures. Generally, the algorithms perform better in the repeating change cycle event for the problems with shorter change cycle duration because more iterations have passed and the pheromone trails had more time to converge. Table 4.3 demonstrates the best and mean value of the test of each algorithm corresponding with each change cycle event. In order to evaluate the performance of each algorithm for this change cycle event and the future change cycle events, we can compare the results of each algorithm in 4.3 with its associated results in the stationary problem from the CS point of view in Table 4.2. Fig. 4.1 and Table 4.3 depict the superiority of IEigenAnt to such an extent that for the repeat of the original change cycle event (6^{th} change cycle event) IEigenAnt almost achieved a result as if it is solving a stationary problem. The mean IEigenAnt achieves for the 6^{th} change cycle event is even better than the mean EigenAnt achieves for stationary problem of the corresponding change cycle event from the CS point of view. It is important in the sense that the IEigenAnt adapts fast enough from the 5^{th} change cycle event of radical change with the mean cost of 212.93 to the 6^{th} change cycle event with the mean cost of 76.93 ($5 \mapsto 6$ change). EigenAnt is successful in performing this adaptation while ACS was unsuccessful according to Table 4.3.

Fig. 4.2 demonstrates the results dealing with the second change cycle event and its periodical repeat (7^{th} change cycle event) entitled *increased of the optimal cost*. We can notice that even in the first emergence of second change cycle event, IEigenAnt performs well in spite of the short change cycle duration of the DRN. For the repeat of the second change cycle event (7^{th} change cycle event), we also note the small value of $\sigma = 5.89$ and the Mean=77.73 which is smaller than the value 79.37 of the stationary solution.

Fig. 4.3 demonstrates the results of the algorithms dealing with the third change cycle event and its periodical repeat (8^{th} change cycle event) entitled the *emergence of a new optimal solution*. It can be noticed from Fig. 4.3, IEigenAnt illustrates its superiority over the other algorithms. Even in the third change cycle event, IEigenAnt achieved the best result of 48 which is the best found cost of the stationary problem (Table 4.3). For the change $2 \mapsto 3$ the mean of IEigenAnt changes as $107.7 \mapsto 70.83$ and for the change $7 \mapsto 8$ the mean of IEigenAnt changes as $77.73 \mapsto$

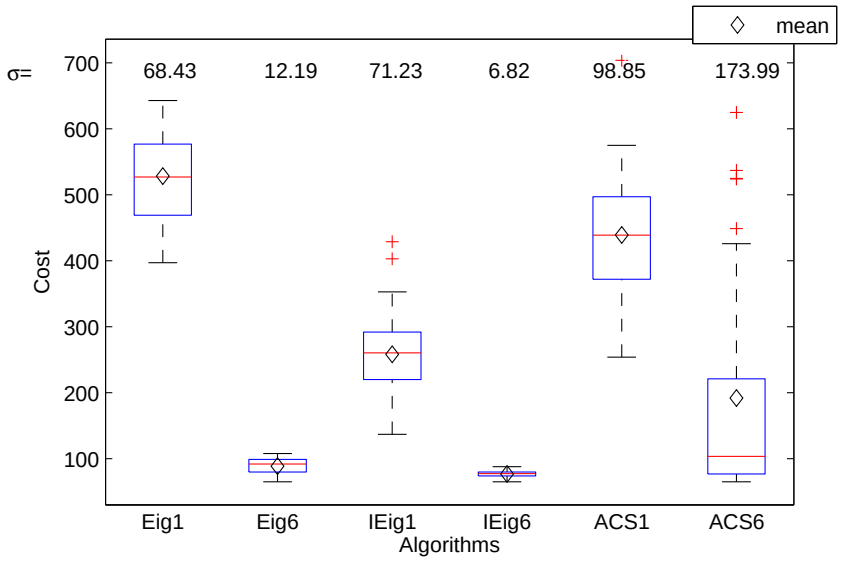

Figure 4.1: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 1 and the repeat of change cycle event 1 (denoted 6) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig6 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 1, which occurs again at the $1 + 5$(period of cycle)$= 6^{th}$ presentation.

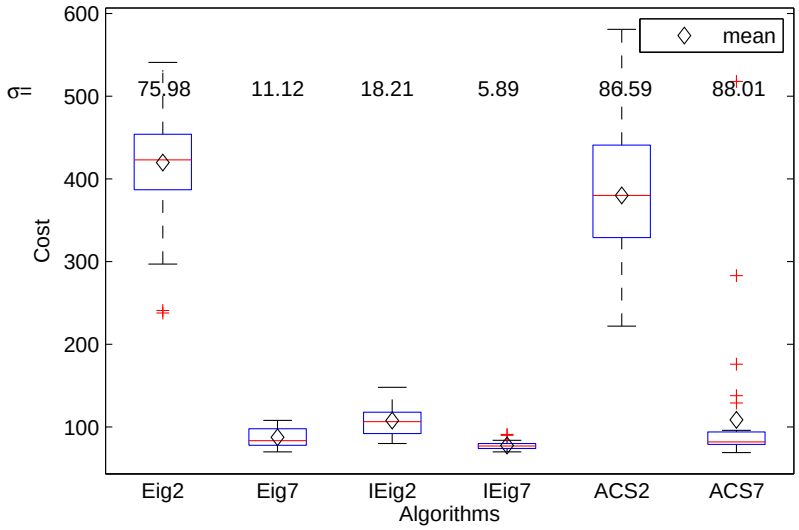

Figure 4.2: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 2 and the repeat of change cycle event 2 (denoted 7) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig7 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 2, which occurs again at the $2 + 5$(period of cycle)$= 7^{th}$ presentation.

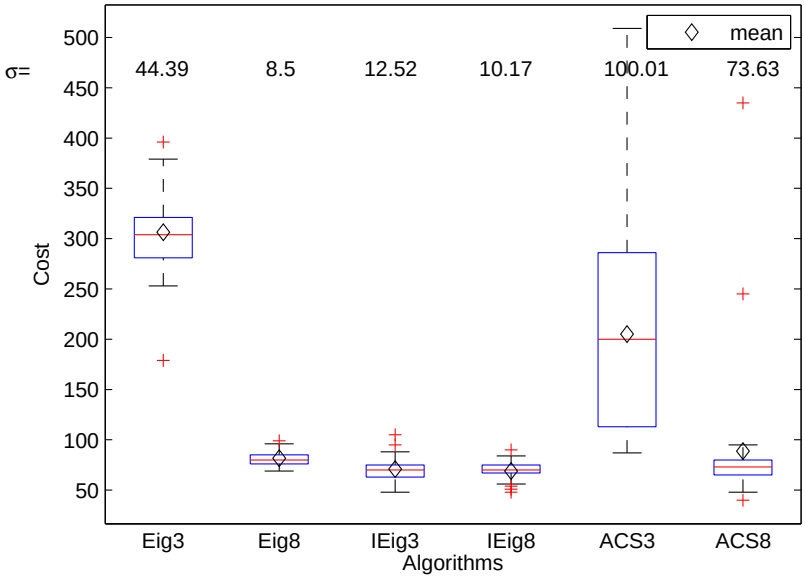

Figure 4.3: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 3 and the repeat of change cycle event 3 (denoted 8) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig8 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 3, which occurs again at the $3 + 5$(period of cycle)$= 8^{th}$ presentation.

68.9. Therefore, IEigenAnt demonstrates an acceptable reaction to the emergence of a new optimal solution. The achievement of ACS is also noticeable in the Table 4.3 with the newly found Best result of 40. It should be mentioned that even Dijkstra's algorithm applied to the stationary problem of the third change cycle event could not achieve a solution better than 48 because the solution corresponding with the value of 40 includes edges with the cost of zero that Dijkstra's algorithm assumes them unreachable. EigenAnt performs poorly in this type of change cycle event.

Fig. 4.4 demonstrates the performance of the algorithms dealing with the fourth change cycle event entitled *return of the original state*. The IEigenAnt illustrates superiority and even in the 4^{th} change cycle event we notice that IEigenAnt performs well. We can notice from Table 4.3 that EigenAnt and IEgenAnt achieve the best result of 65 in the 4^{th} change cycle event while ACS as incapable of doing so.

Fig. 4.5 demonstrates the performance of the algorithms dealing with the last change cycle event entitled *radical change* which is the most challenging dynamical change to the problem due to its high magnitude of the change. In fact, in (JIN and BRANKE, 2005) it is suggested to reset the problems dealing with such changes, which we are ruling out by supposing that the change time is unknown. The

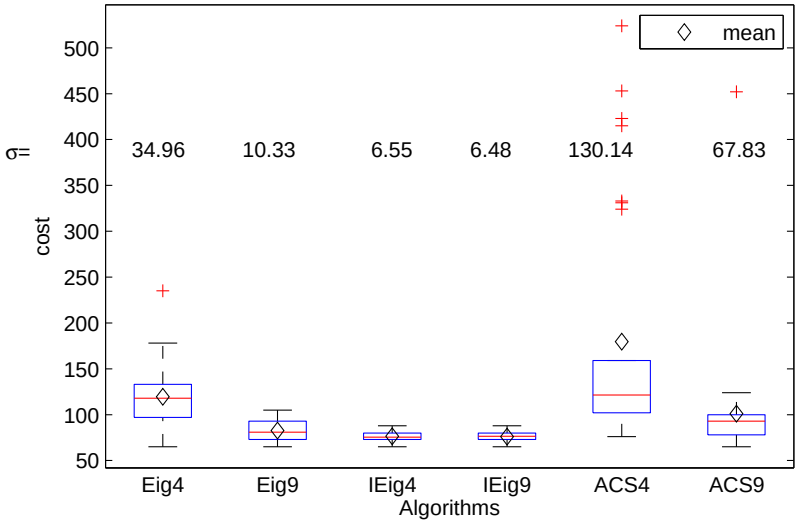

Figure 4.4: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 4 and the repeat of change cycle event 4 (denoted 9) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig9 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 4, which occurs again at the $4 + 5$(period of cycle)$= 9^{th}$ presentation.

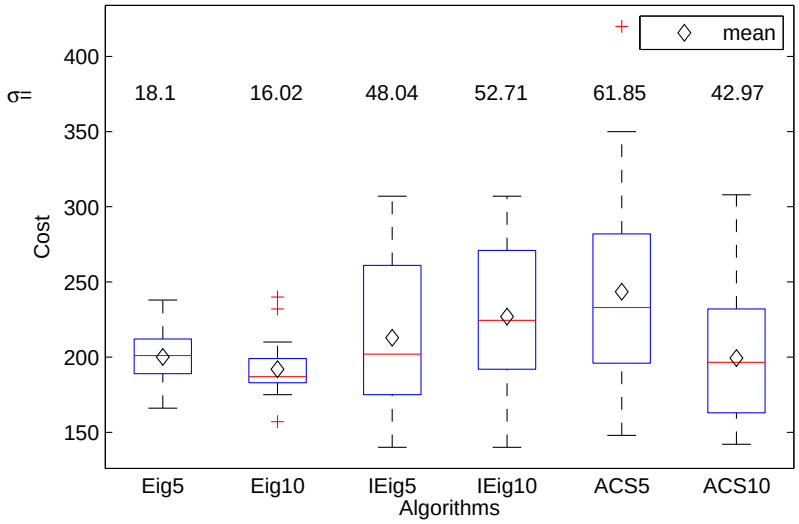

Figure 4.5: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 5 and the repeat of change cycle event 5 (denoted 10) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig10 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 5, which occurs again at the $5 + 5$(period of cycle)$= 10^{th}$ presentation.

IEigenAnt performs worse than the other algorithms in this condition, although it achieved the best cost of 140 in both of the 5^{th} and 10^{th} change cycle events, while EigenAnt demonstrates the superiority over other algorithms especially in the 5^{th} change cycle event. The lower evaporation parameter of EigenAnt $\rho = 0.1$ contradicts the conclusion from (MAVROVOUNIOTIS and YANG, 2013) in which it is claimed that the higher magnitude of change, the higher evaporation rate is needed.

Long Change Cycle Duration Test

We performed the previous experiment for the DRN problem with the change cycle duration of $Du_e = 7200$ cost evaluations in order to compare different algorithms in case of a DOP with longer change cycle duration (lower frequency of dynamics). This problem is easier than the previous one with shorter change cycle duration in the sense that the algorithm has more time to adapt to the change; however, the stagnation challenge in this case might make the problem more difficult to solve.

Fig. 4.6 demonstrates the comparison of EigenAnt, IEigenAnt and ACS algorithms in the first change cycle event (original problem) of DOP with long

Change Cycle Event Number	EigenAnt	IEigenAnt	ACS
1	Mean= 528.5 Best=397	Mean= 258.43 Best= 137	Mean=439.23 Best=254
2	Mean= 419.83 Best=238	Mean=107.7 Best= 80	Mean=380.13 Best=222
3	Mean= 306.4 Best=179	Mean=70.83 Best= 48	Mean=205.1 Best=87
4	Mean= 119.57 Best=65	Mean=76.4 Best= 65	Mean=179.6 Best=76
5	Mean= 200.1 Best=166	Mean= 212.93 Best=140	Mean=243.53 Best= 148
6	Mean= 88.83 Best=65	Mean= 76.93 Best= 65	Mean=191.87 Best=65
7	Mean= 87.5 Best=70	Mean=77.73 Best= 70	Mean=108.7 Best=69
8	Mean= 81.7 Best=69	Mean=68.9 Best= 48	Mean=88.77 Best=**40**
9	Mean= 82.63 Best=65	Mean=76.1 Best= 65	Mean=101.1 Best=65
10	Mean= 191.97 Best=157	Mean= 226.9 Best=140	Mean=199.37 Best= 142

Table 4.3: The Mean and Best values of each change cycle event in the application of EigenAnt, IEigenAnt and ACS to DRN with short change cycle duration and undetectable change time

change cycle duration. In contrast with EigenAnt and ACS, IEigenAnt has equal performance for the first and the repeating change cycle event of the sixth due to the faster convergence speed of IEigenAnt. Table 4.4 exhibits the best and mean value of the test of each algorithm corresponding with each change cycle event. The best value is found by IEigenAnt, whereas ACS could not even achieve 65 for the repeated change cycle event.

Fig. 4.7 demonstrates the results dealing with the second change cycle event and its periodical repeat (7^{th} change cycle event) entitled *increased optimal cost*. The IEigenAnt demonstrates superiority over the other algorithms. For the ACS algorithm, we note from Fig. 4.7 and Table 4.4 that its performance for the repeated change cycle event (7^{th} change cycle event) is worse than the second which is in contrast with the other algorithms and all the previous tests.

Fig. 4.8 demonstrates the results of the algorithms dealing with the third change cycle event and its periodical repeat. In contrast with the short change cycle duration, the ACS has superiority over the other algorithms.

Fig. 4.9 demonstrates the performance of the algorithms dealing with the fourth change cycle event entitled *return of the original state*. IEigenAnt demonstrates its superiority and good performance for both of the change cycle events. Again, we

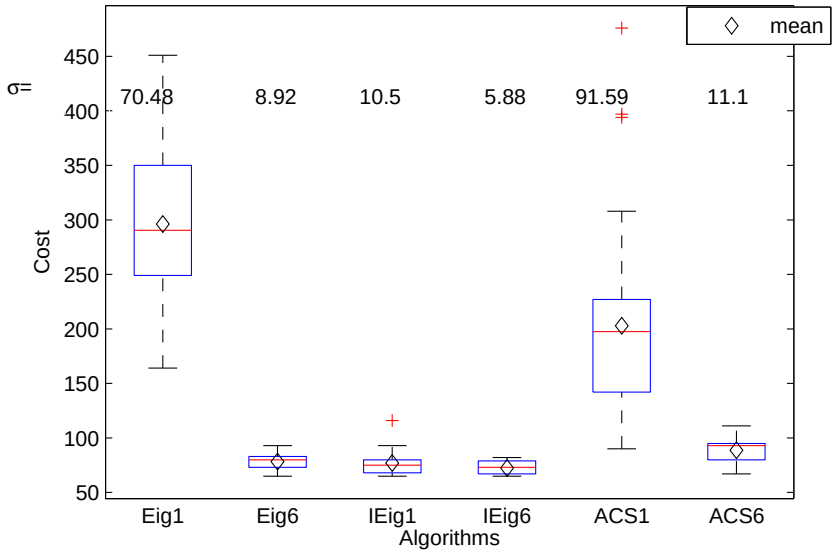

Figure 4.6: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle events 1 and the repeat of change cycle event 1 (denoted 6) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig6 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 1, which occurs again at the $1 + 5$(period of cycle)$= 6^{th}$ presentation.

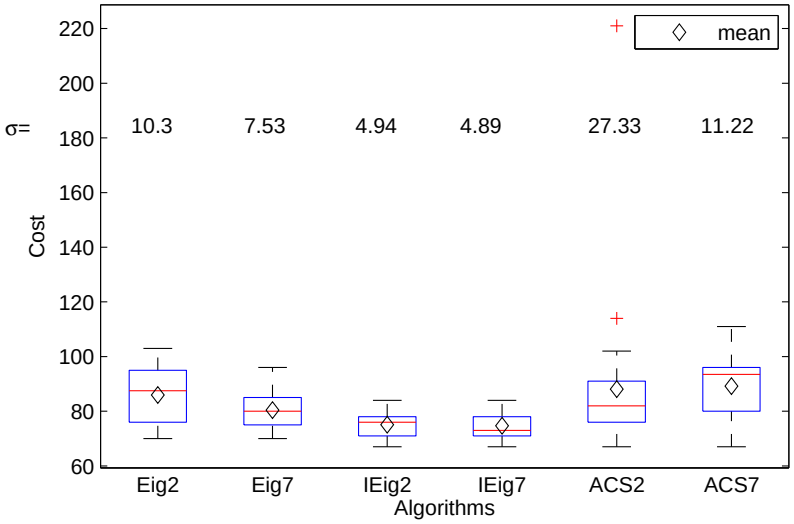

Figure 4.7: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle events 2 and the repeat of change cycle event 2 (denoted 7) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig7 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 2, which occurs again at the $2 + 5$(period of cycle)$= 7^{th}$ presentation.

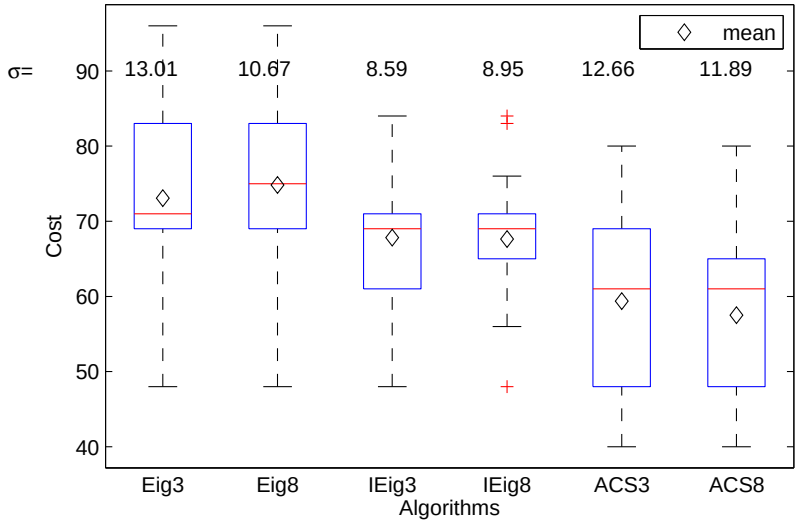

Figure 4.8: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle events 3 and the repeat of change cycle event 3 (denoted 8) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig8 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 3, which occurs again at the $3 + 5$(period of cycle)$= 8^{th}$ presentation.

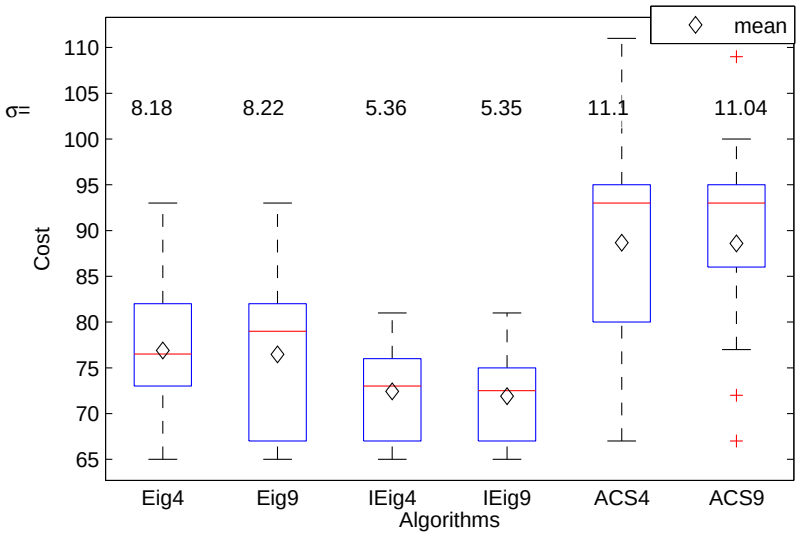

Figure 4.9: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle events 4 and the repeat of change cycle event 4 (denoted 9) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig9 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 4, which occurs again at the $4 + 5$(period of cycle)$= 9^{th}$ presentation.

can notice the decrease in the performance of ACS for the repeated change cycle event (8^{th} change cycle event).

Fig. 4.10 demonstrates the performance of the algorithms dealing with the last change cycle event entitled *radical change*. EigenAnt demonstrates its superiority over the other algorithms in this type of change cycle event. Again, the ACS algorithm showed decrease of performance in the repeated change cycle event.

In a nutshell, CS point of view is suggested for the DOPs in which the change detection is not possible. The EigenAnt, IEigenAnt and ACS algorithms are suggested for such problems due to their acceptable performance from CS point of view and the local pheromone evaporation feature. However, IEigenAnt demonstrates superiority in most cases of change severity, except for problems with radical changes; rather EigenAnt is suggested for such problems. From the results for the long change cycle duration, we conclude that ACS has a stagnation problem with this case and IEigenAnt performs well to such an extent that it performs almost equally well for the first emerging and the periodical repeat change cycle events.

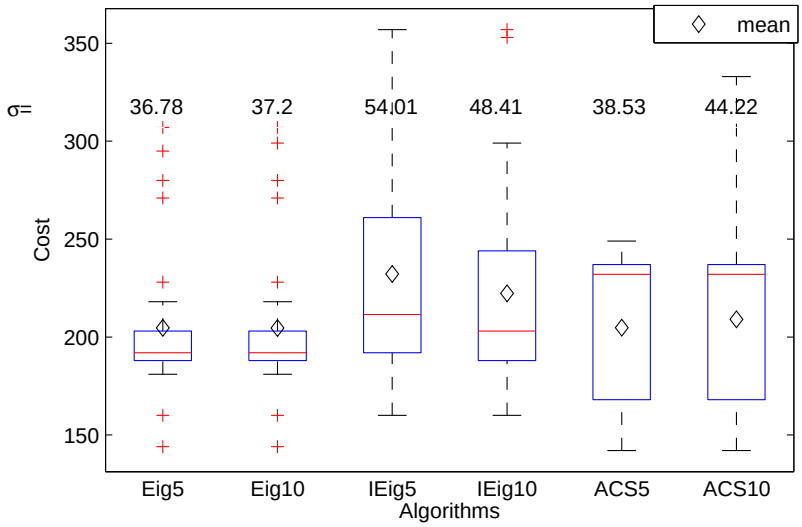

Figure 4.10: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CS point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle events 5 and the repeat of change cycle event 5 (denoted 10) assuming that no information is available about the change time in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig10 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 5, which occurs again at the $5 + 5$(period of cycle)$= 10^{th}$ presentation.

Change Cycle Event Number	EigenAnt	IEigenAnt	ACS
1	Mean= 296.27 Best=164	Mean= 76.97 Best= 65	Mean=202.93 Best=90
2	Mean= 85.97 Best=70	Mean=75.1 Best= 67	Mean=88.1 Best=67
3	Mean= 73.07 Best=48	Mean=67.83 Best= 48	Mean=59.37 Best=**40**
4	Mean= 76.9 Best=65	Mean=72.43 Best= 65	Mean=86.67 Best=67
5	Mean= 204.7 Best=144	Mean= 232.2 Best=160	Mean=204.73 Best= 142
6	Mean= 78.33 Best=65	Mean= 72.47 Best= 65	Mean=88.67 Best=67
7	Mean= 80.4 Best=70	Mean=74.7 Best= 67	Mean=89.2 Best=67
8	Mean= 74.83 Best=48	Mean=67.63 Best= 48	Mean=57.5 Best=**40**
9	Mean= 76.47 Best=65	Mean=71.9 Best= 65	Mean=88.6 Best=67
10	Mean= 204.6 Best=144	Mean= 222.3 Best=160	Mean=209.07 Best= 142

Table 4.4: The Mean and Best values of each change cycle event in the application of EigenAnt, IEigenAnt and ACS to DRN with long change cycle duration and undetectable change time

4.1.2 Solving DRN with Detectable Change Time

Most works in the area of solving DOPs assume that the change time is detectable and algorithms are designed on this basis. For the ACO algorithms, CV point of view, which is known to get better results, can be used since the best-to-date time can be reset before the change. In addition, other strategies, as explained previously, can be used before the change, in order to generate diversity after the change. However, the method used in (EYCKELHOF and SNOEK, 2002) is based on the location of change which is specifically designed for the TSP; hence, it is not applicable to DRN. Moreover, the method used in (ANGUS and HENDTLASS, 2002) is not competitive with the algorithms we are using in our experiment. Thus, we perform an empirical comparison between EigenAnt, IEigenAnt and ACS algorithm without heuristic. For the ACS algorithm, we update the cost of a path constructed through the greedy choice of each edge from the source to the destination node denoted L_{nn} at each change time. For the EigenAnt ($\rho = 0.1$) and IEigenAnt ($\alpha_1 = 0.2$ and $\rho = 0.5$) we use the optimal parameter setting for the CV points of view. Other settings are also the same as previous experiments for the DRN.

Short Change Cycle Duration

We performed the experiments for the DRN with short change cycle duration of $Du_e = 2400$ cost evaluations with the assumption of detecting the change and resting the best-to-date cost at each change time.

Table 4.5 exhibits the mean and best values of each algorithm for each of the 10 change cycle events. We can compare the results of each algorithm in 4.3 with its associated results in the stationary problem from the CV point of view in Table 4.2. Fig. 4.11 demonstrates the box plots of EigenAnt, IEigenAnt and ACS from CV point of view applied to the change cycle events 1 and the repeat of change cycle event 1 (denoted 6) of DRN with detectable change. We can observe the superiority of EigenAnt and IEigenAnt algorithms over the ACS from Fig. 4.11. In fact, the adaptability of an algorithm can be interpreted more clearly by observing the change cycle event 6 (instead of the change cycle event 1) where algorithms should react to the sudden transition of change cycle $5 \longmapsto 6$. We can observe from the Best value of Table 4.5 that EigenAnt and IEigenAnt achieved this best value of 65 for the change cycle event 6. The mean transition of EigenAnt and IEigenAnt are $193.7 \longmapsto 70.03$ and $164.03 \longmapsto 71.67$, respectively. Such a transition together with their small SD values in Fig. 4.11 demonstrate the fast reaction of the EigenAnt and IEigenAnt to the change $5 \longmapsto 6$.

Fig. 4.12 demonstrates box plots of the algorithms associated with change cycle events 2 and the repeat of change cycle event 2 (denoted 7) of the DRN. Both EigenAnt and IEigenAnt demonstrate superiority over the ACS. We can see that all the algorithms require more time to be able to react to the change, and therefore the results for the change cycle event 7 are noticeable in the sense that it gives information about the reaction of each algorithm to change. From Table 4.5 we can observe that only IEigenAnt achieved the Best value of 67 (equal to the value it achieved in the stationary condition). The change from $6 \longmapsto 7$ means that the optimal path of 65 in the change cycle event 6 is not available anymore while the algorithm should look for an alternative path with a slightly larger cost. From the Mean values in the Table 4.5 we can observe the superiority of IEigenAnt over EigenAnt in this sense ($71.17 \longmapsto 71.67$). The mean value 71.67 that IEigenAnt achieves is better than those which EigenAnt and ACS achieve in their stationary results (Fig. 4.1).

Fig. 4.13 demonstrates box plots of the algorithms associated with change cycle events 3 and the repeat of change cycle event 3 (denoted 8) of the DRN. For the more challenging change cycle event 3, we can see that only IEigenAnt demonstrated a weak reaction to the change. We can even observe that IEigenAnt once achieved a path with the cost of 55 which is almost optimal. For the change cycle event 8 we

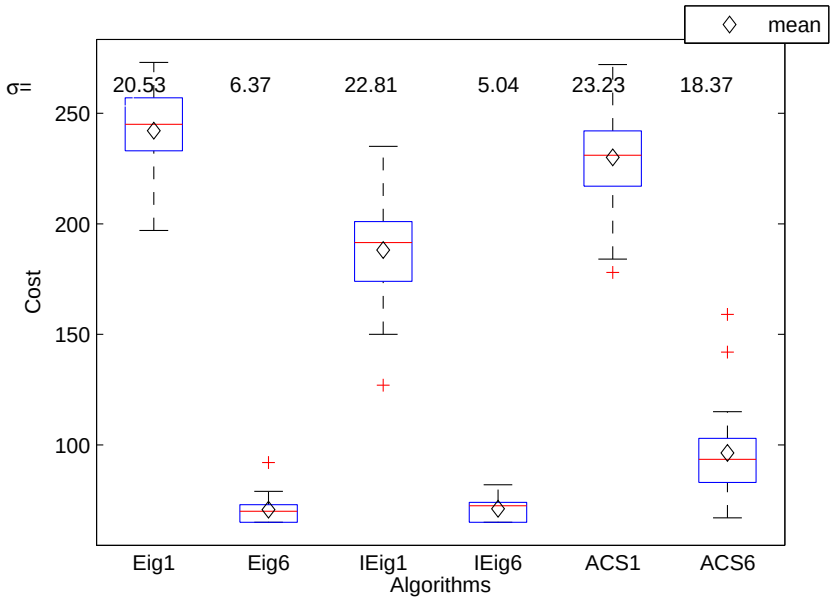

Figure 4.11: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 1 and the repeat of change cycle event 1 (denoted 6) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig6 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 1, which occurs again at the $1 + 5$(period of cycle)$= 6^{th}$ presentation.

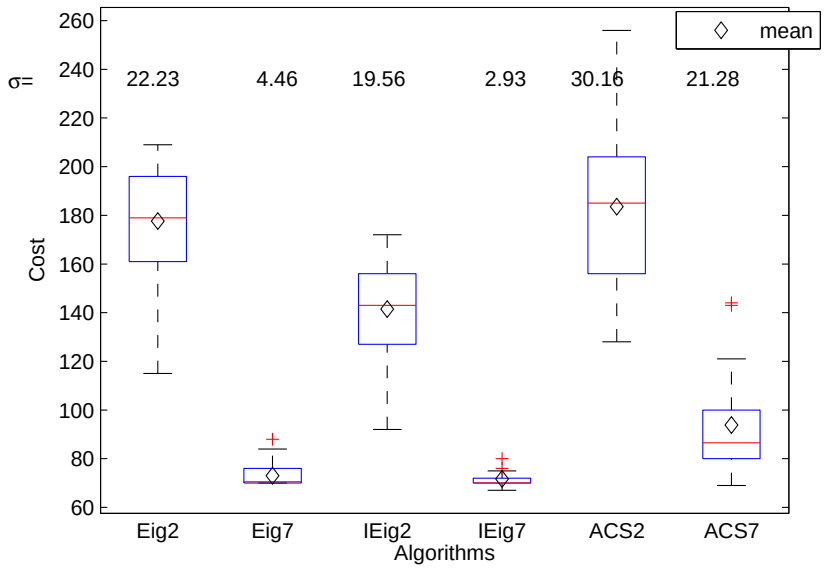

Figure 4.12: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 2 and the repeat of change cycle event 2 (denoted 7) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig7 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 2, which occurs again at the $2 + 5$(period of cycle)$= 7^{th}$ presentation.

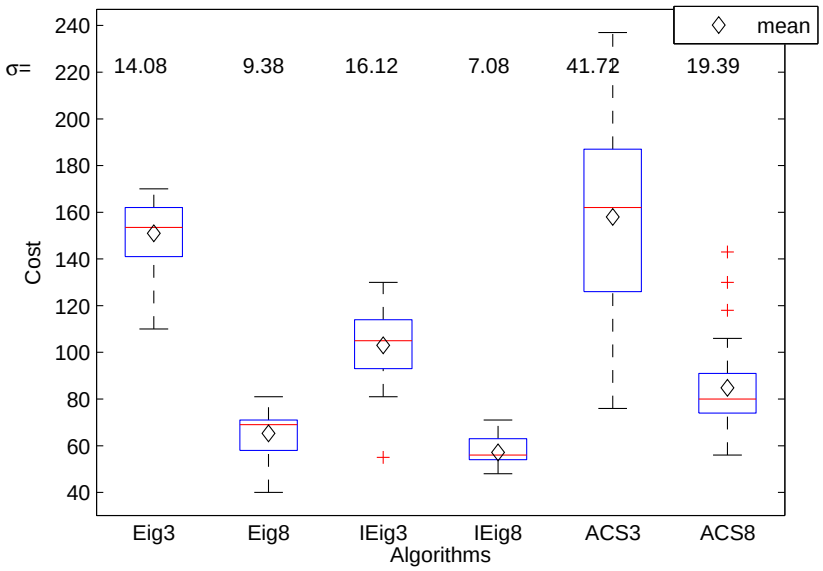

Figure 4.13: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 3 and the repeat of change cycle event 3 (denoted 8) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig8 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 3, which occurs again at the $3 + 5$(period of cycle)$= 8^{th}$ presentation.

can also observe the superiority of IEigenAnt. The reaction of IEigenAnt to change in this change cycle event is so good that although IEigenAnt is incapable of finding the optimal path of 40 even in the stationary problem; it has a noticeable smaller mean than the EigenAnt (Table 4.5).

Fig. 4.14 demonstrates box plots of the algorithms associated with change cycle events 4 and the repeat of change cycle event 4 (denoted 9) of the DRN. Again, only IEigenAnt is capable of reacting to change in the more challenging change cycle event 4; however, it does not achieve the best cost of 65 but achieved the best cost of 67 which is very good comparing with the other algorithms (Table 4.5). For the change cycle event 9, we observe that all the algorithms react to the change. However, IEigenAnt demonstrates superiority while ACS has a weak reaction. The mean value of 67.07 achieved by IEigenAnt (which is smaller than the stationary results of the EigenAnt and ACS) is noticeable in Table 4.5.

Fig. 4.15 demonstrates box plots of the algorithms associated with change cycle events 5 and the repeat of change cycle event 5 (denoted 10) of the DRN. The performance of ACS in reaction to these change cycle events entitled radical change is improved. We can observer that ACS and IEigenAnt react to the change for the

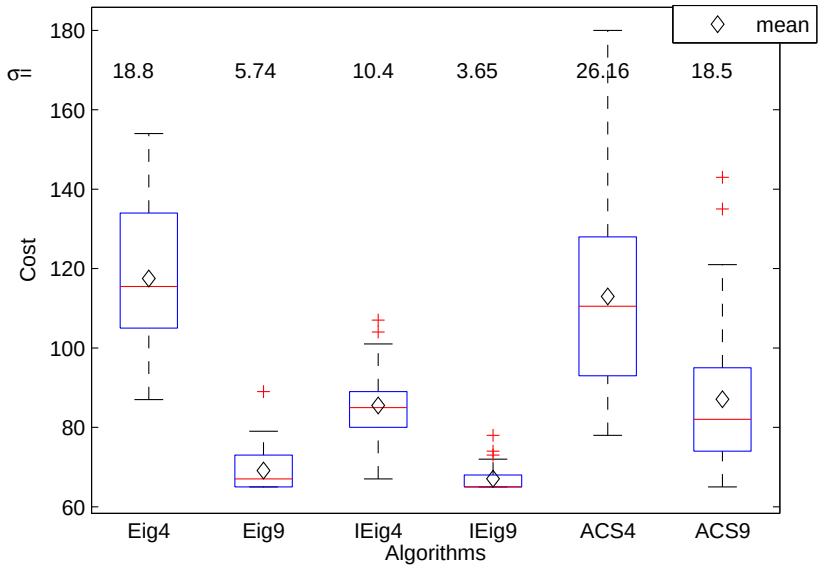

Figure 4.14: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 4 and the repeat of change cycle event 4 (denoted 9) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig9 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 4, which occurs again at the $4 + 5$(period of cycle)$= 9^{th}$ presentation.

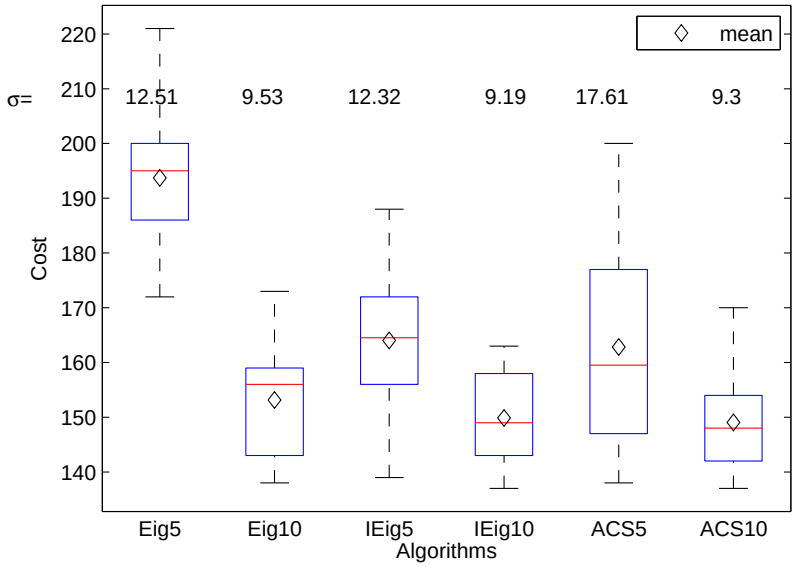

Figure 4.15: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with short change cycle duration. The performance of the algorithms are compared for change cycle events 5 and the repeat of change cycle event 5 (denoted 10) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig10 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 5, which occurs again at the $5 + 5$(period of cycle)$= 10^{th}$ presentation.

change cycle events 5 and 10 with a competitive performance.

Long Change Cycle Duration

Finally, we test the DRN with long change cycle duration of $Du_e = 7200$ cost evaluations with the assumption of detectable change from the CV point of view. Table. 4.6 exhibits the best and mean value of the algorithms EigenAnt, IEigenAnt and ACS for such an experiment.

Fig. 4.16 demonstrates the box plots of each algorithm dealing with the change cycle event 1 and and the repeat of change cycle event 1 (denoted 6). For the change cycle event 1, IEigenAnt demonstrate its superiority due to its faster speed of convergence. The reaction to change of IEIgenAnt also demonstrates superiority over the other algorithms for the change cycle event 6.

Fig. 4.17 demonstrates the box plots of each algorithm dealing with the change cycle event 2 and and the repeat of change cycle event 2 (denoted 7). IEigenAnt demonstrates superiority over the other algorithms for the change cycle events 2 and 7. Also, we can observe that all the algorithms react to the change in the change cycle event 2 since they have more time to react with long change cycle duration.

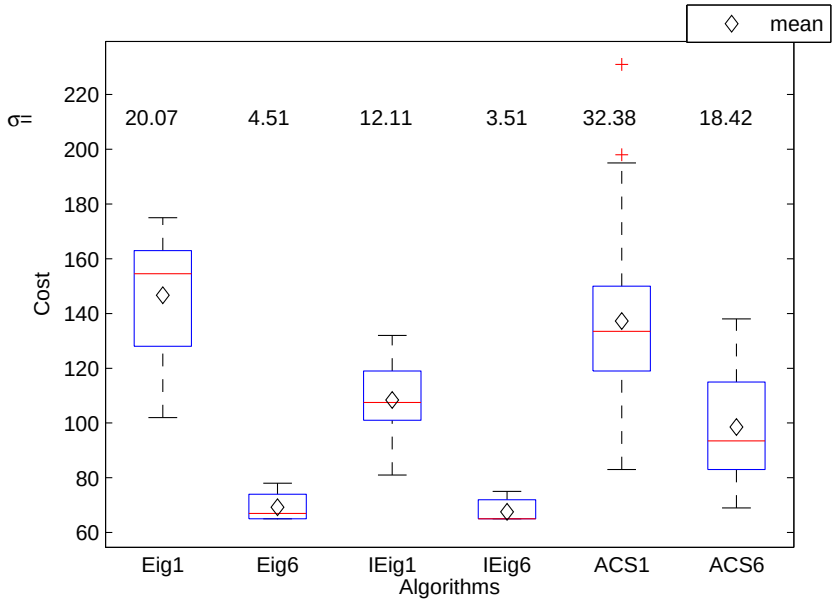

Figure 4.16: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle events 1 and the repeat of change cycle event 1 (denoted 6) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig6 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 1, which occurs again at the $1 + 5$(period of cycle)$= 6^{th}$ presentation.

Change Cycle Event Number	EigenAnt	IEigenAnt	ACS
1	Mean= 242.13 Best=197	Mean= 188.17 Best= 127	Mean= 230.07 Best= 178
2	Mean= 177.67 Best=115	Mean=141.47 Best= 92	Mean= 183.57 Best= 128
3	Mean= 150.97 Best=110	Mean=103.03 Best= 55	Mean= 158 Best= 76
4	Mean= 117.5 Best=87	Mean=85.5 Best= 67	Mean= 112.97 Best= 78
5	Mean= 193.7 Best=172	Mean= 164.03 Best=139	Mean= 162.83 Best= 138
6	Mean= 70.73 Best=65	Mean= 71.17 Best= 65	Mean= 96.4 Best=67
7	Mean= 73.03 Best=70	Mean=71.67 Best= 67	Mean= 93.83 Best= 69
8	Mean= 65.3 Best=**40**	Mean=51.17 Best= 48	Mean= 84.83 Best= 56
9	Mean= 69.1 Best=65	Mean=67.07 Best= 65	Mean= 87.1 Best=65
10	Mean= 153.13 Best=138	Mean= 149.87 Best=137	Mean= 149.07 Best= 137

Table 4.5: The Mean and Best values of each change cycle event in the application of EigenAnt, IEigenAnt and ACS to DRN with short change cycle duration and detectable change time

However, the challenge in DRN with long change cycle duration can be observed from the performance of ACS in which its performance is poorer for the change cycle event 7 than the 2 since ACS encountered the stagnation problem in the transition of change cycle $6 \longmapsto 7$.

Fig. 4.18 demonstrates the box plots of each algorithm dealing with the change cycle event 3 and and the repeat of change cycle event 3 (denoted 8). For the change cycle event 3, all the algorithms react to the change since they have enough time to react due to the longer change cycle duration. Moreover, IEigenAnt is capable of finding the best path of cost 40, hence it demonstrates a significant superiority over the other algorithms. For the change cycle event 8, we can observe that all the algorithms have poorer performance than the change cycle event 3 due to the stagnation problem occurred with longer change cycle duration. Table 4.6 depicts that ACS do not react well to the change due to the stagnation problem while EigenAnt and IEigenAnt still can cope with this problem, and IEigenAnt demonstrates superiority over all the other algorithms.

Fig. 4.19 demonstrates the box plots of each algorithm dealing with the change cycle event 4 and and the repeat of change cycle event 4 (denoted 9). Again we can observe the stagnation problem due to the poorer performance of IEigenAnt and

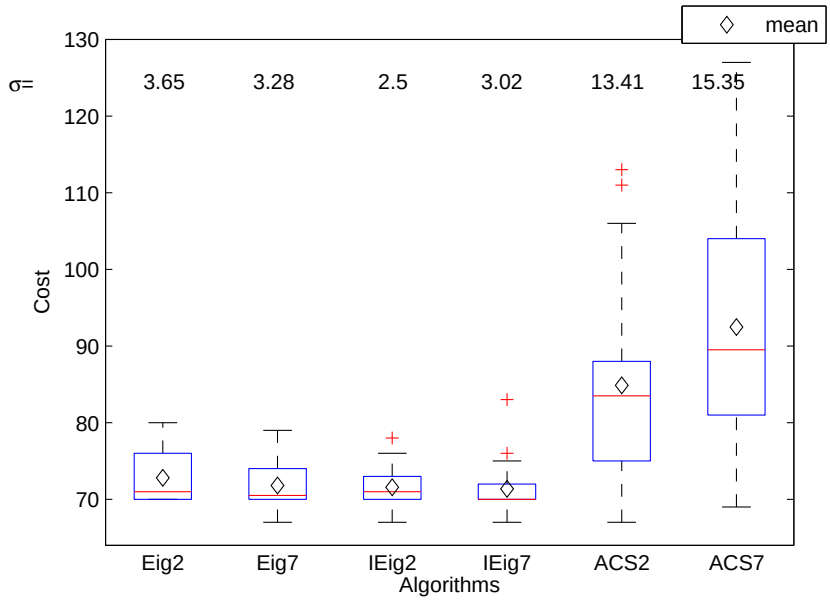

Figure 4.17: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle event 2 and the repeat of change cycle event 2 (denoted 7) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig7 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 2, which occurs again at the $2 + 5$(period of cycle)$= 7^{th}$ presentation.

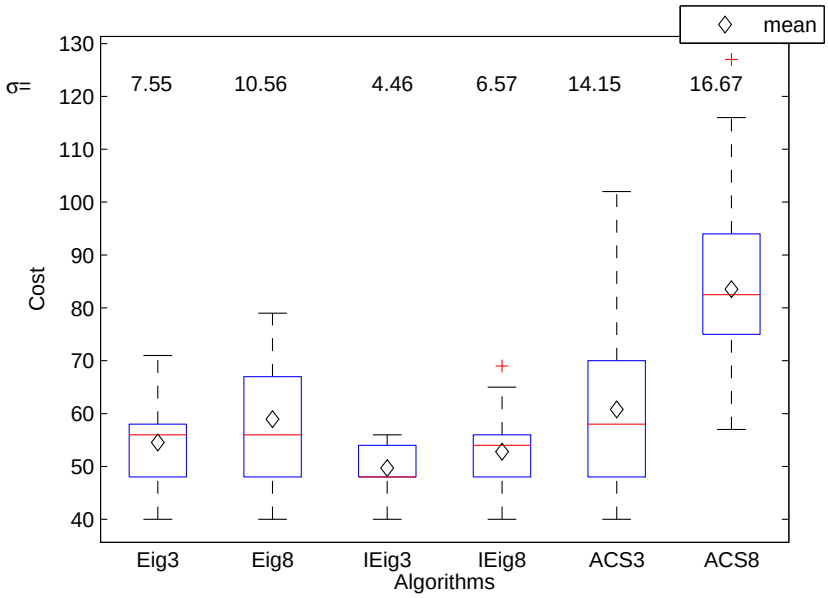

Figure 4.18: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle events 3 and the repeat of change cycle event 3 (denoted 8) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig8 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 3, which occurs again at the $3 + 5$(period of cycle)$= 8^{th}$ presentation.

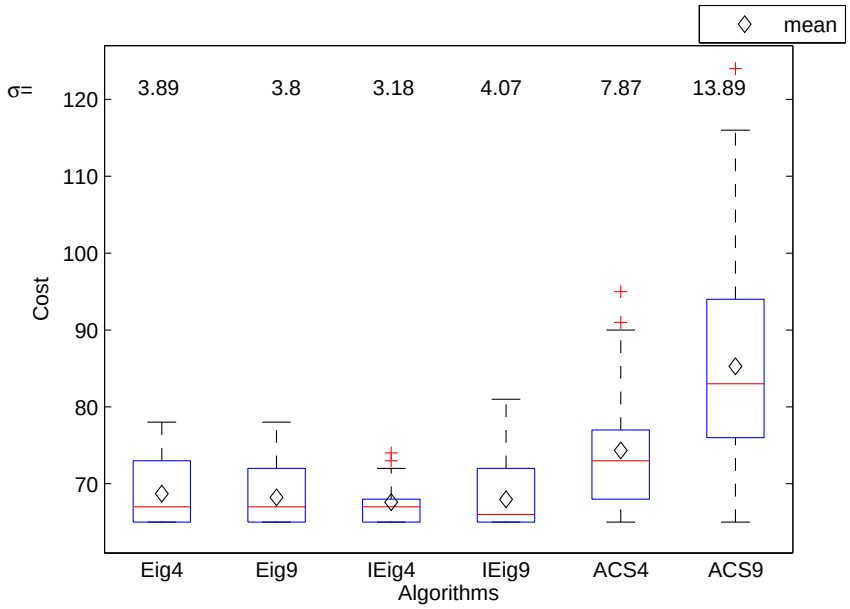

Figure 4.19: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle event 4 and the repeat of change cycle event 4 (denoted 9) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig9 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 4, which occurs again at the $4 + 5$(period of cycle)$= 9^{th}$ presentation.

ACS in change cycle event 9 than the change cycle event 4. However, IEigenAnt still demonstrates superiority for both of the change cycle event 4 and 9. Due to the stagnation problem occurred in IEigenAnt, EigenAnt demonstrates a competitive performance with the IEigenAnt in the change cycle event 9 (Table 4.6).

Fig. 4.20 demonstrates the box plots of each algorithm dealing with the change cycle event 5 and and the repeat of change cycle event 5 (denoted 10). IEigenAnt does not demonstrate a good reaction to these change cycle events entitled radical change while ACS demonstrate a significant superiority. Radical change cycle events are so challenging that it is suggested in (JIN and BRANKE, 2005) to restart the algorithms; hence IEigenAnt encountered a problem dealing with such a change. On the other hand, the ACS which is the worst algorithm for all the changes, demonstrates a great performance with such a change (with the mean almost near its stationary results).

In summary, we observed from the experience of applying the algorithm to DRN with detectable change time that IEigenAnt has a noticeable superiority due to its faster speed and reaction to the change. For the long change cycle duration, we observed the stagnation problem in all the algorithms, however, IEigenAnt handled

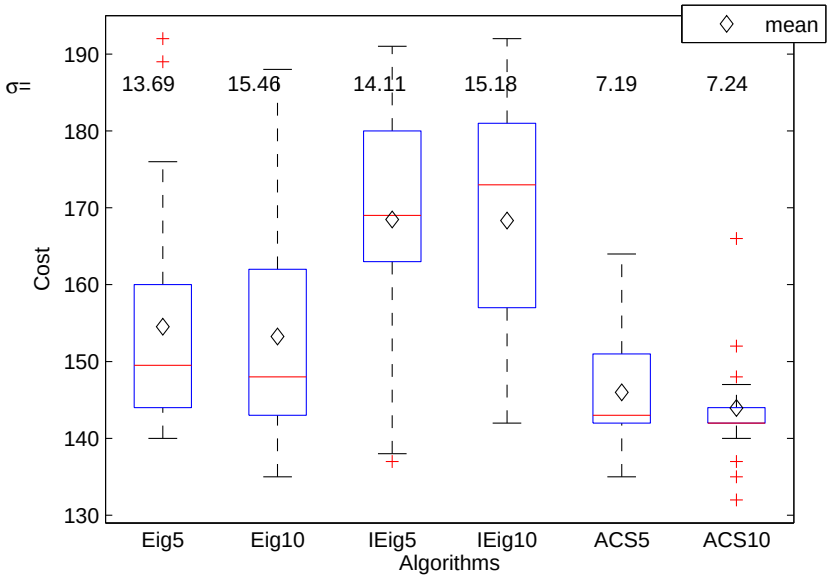

Figure 4.20: Box Plots comparing the results of 30 tests of EigenAnt, IEigenAnt and ACS from CV point of view applied to the DRN with long change cycle duration. The performance of the algorithms are compared for change cycle events 5 and the repeat of change cycle event 5 (denoted 10) assuming that the change time is detectable in DRN. The number of each change cycle event is brought as a suffix to the name of its corresponding algorithm. Thus, for instance, the label IEig10 refers to the IEigenAnt algorithm applied to the second presentation of change cycle event 5, which occurs again at the $5 + 5$(period of cycle)$= 10^{th}$ presentation.

the problems in a way that although it had a poor performance in its second periodical change cycle events than its first periodical change cycle events, the results of IEigenAnt in the second periodical change cycle events were still competitive with its corresponding stationary results.

4.2 IEigenAnt for DMKP

Solving DMKP is a challenge because the change to the constraints will make the feasible solutions infeasible and vice versa. Hence, there are only a few papers on evolutionary algorithms applied to DMKP (UYAR and UYAR, 2009), (BRANKE *et al.*, 2006) and (ÜNAL and KAYAKUTLU, 2016). To the best of our knowledge, only one earlier attacks DMKP with ACO algorithms in (RANDALL, 2005). Moreover, DMKP is also solved with firefly algorithm in (BAYKASOĞLU and OZSOYDAN, 2014). In (BRANKE *et al.*, 2006), the effect of modeling the constraint handling technique is also discussed. It is concluded that for the repair method (entitled real valued representation with weight coding in the paper) explained in Chapter 3, the pseudo-utility value (Eq. (3.6)) should be calculated

Change Cycle Event Number	EigenAnt	IEigenAnt	ACS
1	Mean= 146.67 Best=102	Mean= 103.43 Best= 81	Mean= 137.27 Best= 83
2	Mean= 72.8 Best=70	Mean= 71.57 Best= 67	Mean= 84.87 Best= 67
3	Mean= 54.53 Best=**40**	Mean=49.7 Best= **40**	Mean= 60.8 Best= **40**
4	Mean= 68.7 Best=65	Mean=67.6 Best= 65	Mean= 74.33 Best= 65
5	Mean= 154.53 Best=140	Mean= 168.47 Best=137	Mean= 146 Best= 135
6	Mean= 69.23 Best=65	Mean= 67.57 Best= 65	Mean= 98.53 Best=69
7	Mean= 71.8 Best=67	Mean=71.37 Best= 67	Mean= 92.47 Best= 69
8	Mean= 58.93 Best=**40**	Mean=52.8 Best= **40**	Mean= 83.53 Best= 57
9	Mean= 68.23 Best=65	Mean=67.97 Best= 65	Mean= 85.27 Best=65
10	Mean= 153.27 Best=135	Mean= 168.33 Best=142	Mean= 143.97 Best= 132

Table 4.6: The Mean and Best values of each change cycle event in the application of EigenAnt, IEigenAnt and ACS to DRN with long change cycle duration and detectable change time

after each change which is time consuming and also the algorithm should not only now the change time but also the details of the change in the constraints which is not practical. Using the BCP modeling together with penalty function as explained in Chapter 3 brings an adaptive feature to the ACO family algorithms. Since in (RANDALL, 2005) such a modeling is not used, a type of repair method entitled the solution deconstruction process is suggested to take charge after each change. It should be mentioned that ACS is used in (RANDALL, 2005) due to the local pheromone removal used in the algorithm. As a result, we model our DMKP as an N-node BCP with a penalty function handling method in order to have an adaptability to the change in constraints. We use the adaptive penalty function from (RASHEED, 1998) in which the penalty factors are increased whenever a violation happens and decreased when no violations occur after a defined amount of cost evaluations. As a result, the algorithm has an intrinsic adaptive behavior to the change in constraints.

The change to the MKP can occur from the change of each item's profit, weight of each item associated to a constraint and/or constraint capacity. We compare EigenAnt, IEigenAnt and ACS with each other. We assume that the change time is detectable, and thus we have the results from the CV point of view. We use the

benchmark introduced in (WEINGARTNER and NESS, 1967), with $N = 28$ items and $Z = 2$ number of constraints, that is available in the OR library (BEASLEY, 1990). We use the change cycle events the same as the experiments for DRN except for the periodical repeat of the change cycle events. The details of each change cycle for DMKP is given in Appendix B.

For the adaptive penalty function, the expanded objective function ϕ is written as follows:

$$\phi = \sum_{j=1}^{N} v_j . (1 - x_j) \left(1 + \sum_{z=1}^{Z} \lambda_1 \mu \right) + \sum_{z=1}^{Z} \lambda_2 \mu \tag{4.2}$$

where μ is derived from Eq. (3.13). The violation μ is not dependent on the constraints and is the total violation. Moreover, the penalty factors λ_1 and λ_2 are not dependent on the constraints in contrast with our novel static penalty function in Chapter 3. We choose the initial value of $\lambda_1 = 2$ and $\lambda_2 = 6$ for the penalty factors. After every 200 cost evaluations during which no violation occurs, we perform the following decrease for the penalty factors:

$$\lambda_1 = \lambda_1 - 0.5$$
$$\lambda_2 = \lambda_2 - 1.5$$

In case of a constraint violation the following increase is applied to the penalty factors:

$$\lambda_1 = \lambda_1 + 1$$
$$\lambda_2 = \lambda_2 + 3$$

It can be noted that the adaptive penalty method has the shortcoming of requiring tuning of too many parameters; however, the adaptive method provides an intrinsic reaction to the change in constraints.

The results for the ACS algorithm were so poor that we will not show them. We use the parameter choice ($\alpha_1 = .2$, $\alpha_2 = 1$) for IEigenAnt and the pheromone removal parameter is chosen as $\rho = 0.1$ for both IEigenAnt and EigenAnt. We use the change cycle duration $Due = 4800$ cost evaluations in the following experiments and have the initial pheromone values of 300. Finally, we choose $Q = 320 \times N = 7360$ in Eq. 1.8. The box plots of Comparing EigenAnt with IEigenAnt is shown in Fig. 4.21. Since the goal is to maximize the profit in MKP, the box plots with higher

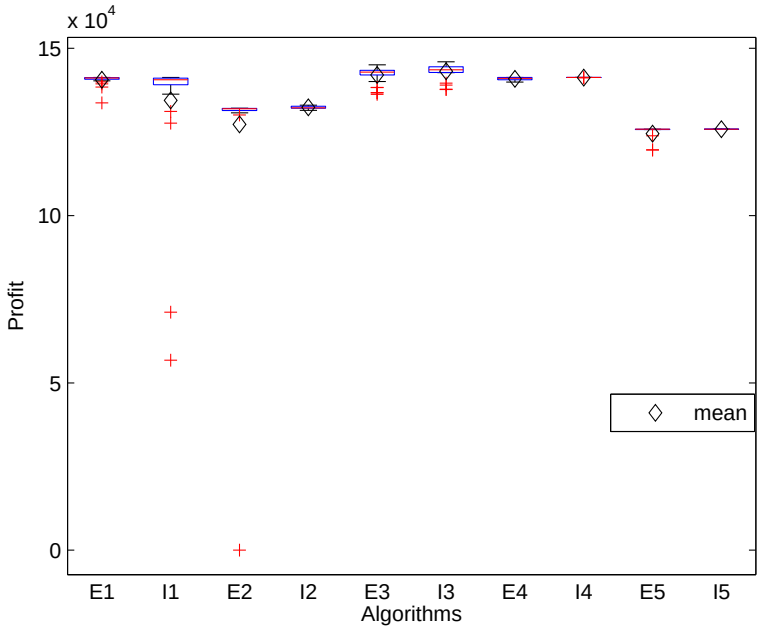

Figure 4.21: Comparing EigenAnt with IEigenAnt for solving DMKP

profit is better. From Fig. 4.21, we can observe the superiority of IEigenAnt, except for the first change cycle event, over EigenAnt (E stands for EigenAnt and I stands for IEigenAnt in the figure).

We apply EigenAnt and EigenAnt for 30 times to the stationary problems of each change cycle event within $5 \times Du = 24000$ cost evaluations. Table 4.7 exhibits the mean and best value that EigenAnt and IEigenAnt achieved for each change cycle event in the stationary mode while Table 4.8 exhibits the results achieved for the DMKP change cycle events. From Tables 4.7 and 4.8, we can observe the acceptable reaction to the change for EigenAnt and IEigenAnt applications to DMKP. It should be mentioned that since MKP is a maximization problem, the larger the solution cost the better.

In this second change cycle event entitled the increased optimal cost, the profit of one item is decreased as $14148 \mapsto 4952$. As a result, the optimal cost of unselected items are increased while the optimal cost of selected items brought in the tables are decreased. The mean solution cost of EigenAnt algorithm applied to the stationary problem of the second change cycle event is 133400 (Table 4.7) while in the DMKP after changing from the original problem it achieves the mean solution cost of 127280 (Table 4.8) which is worse than the solution to the stationary problem. On the other hand, IEigenAnt achieves the mean solution cost of 132010 for the stationary problem (Table 4.7) while in the DMKP problem it achieves the mean solution cost of 132350 (Table 4.8) which is better than its stationary solution while worse than the stationary solution of EigenAnt. We conclude that IEigenAnt has a better

reaction to the change than the EigenAnt in this change cycle event.

In the third change cycle event entitled emergence of new optimal solution, the profit of four items are increased as $\{1898 \mapsto 2164, 440 \mapsto 792, 560 \mapsto 767, 3720 \mapsto 5208\}$, two weight coefficients associated with each of the two constraints are decreased as $\{85 \mapsto 82, 20 \mapsto 4\}$, $\{30 \mapsto 10, 12 \mapsto 8\}$ and both of the constraint capacities are increased as $\{600 \mapsto 738, 600 \mapsto 750\}$. As a result, the cost of optimal solution for the unselected items decreases while the cost of optimal solution for the selected items increases. This case is challenging as it includes all possible type of change that might take place in a DCOP with constraints. For the application of EigenAnt to the third stationary problem the mean solution cost of the achieved optimal solution is 137700 (Table 4.7) while the mean solution cost of optimal solution found by EigenAnt in DMKP after the change from the second optimal solution is 142020 (Table 4.8) which is even better than its stationary solution. For the IEigenAnt application to the stationary problem of the third change cycle event, the mean solution cost of 138700 is achieved (Table 4.7) while for its DMKP application the mean solution cost of 143180 is achieved (Table 4.8). Therefore, EigenAnt and IEigenAnt have a good reaction to the change for this change cycle event while IEigeAnt demonstrates superiority over EigenAnt.

In the fourth change cycle event entitled the return of the original problem, the change is more challenging because all the changes in the third and second change cycle event are undone. The application of EigenAnt to its stationary problem results in the mean solution cost of 140530 (Table 4.7) while the EigenAnt application to DMKP achieves the mean solution cost of 140890 (Table 4.8) which is even better than its stationary solution. For the application of IEigenAnt to the fourth change cycle event, the mean solution cost of 140690 is achieved (Table 4.7) while its application to DMKP achieves the mean solution cost of 141270 (Table 4.8) which is almost near the best found solution with the cost of 141278. Therefore, EigenAnt and IEigenAnt both demonstrate a great performance to the this type of change that usually requires the memory based strategies while IEigenAnt demonstrate an almost ideal performance to this type of change.

In the fifth change cycle event entitled radical change, the profit of one item is decreased as $30800 \mapsto 24948$ and one of the weight coefficients are increased as $0 \mapsto 61$. Therefore, this type of change is less challenging the the second and the fourth change cycle event. This type of change is called radical change in the sense of its radical magnitude of change. The best known optimal solution for this change cycle event is 125821 while the optimal solution of the original problem is 141278 ($142278 - 125821 = 15457$). The application of EigenAnt to the stationary problem achieved the mean solution cost of 125190 (Table 4.7) while its application to DMKP achieves the mean solution cost of 124500 (Table 4.7) which is worse

that its stationary result. The application of IEigenAnt to the stationary problem of the fifth change cycle event achieves the mean solution cost of 124570 (Table 4.7) while the application of IEigenAnt to the DMKP achieves the mean solution cost of 125821 (Table 4.8) which is equal to the best found solution. Therefore, EigenAnt algorithm has a poor performance dealing with this type of change in its application to DMKP than solving its corresponding stationary problem while IEigenAnt algorithm application to DMKP for this type of change cycle event has such a great performance that it tracks the optimal solution accurately.

Change Cycle Event	EigenAnt	IEigenAnt
Original Problem	Mean= 140530 Best=141278	Mean= 140690 Best= 141278
Increased Optimal Cost*	Mean= 133400 Best=133610	Mean= 132010 Best= 133615
Emergence of a New Optimal Solution	Mean= 137700 Best= 147277	Mean=138700 Best= 147277
Return of the Original Problem	Mean= 140530 Best=141278	Mean= 140690 Best= 141278
Radical Change	Mean= 125190 Best=125821	Mean= 124570 Best=125821

* It is true that MKP is a maximization problem; however as explained in Chapter 3, we minimize the total profit of deselected items (Eq. 3.8). In this sense increased optimal cost means that the total profit of deselected items is increased, and thus the profit of selected items is decreased.

Table 4.7: The Mean and Best values of each change cycle event of DMKP in the stationary mode

Change Cycle Event Number	EigenAnt	IEigenAnt
1	Mean= 140630 Best=141278	Mean= 134430 Best= 141278
2	Mean= 127280 Best=132082	Mean= 132350 Best= 133015
3	Mean= 142020 Best= 145084	Mean=143180 Best= 145987
4	Mean= 140890 Best= 141278	Mean=141270 Best= 141278
5	Mean= 124500 Best=125821	Mean= 125821 Best=125821

Table 4.8: The Mean and Best values of each change cycle event in the application of EigenAnt and IEigenAnt to DMKP

4.3 Summary

In this Chapter, we defined concepts regarding DOP and explained that an algorithm dealing with these problems has to track the optimal solution over time. Then, we explained the challenges in this area: increased optimal cost and stagnation.

We explained that IEigenAnt is a good choice for these problems in the sense that it has a CS feature which tackles the increased optimal cost challenge so that the algorithm does not require to detect the change time in order to reset its best-to-date value. Moreover, we explained that IEigenAnt has local pheromone evaporation (pheromone removal) which is useful in the sense of tackling the stagnation challenge to the older problem in DOPs.

The simulations of IEigenAnt, EigenAnt and ACS, that have all the mentioned features of local pheromone evaporation and convergence solution, to DRN problem demonstrated that all the algorithms can perform well when the change time is not detectable. However, IEigenAnt demonstrated superiority over the other algorithms due to its faster convergence property that causes a faster reaction for the IEigenAnt. The faster convergence property of IEigenAnt is attributed to its pheromone amplification parameter smaller than one (α_1). For the DRN problems with detectable change, the superiority of the IEigenAnt was also observable.

On the other hand, we concluded that when the speed dynamic is fast, the algorithms require more time to be able to handle the dynamic problem while for the slower dynamics, the algorithms perform better at the beginning of the problem, since the stagnation occurs after a time.

Finally, we applied the IEigenAnt and EigenAnt to DMKP that is challenging in the sense that the feasibility of the problem is also changing over time. For this reason, we suggested adaptive penalty function as a constraint handling strategy. Both of the algorithms, especially IEigenAnt, demonstrated good performance dealing with our suggested DMKP.

Chapter 5

Contributions, Conclusions and Future Works

5.1 Contributions

The main contributions of the thesis are summarized as follows:

1. We proposed an Improved version of EigenAnt in which the tuning of speed of convergence is decoupled from the convergence of pheromone trails to the local optimal solutions.

2. We extended the stability analysis of the EigenAnt to the N-node binary chain problems through an Improved version of the EigenAnt.

3. We proposed an algorithm entitled Sorting Improved EigenAnt algorithm that can sort the paths between two nodes through a vector of pheromone trails of dimension with equal to the number of paths.

4. A hybrid Randomized Rounding dynamical Max-Min IEigenAnt algorithm was proposed in solving Multidimensional Knapsack problems modeled as a N-node Binary Chain problem in which a novel static penalty method was proposed for constraint handling. The dynamical increase of minimum pheromone trail limit based on the iteration number is also suggested.

5. A scenario for Dynamic Optimization was proposed in which the problem changes in a controlled manner in order to test the power of an algorithm dealing with different challenges might encounter in solving Dynamic Optimization Problems. When the change time is unknown, the increased Optimal solution change is challenging to solve via Ant Colony Optimization Algorithms since they store the best-to-date solution as their final results. Moreover, we included the return of the original problem that usually requires

memory based techniques to resolve. Emergence of a new optimal solution and the most challenging radical change were also included. Finally, all the change cycles were repeated twice in order to monitor performance when dynamics are periodic.

6. We proposed adaptive penalty function method for solving dynamical optimization problems with constraints.

5.2 Conclusions

From the experimental results in Chapter 2, 3 and 4 from the applications of Improved EigenAnt and Sorting Improved EigenAnt, several concluding remarks are drawn as follows:

- Using the Improved EigenAnt algorithm the speed of the algorithm can be tuned while the pheromone trails converges to the one corresponding to the shortest path, whereas, the algorithm does not converge when the speed and convergence parameters are identical and smaller than one.

- IEigenAnt algorithm with the speed tuning parameter $\alpha_1 < 1$ and the convergence tuning parameter $\alpha_2 = 1$ demonstrates a superiority over the other versions of IEigenAnt algorithm in the Routing Network application. This conclusion also verifies the stability analysis and speed of convergence analysis done in Chapter 2.

- IEigenAnt algorithm demonstrated superiority over EigenAnt, Ant Colony System and Ant system algorithms in solving Routing Network problems from the point of view entitled Convergence in Value, in which the best-to-date solution is considered as a final result.

- IEigenAnt algorithm demonstrated superiority over EigenAnt and Ant system algorithms, and competitive results with Ant Colony System algorithm in solving Routing Network problems from the point of view entitled Convergence in Solution in which the construction of a solution based on the choice of pheromone trails with maximum concentration is considered as the final result.

- The hybrid Randomized Rounding dynamical Max-Min IEigenAnt algorithm application to 500 item, 5 constraint Multidimensional Knapsack benchmarks demonstrated competitive results with the best known ACO algorithm. It should be mentioned that our algorithm had better results than its competitors in 2 benchmarks.

- In solving Dynamic Routing Networks, IEigenAnt algorithm demonstrated superiority except for the case that radical change takes place over EigenAnt and Ant Colony System algorithms.

- Convergence in solution point of view resolved the challenge entitled increased optimal solution dealing with the Dynamic Routing Network when the change time is not detectable.

- The number of consecutive iterations until a change takes place, entitled change cycle duration, is critical for an algorithm solving a Dynamic Optimization Problem. When the change cycle duration is short, the reaction of the algorithm to the change should be fast. Hence, ACS and EigenAnt algorithm could only react to the change in the second (periodical) presentation of the changes while IEigenAnt could react even to the first presentation of the changes, due to its faster speed of convergence attributed to the decoupling feature. When the change cycle is long, the challenge is the convergence of the algorithm to the solutions before the future changes take place. ACS demonstrated a stagnation problem in the second presentation of the changes in such a way that its results were near the value of radical change for the second presentation of other changes, which suggests that its pheromone trails were stagnated to the first presentation of radical change cycle. EigenAnt and IEigenAnt could handle this challenge which means that local evaporation in EigenAnt algorithms can handle the stagnation problem more effectively than that of the ACS algorithm.

- IEigenAnt and EigenAnt that used adaptive penalty method were capable of tracking the optimal solution in a Dynamic Multidimensional Knapsack problem in which its constraint parameters change over time. IEigenAnt demonstrated a superiority in solving DMKP. The apex of IEigenAnt performance was in solving the return of original problem in which the profit of 5 items, 4 weight coefficients and 2 constraint capacities were changed. After applying IEigenAnt algorithm to the DMKP for 30 experiments, a mean result of 141270 was achieved in the return of original problem, which is almost equal to the optimal solution of 141278.

5.3 Future Works

For future work, it is suggested to develop the stability analysis from the stochastic perspective. An approach using stochastic approximation theory would give insights about the impacts of parameters on convergence. Theoretical analysis

for the proposed sorting Improved EigenAnt or its extension to larger problems is also suggested. Extending Improved EigenAnt to Multi-Objective Optimization Problems, and then applying it to dynamic Multi-Objective Optimization Problems is also suggested.

Bibliography

ALAYA, I., SOLNON, C., GHÉDIRA, K., "Ant Algorithm for the Multi-dimensional Knapsack Problem". In: *International Conference on Bioinspired Optimization Methods and their Applications (BIOMA)*, pp. 63–72, 2004.

ALI, M., GOLALIKHANI, M., ZHUANG, J., "A Computational Study on Different Penalty Approaches for Solving Constrained Global Optimization Problems with the Electromagnetism-Like Method", *Optimization*, v. 63, n. 3, pp. 403–419, 2014.

ANGUS, D., HENDTLASS, T., "Ant Colony Optimisation Applied to a Dynamically Changing Problem", *Developments in Applied Artificial Intelligence*, pp. 24–38, 2002.

BAYKASOĞLU, A., OZSOYDAN, F. B., "An improved firefly algorithm for solving dynamic multidimensional knapsack problems", *Expert Systems with Applications*, v. 41, n. 8, pp. 3712–3725, 2014.

BEASLEY, J. E., "OR-Library: Distributing Test Problems by Electronic Mail", *Journal of the Operational Research Society*, pp. 1069–1072, 1990.

BIRATTARI, M., CARO, G. D., DORIGO, M., *For a Formal Foundation of the Ant Programming Approach to Combinatorial Optimization - Part 1: The Problem, the Representation, and the General Solution Strategy.* In: Report TR-H-301, ATR Human Information Processing Research Laboratories, 2000.

BONABEAU, E., DORIGO, M., THERAULAZ, G., *Swarm Intelligence: From Natural to Artificial Systems.* N. 1. Oxford University Press, 1999.

BRANKE, J., ORBAYI, M., UYAR, Ş., "The Role of Representations in Dynamic Knapsack Problems". In: *Workshops on Applications of Evolutionary Computation*, pp. 764–775, 2006.

CHU, P. C., BEASLEY, J. E., "A Genetic Algorithm for the Multidimensional Knapsack Problem", *Journal of heuristics*, v. 4, n. 1, pp. 63–86, 1998.

COELLO, C. A. C., "Theoretical and Numerical Constraint-handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art", *Computer Methods in Applied Mechanics and Engineering*, v. 191, n. 11, pp. 1245–1287, 2002.

CVX RESEARCH, I. "CVX: Matlab Software for Disciplined Convex Programming, version 2.0". `http://cvxr.com/cvx`, Aug, 2012.

DENEUBOURG, J.-L., ARON, S., GOSS, S., et al., "The Self-Organizing Exploratory Pattern of the Argentine Ant", *Journal of Insect Behavior*, v. 3, n. 2, pp. 159–168, 1990.

DIJKSTRA, E. W., "A Note on Two Problems in Connexion With Graphs", *Numerische Mathematik*, v. 1, n. 1, pp. 269–271, 1959.

DORIGO, M., GAMBARDELLA, L. M., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, pp. 53–66, 1997.

DORIGO, M., STÜTZLE, T., "An Experimental Study of the Simple Ant Colony Optimization Algorithm". In: *WSES International Conference on Evolutionary Computation (EC'01)*, pp. 253–258, 2001.

DORIGO, M., MANIEZZO, V., COLORNI, A., *Positive Feedback as a Search Strategy*. In: Report 91-016, Laboratorio di Calcolatori Dipartimento di Elettronica Politecnico di Milano, Milan, Italy, 1991.

DORIGO, M., MANIEZZO, V., COLORNI, A., "Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 26, n. 1, pp. 29–41, 1996.

DORIGO, M., BIRATTARI, M., STÜTZLE, T., "Ant Colony Optimization", *IEEE Computational Intelligence Magazine*, v. 1, n. 4, pp. 28–39, 2006.

DUENAS, A., DI MARTINELLY, C., TÜTÜNCÜ, G. Y., "A Multidimensional Multiple-Choice Knapsack Model for Resource Allocation in a Construction Equipment Manufacturer Setting Using an Evolutionary Algorithm". In: *IFIP International Conference on Advances in Production Management Systems*, pp. 539–546. Springer, 2014.

EYCKELHOF, C. J., SNOEK, M., "Ant Systems for a Dynamic TSP". In: *International Workshop on Ant Algorithms*, pp. 88–99. Springer, 2002.

EZZAT, A., ABDELBAR, A. M., WUNSCH, D. C., "A Bare-bones Ant Colony Optimization Algorithm that Performs Competitively on the Sequential Ordering Problem", *Memetic Computing*, v. 6, n. 1, pp. 19–29, 2014.

FIDANOVA, S., "Evolutionary Algorithm for Multiple Knapsack Problem". In: *Seventh International Conference on Parallel Problem Ssolving from Nature (PPSN-VII), Lecture*. Citeseer, 2002.

GRANT, M., BOYD, S., "Graph Implementations for Nonsmooth Convex Programs". In: Blondel, V., Boyd, S., Kimura, H. (Eds.), *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, Springer-Verlag Limited, pp. 95–110, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

GUNTSCH, M., MIDDENDORF, M., SCHMECK, H., "An Ant Colony Optimization Approach to Dynamic TSP". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pp. 860–867. Morgan Kaufmann Publishers Inc., Jul, 2001.

GUTJAHR, W. J., "A Graph-based Ant System And Its Convergence", *Future Generation Computer Systems*, v. 16, n. 8, pp. 873–888, 2000.

GUTJAHR, W. J., "ACO Algorithms with Guaranteed Convergence to the Optimal Solution", *Information Processing Letters*, v. 82, n. 3, pp. 145–153, 2002.

GUTJAHR, W. J., "On the Finite-time Dynamics of Ant Colony Optimization", *Methodology and Computing in Applied Probability*, v. 8, n. 1, pp. 105–133, 2006.

IACOPINO, C., PALMER, P., "The Dynamics of Ant Colony Optimization Algorithms Applied to Binary Chains", *Swarm Intelligence*, v. 6, n. 4, pp. 343–377, 2012.

JAYADEVA, SHAH, S., BHAYA, A., et al., "Ants Find the Shortest Path: A Mathematical Proof", *Swarm Intelligence*, v. 7, n. 1, pp. 43–62, 2013.

JIN, Y., BRANKE, J., "Evolutionary Optimization in Uncertain Environments-a Survey", *IEEE Transactions on Evolutionary Computation*, v. 9, n. 3, pp. 303–317, 2005.

KE, L., FENG, Z., REN, Z., et al., "An Ant Colony Optimization Approach for the Multidimensional Knapsack Problem", *Journal of Heuristics*, v. 16, n. 1, pp. 65–83, 2010.

KONG, M., TIAN, P., KAO, Y., "A New Ant Colony Optimization Algorithm for the Multidimensional Knapsack Problem", *Computers & Operations Research*, v. 35, n. 8, pp. 2672–2683, 2008.

KUMAR, U., *The EigenAnt Algorithm: Extensions, Applications and Hardware Implementation*. PhD Thesis, Indian Institute of Technology, New Delhi, India, Jan, 2016.

LAHAMI, M., KRICHEN, M., BOUCHAKWA, M., et al., "Using Knapsack Problem Model to Design a Resource Aware Test Architecture for Adaptable and Distributed Systems". In: *IFIP International Conference on Testing Software and Systems*, pp. 103–118. Springer, 2012.

LEGUIZAMON, G., MICHALEWICZ, Z., "A New Version of Ant System for Subset Problems". In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, v. 2, pp. 1459–1464. IEEE, 1999.

LORPUNMANEE, S., SAP, M. N., ABDULLAH, A. H., et al., "An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment", *International Journal of Computer and Information Science and Engineering*, v. 1, n. 4, pp. 207–214, 2007.

MAVROVOUNIOTIS, M., YANG, S., "Adapting the Pheromone Evaporation Rate in Dynamic Routing Problems". In: *European Conference on the Applications of Evolutionary Computation*, pp. 606–615. Springer, 2013.

MAVROVOUNIOTIS, M., LI, C., YANG, S., "A Survey of Swarm Intelligence for Dynamic Optimization: Algorithms and Applications", *Swarm and Evolutionary Computation*, v. 33, pp. 1–17, 2017.

MEYER, B., "Convergence Control in ACO". In: *Genetic and Evolutionary Computation Conference (GECCO), Seattle, WA, late-breaking paper available on CD*, 2004.

MEZURA-MONTES, E., COELLO, C. A. C., "Constraint-handling in Nature-inspired Numerical Optimization: Past, Present and Future", *Swarm and Evolutionary Computation*, v. 1, n. 4, pp. 173–194, 2011.

MOHAMMADI, S., SHANG, C., OUHIB, Z., et al., "A computational Study on Different Penalty Approaches for Constrained Optimization in Radiation Therapy Treatment Planning with a Simulated Annealing Algorithm". In: *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on*, pp. 1–6. IEEE, 2015.

MONTEMANNI, R., GAMBARDELLA, L. M., RIZZOLI, A. E., et al., "Ant Colony System for a Dynamic Vehicle Routing Problem", *Journal of Combinatorial Optimization*, v. 10, n. 4, pp. 327–343, 2005.

PIRKUL, H., "A Heuristic Solution Procedure for the Multiconstraint Zero-one Knapsack Problem", *Naval Research Logistics*, v. 34, n. 2, pp. 161–172, 1987.

RAGHAVAN, P., TOMPSON, C. D., "Randomized Rounding: A Technique for Provably Good Algorithms And Algorithmic Proofs", *Combinatorica*, v. 7, n. 4, pp. 365–374, 1987.

RANDALL, M., "A Dynamic Optimisation Approach for Ant Colony Optimisation Using the Multiple Knapsack Problem". In: *The Second Australian Conference on Artificial Life, Sydney*, 2005.

RASHEED, K., "An Adaptive Penalty Approach for Constrained Genetic-algorithm Optimization". In: *Proceedings of the Third Annual Genetic Programming Conference*, pp. 584–590, 1998.

SHAH, S., *Ant Trail Formation: Analysis, Algorithms And Applications.* PhD Thesis, Indian Institute of Technology, New Delhi, India, Jan, 2011.

SHAH, S., JAYADEVA, R. K., KOTHARI, R., et al., "M-Unit EigenAnt: An Ant Algorithm to Find the M Best Solutions." In: *AAAI*, 2011.

SONG, Y., ZHANG, C., FANG, Y., "Multiple Multidimensional Knapsack Problem and Its Applications in Cognitive Radio Networks". In: *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pp. 1–7. IEEE, 2008.

STÜTZLE, T., DORIGO, M., "A Short Convergence Proof for A Class of Ant Colony Optimization Algorithms", *IEEE Transactions on Evolutionary Computation*, v. 6, n. 4, pp. 358–365, 2002.

STÜTZLE, T., HOOS, H. H., "MAX–MIN Ant System", *Future Generation Computer Systems*, v. 16, n. 8, pp. 889–914, 2000.

TINÓS, R., YANG, S., "Analysis of Fitness Landscape Modifications in Evolutionary Dynamic Optimization", *Information Sciences*, v. 282, pp. 214–236, 2014.

ÜNAL, A. N., KAYAKUTLU, G., "A Partheno-genetic Algorithm for Dynamic 0-1 Multidimensional Knapsack Problem", *RAIRO-Operations Research*, v. 50, n. 1, pp. 47–66, 2016.

UYAR, Ş., UYAR, H., "A Critical Look at Dynamic Multi-dimensional Knapsack Problem Generation", *Applications of Evolutionary Computing*, pp. 762–767, 2009.

WEINGARTNER, H. M., NESS, D. N., "Methods for the Solution of the Multidimensional 0/1 Knapsack Problem", *Operations Research*, v. 15, n. 1, pp. 83–103, 1967.

WONG, K. Y., OTHERS, "Parameter Ttuning for Ant Colony Optimization: A Review". In: *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pp. 542–545. IEEE, 2008.

YANG, S., "Non-stationary Problem Optimization Using the Primal-dual Genetic Algorithm". In: *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, v. 3, pp. 2246–2253. IEEE, 2003.

YANG, S., JIANG, Y., NGUYEN, T. T., "Metaheuristics for Dynamic Combinatorial Optimization Problems", *IMA Journal of Management Mathematics*, v. 24, n. 4, pp. 451–480, 2013.

Appendix A

Routing Network Benchmarks

The data for the 10×10 RN benchmarks solved in Chapter 3 and Chapter 4, are given as matrices below in MATLAB notation. Each Matrix represents the edge's costs of the corresponding layer from 1 to 10. Since in the first layer we only have 10 edges as in Fig. 3.1, the first edge cost matrix is actually a vector. We have a 10×10 matrix for the other layers in which each row of the matrix represents the 10 output edges relating to the corresponding node in the previous layer. The Final layer which is not counted as a numbered layer in the model (as explained in Chapter 3) has edges with equal lengths of 1.

A.1 Original Problem

The Dijkstra's algorithm applied to the problem results in the optimal solution with the cost of 65.

```
D=[23 84 57 78 50 10 61 30 74 81],...
[40 85 27 17 2 26 46 64 7 26;86 17 81 2 58 10 46 77 66 54;...
24 15 33 86 83 12 9 91 45 33;44 32 50 59 4 37 87 86 12 35;...
84 96 37 45 92 29 20 68 82 66;71 65 47 77 45 57 66 74 58 58;...
11 53 49 49 60 65 61 64 94 55;42 55 91 7 5 21 54 32 45 39;...
30 39 96 41 54 49 16 67 41 37;49 36 13 67 38 81 54 56 84 18]...
,[59 69 37 34 55 90 34 62 28 10;19 59 94 94 18 7 8 27 11 26;...
61 52 90 2 9 17 46 8 33 46;32 59 55 90 85 17 67 33 40 75;...
37 22 48 6 90 25 38 77 18 34;91 39 20 48 76 55 63 71 69 75;...
82 42 45 12 20 58 63 77 55 68;11 39 70 36 36 66 66 37 70 55;...
95 100 80 55 34 9 51 27 7 51;53 58 27 44 16 16 9 26 27 2]...
,[40 83 27 24 53 19 19 76 38 19;71 79 56 16 91 34 16 97 9 87;...
99 22 58 40 25 69 30 56 65 42;98 52 56 5 79 85 43 13 2 39;...
73 11 43 60 16 38 16 71 16 92;89 93 16 20 28 31 99 64 98 44;...
```

```
5 60 64 62 40 58 49 22 4 30;47 45 53 31 27 5 45 28 28 75;...
97 9 9 1 59 99 35 6 32 98;94 82 25 35 62 56 80 3 38 52]...
,[2 81 81 8 38 60 78 85 16 4;58 9 95 23 71 82 18 3 40 99;...
12 33 18 40 89 60 9 66 97 4;67 65 85 19 45 58 25 30 10 13;...
51 22 77 74 47 98 35 79 8 76;36 44 47 21 6 40 76 32 87 15;...
34 84 15 81 92 64 41 83 19 21;45 84 15 79 40 67 35 4 24 65;...
16 96 73 14 73 83 46 21 85 17;46 34 67 84 65 77 34 23 78 32]...
,[28 81 65 46 91 68 22 86 94 74;3 71 96 29 92 37 82 76 88 41;...
53 80 20 55 80 17 38 57 28 87;96 66 42 42 15 57 44 30 95 4;...
93 31 13 33 94 73 13 39 51 20;69 46 25 82 50 91 27 18 91 76;...
56 67 78 54 29 35 4 46 66 80;32 76 11 4 39 15 10 76 89 91;...
86 25 23 51 32 81 55 16 65 44;71 27 61 24 15 98 80 82 85 71]...
,[100 45 81 22 23 77 87 61 98 74;94 4 40 9 35 9 61 23 63 51;...
82 75 18 94 47 87 24 59 82 38;78 24 56 12 24 80 85 46 49 78;...
71 54 63 45 27 30 3 2 100 77;85 30 32 68 50 7 65 24 83 50;...
2 3 8 54 67 35 23 69 21 76;28 12 24 54 80 7 19 80 71 88;...
22 22 70 13 52 14 21 64 74 59;49 18 7 94 38 84 38 9 75 83]...
,[14 17 90 85 90 15 6 25 71 65;31 60 4 11 90 19 63 95 21 15;...
22 2 50 38 31 26 21 16 68 32;93 15 29 38 5 22 67 35 22 43;...
36 55 68 64 47 86 21 22 42 6;84 36 40 25 69 82 38 54 20 37;...
62 47 43 3 78 77 77 4 3 15;28 90 79 62 15 94 25 3 55 35;...
64 56 53 24 18 61 61 53 68 50;80 62 46 36 97 1 93 3 92 38]...
,[11 78 82 94 20 83 57 6 64 57;10 49 19 15 93 35 16 1 5 52;...
4 45 68 26 13 99 91 52 12 1;83 4 24 44 34 55 48 40 47 71;...
88 59 87 43 49 40 23 99 81 5;48 91 80 60 69 36 19 63 70 70;...
25 49 28 19 74 100 15 16 95 93;25 15 86 8 40 53 78 6 54 99;...
10 30 10 71 58 69 41 66 56 49;22 22 18 62 19 52 3 36 20 31]...
,[57 81 91 87 62 61 91 60 95 1;29 36 55 11 1 37 94 61 83 84;...
56 6 16 42 1 28 99 24 69 40;44 89 6 26 38 70 42 11 79 90;...
87 69 69 33 17 33 13 24 23 93;27 73 20 36 91 56 97 96 28 38;...
67 2 28 25 73 75 77 8 28 28;42 31 28 46 10 37 13 50 3 53;...
29 32 51 38 8 37 43 44 1 14;98 38 26 58 68 76 88 1 78 37]...
,ones(10,1);
```

A.2 Increased Optimal Cost

The Dijkstra's algorithm applied to this problem results in the optimal solution with a cost of 67. We generated the increased optimal cost problem by increasing the cost of some randomly selected edges from the original problem. The edges in red

color are the changed edges.

```
D=[23 84 57 78 50 10 61 30 74 81],[40 85 27 17 2 26 46 64 7 26;...
86 17 81 2 58 10 46 77 66 54;24 15 33 86 83 12 9 91 45 33;...
44 32 50 59 4 37 87 86 12 35;84 96 37 45 92 29 20 68 82 66;...
71 65 47 77 45 57 66 74 58 58;11 53 49 49 60 65 61 64 94 55;...
42 55 91 7 5 21 54 32 45 39;30 39 96 41 54 49 16 67 41 37;...
49 36 13 67 38 81 54 56 84 18],[59 69 37 34 55 90 34 62 28 10;...
19 59 94 94 18 7 8 27 11 26;61 52 90 2 9 17 46 8 33 46;...
32 59 55 90 85 17 67 33 40 75;37 22 48 9 90 25 38 77 18 34;...
91 39 20 48 76 55 63 71 69 75;82 42 45 12 20 58 63 77 55 68;...
11 39 70 36 36 66 66 37 70 55;95 100 80 55 34 9 51 27 7 51;...
53 58 27 44 16 16 9 26 27 2],[40 83 27 24 53 19 19 76 38 19;...
71 79 56 16 91 34 16 97 9 87;99 22 58 40 25 69 30 56 65 42;...
98 52 56 5 79 85 43 13 2 39;73 11 43 60 16 38 16 71 16 92;...
89 93 16 20 28 31 99 64 98 44;5 60 64 62 40 58 49 22 4 30;...
47 45 53 31 27 5 45 28 28 75;97 9 9 1 59 99 35 6 32 98;...
94 82 25 35 62 56 80 3 38 52],[2 81 81 8 38 60 78 85 16 4;...
58 9 95 23 71 82 18 3 40 99;12 33 18 40 89 60 9 66 97 4;...
67 65 85 19 45 58 25 30 10 13;51 22 77 74 47 98 35 79 8 76;...
36 44 47 21 6 40 76 32 87 15;34 84 15 81 92 64 41 83 19 21;...
45 84 15 79 40 67 35 4 24 65;16 96 73 16 73 83 46 21 85 17;...
46 34 67 84 65 77 34 23 78 32],[28 81 65 46 91 68 22 86 94 74;...
3 71 96 29 92 37 82 76 88 41;53 80 20 55 80 17 38 57 28 87;...
96 66 42 42 15 57 44 30 95 4;93 31 13 33 94 73 13 39 51 20;...
69 46 25 82 50 91 27 18 91 76;56 67 78 54 29 35 4 46 66 80;...
32 76 11 4 39 15 10 76 89 91;86 25 23 51 32 81 55 16 65 44;...
71 27 61 24 15 98 80 82 85 71],[100 45 81 22 23 77 87 61 98 74;...
94 4 40 9 35 9 61 23 63 51;82 75 18 94 47 87 24 59 82 38;...
78 24 56 12 24 80 85 46 49 78;71 54 63 45 27 30 3 2 100 77;...
85 30 32 68 50 7 65 24 83 50;2 3 8 54 67 35 23 69 21 76;...
28 12 24 54 80 7 19 80 71 88;22 22 70 13 52 14 21 64 74 59;...
49 18 7 94 38 84 38 9 75 83],[14 17 90 85 90 15 6 25 71 65;...
31 60 4 11 90 19 63 95 21 15;22 2 50 38 31 26 21 16 68 32;...
93 15 29 38 5 22 67 35 22 43;36 55 68 64 47 86 21 22 42 6;...
84 36 40 25 69 82 38 54 20 37;62 47 43 3 78 77 77 4 3 15;...
28 90 79 62 15 94 25 3 55 35;64 56 53 24 18 61 61 53 68 50;...
80 62 46 36 97 1 93 3 92 38],[11 78 82 94 20 83 57 6 64 57;...
10 49 19 15 93 35 16 1 5 52;4 45 68 26 13 99 91 52 12 1;...
83 4 24 44 34 55 48 40 47 71;88 59 87 43 49 40 23 99 81 5;...
```

```
48 91 80 60 69 36 19 63 70 70;25 49 28 19 74 100 15 16 95 93;...
25 15 86 8 40 53 78 6 54 99;10 30 10 71 58 69 41 66 56 49;...
22 22 18 62 19 52 3 36 20 31],[57 81 91 87 62 61 91 60 95 1;...
29 36 55 11 1 37 94 61 83 84;56 6 16 42 1 28 99 24 69 40;...
44 89 6 26 38 70 42 11 79 90;87 69 69 33 17 33 13 24 23 93;...
27 73 20 36 91 56 97 96 28 38;67 2 28 25 73 75 77 8 28 28;...
42 31 28 46 10 37 13 50 3 53;29 32 51 38 8 37 43 44 1 14;...
98 38 26 58 68 76 88 1 78 37],ones(10,1);
```

A.3 Emergence of a New Optimal Solution

In order to achieve the emergence of new optimal solution we decreased some edge's costs which are marked in red. Dijkstra's algorithm achieved the optimal cost of 48 while the known optimal cost by ACO and EigenAnt algorithms is 40. The edges with the cost of zero caused the Dijkstra algorithm to assume the corresponding edges to be unreachable and thus failed to find the optimal solution.

```
D=[23 84 57 78 50 10 61 30 74 81],[40 85 2 17 2 26 46 64 7 26;...
86 17 81 2 58 10 46 77 66 54;24 15 33 86 83 12 9 91 45 33;...
44 32 50 59 4 37 87 86 12 35;84 96 37 45 92 29 20 68 82 66;...
71 65 47 77 45 57 66 74 58 58;11 53 49 49 60 65 61 64 94 55;...
42 55 91 7 5 21 54 32 45 39;30 39 96 41 54 49 16 67 41 37;...
49 36 13 67 38 81 54 56 84 18],[59 69 37 34 55 90 34 62 28 10;...
19 59 94 94 18 7 8 27 11 26;61 52 90 1 9 17 46 8 33 46;...
32 59 55 90 85 17 67 33 40 75;37 22 48 9 90 25 38 77 18 34;...
91 39 20 48 76 55 63 71 69 75;82 42 45 12 20 58 63 77 55 68;...
11 39 70 36 36 66 66 37 70 55;95 100 80 55 34 9 51 27 7 51;...
53 58 27 44 16 16 9 26 27 2],[40 83 27 24 53 19 19 76 38 19;...
71 79 56 16 91 34 16 97 9 87;99 22 58 40 25 69 30 56 65 42;...
98 52 56 5 79 85 43 13 1 39;73 11 43 60 16 38 16 71 16 92;...
89 93 16 20 28 31 99 64 98 44;5 60 64 62 40 58 49 22 4 30;...
47 45 53 31 27 5 45 28 28 75;97 9 9 1 59 99 35 6 32 98;...
94 82 25 35 62 56 80 3 38 52],[2 81 81 8 38 60 78 85 16 4;...
58 9 95 23 71 82 18 3 40 99;12 33 18 40 89 60 9 66 97 4;...
67 65 85 19 45 58 25 30 10 13;51 22 77 74 47 98 35 79 8 76;...
36 44 47 21 6 40 76 32 87 15;34 84 15 81 92 64 41 83 19 21;...
45 84 15 79 40 67 35 4 24 65;16 96 73 16 73 83 46 3 85 17;...
46 34 67 84 65 77 34 23 78 32],[28 81 65 46 91 68 22 86 94 74;...
3 71 96 29 92 37 82 76 88 41;53 80 20 55 80 17 38 57 28 87;...
```

```
96 66 42 42 15 57 44 30 95 4;93 31 13 33 94 73 13 39 51 20;...
69 46 25 82 50 91 27 18 91 76;56 67 78 54 29 35 4 46 66 80;...
32 76 11 4 39 15 3 76 89 91;86 25 23 51 32 81 55 16 65 44;...
71 27 61 24 15 98 80 82 85 71],[100 45 81 22 23 77 87 61 98 74;...
94 4 40 9 35 9 61 23 63 51;82 75 18 94 47 87 24 59 82 38;...
78 24 56 12 24 80 85 46 49 78;71 54 63 45 27 30 3 2 100 77;...
85 30 32 68 50 7 65 24 83 50;2 0 8 54 67 35 23 69 21 76;...
28 12 24 54 80 7 19 80 71 88;22 22 70 13 52 14 21 64 74 59;...
49 18 7 94 38 84 38 9 75 83],[14 17 90 85 90 15 6 25 71 65;...
31 60 4 11 90 19 63 95 21 10;22 2 50 38 31 26 21 16 68 32;...
93 15 29 38 5 22 67 35 22 43;36 55 68 64 47 86 21 22 42 6;...
84 36 40 25 69 82 38 54 20 37;62 47 43 3 78 77 77 4 3 15;...
28 90 79 62 15 94 25 3 55 35;64 56 53 24 18 61 61 53 68 50;...
80 62 46 36 97 1 93 3 92 38],[11 78 82 94 20 83 57 6 64 57;...
10 49 19 15 93 35 16 1 5 52;4 45 68 26 13 99 91 52 12 1;...
83 4 24 44 34 55 48 40 47 71;88 59 87 43 49 40 23 99 81 5;...
48 91 80 60 69 36 19 63 70 70;25 49 28 19 74 100 15 16 95 93;...
25 15 86 8 40 53 78 6 54 99;10 30 10 71 58 69 41 66 56 49;...
22 22 18 62 19 52 0 36 20 31],[57 81 91 87 62 61 91 60 95 1;...
29 36 55 11 1 37 94 61 83 84;56 6 16 42 1 28 99 24 69 40;...
44 89 6 26 38 70 42 11 79 90;87 69 69 33 17 33 13 24 23 93;...
27 73 20 36 91 56 97 96 28 38;67 2 28 25 73 75 77 8 10 28;...
42 31 28 46 10 37 13 50 3 53;29 32 51 38 8 37 43 44 1 14;...
98 38 26 58 68 76 88 1 78 37],ones(10,1);
```

A.4 Radical Change

In order to achieve the radical change, the cost of more edges is increased with a larger magnitude of change. As a result Dijkstra's algorithm finds the optimal value of 132 for this problem. The changed edges are marked in red.

```
D=[73 84 57 78 50 46 61 117 74 81],[40 85 27 17 2 26 46 64 7 26;...
86 17 81 2 58 10 46 77 66 54;24 27 33 86 83 12 9 91 45 33;...
44 32 50 59 4 37 87 86 12 35;84 96 37 45 92 29 20 68 82 66;...
71 65 85 77 45 57 66 74 58 58;11 53 49 49 60 65 61 64 94 55;...
42 55 91 7 5 21 54 32 45 39;30 39 96 41 54 49 16 67 41 37;...
49 36 13 67 38 81 54 56 84 18],[59 69 37 34 55 90 34 62 28 10;...
19 59 94 94 18 7 8 27 11 26;61 52 90 2 9 17 46 8 33 46;...
32 59 55 90 85 17 67 33 40 75;37 22 48 51 90 25 38 77 75 34;...
```

```
91 39 20 48 76 55 63 71 69 75;82 42 45 12 20 58 63 77 55 68;...
11 39 70 36 36 66 66 37 70 55;95 100 80 55 34 9 51 27 7 51;...
53 58 27 44 16 16 9 26 27 2],[40 83 27 24 53 19 19 76 38 19;...
71 79 56 16 91 34 16 97 9 87;99 22 58 40 25 69 30 56 65 42;...
98 52 56 5 79 85 43 80 15 39;73 24 43 60 16 38 16 71 16 92;...
89 93 16 20 28 31 99 64 98 44;5 60 64 62 40 58 49 22 4 30;...
47 45 53 31 27 7 45 28 28 75;97 19 9 1 59 99 35 6 32 98;...
94 82 25 35 62 56 80 6 38 52],[2 81 81 8 38 60 78 85 16 4;...
58 9 95 23 71 82 18 19 40 99;12 33 18 40 89 60 74 66 97 4;...
67 65 85 19 45 58 25 30 10 13;51 22 77 74 47 98 35 79 8 76;...
36 44 47 21 8 40 76 32 87 15;34 84 15 81 92 64 41 83 19 21;...
45 84 15 79 40 67 35 40 24 65;16 96 73 14 73 83 46 21 85 17;...
46 34 67 84 65 77 34 23 78 32],[28 81 65 46 91 68 22 86 94 74;...
3 71 96 29 92 37 82 76 88 41;53 80 20 55 80 17 38 57 28 87;...
96 66 42 42 44 57 44 30 95 12;93 31 13 33 94 73 13 39 51 20;...
69 46 25 82 50 91 27 18 91 76;56 301 78 54 29 35 4 46 66 80;...
32 76 11 4 39 15 68 76 89 91;86 25 23 51 32 81 55 16 65 44;...
71 27 61 24 24 98 80 82 85 71],[100 45 81 22 23 77 87 61 98 74;...
94 4 40 9 35 9 61 23 63 51;82 75 18 94 47 87 24 59 82 38;...
78 24 56 12 24 80 85 46 49 78;71 54 63 45 27 30 3 3 100 77;...
85 30 32 68 50 7 65 24 83 50;2 20 25 54 67 35 23 69 21 76;...
28 12 24 54 80 7 19 80 71 88;22 22 70 13 52 14 21 64 74 59;...
49 18 7 94 38 84 38 9 75 83],[14 17 90 85 90 15 6 25 71 65;...
31 60 6 11 90 19 63 95 21 15;22 5 50 38 31 26 21 16 68 32;...
93 15 29 38 5 22 67 35 22 43;36 55 68 64 47 86 21 22 42 6;...
84 36 40 25 69 82 38 54 20 37;62 47 43 3 78 77 77 4 3 15;...
28 90 79 62 15 94 25 9 55 35;64 56 53 24 18 61 61 53 68 50;...
80 62 46 36 97 1 93 3 92 38],[11 78 82 94 20 83 57 6 64 57;...
10 49 19 15 93 35 16 7 7 52;4 45 68 26 13 99 91 52 12 3;...
83 4 24 44 34 55 48 40 47 71;88 59 87 43 49 40 23 99 81 5;...
48 91 80 60 69 36 19 63 70 70;25 49 28 19 74 100 15 16 95 93;...
25 15 86 8 40 53 78 6 54 99;10 30 10 71 58 69 41 66 56 49;...
22 22 18 62 19 52 3 36 20 31],[57 81 91 87 62 61 91 60 95 1;...
29 36 55 11 7 37 94 61 83 84;56 6 16 42 1 28 99 24 69 40;...
44 89 6 26 38 70 42 11 79 90;87 69 69 33 17 33 13 24 23 93;...
27 73 20 36 91 56 97 96 28 38;67 2 28 25 73 75 77 8 28 28;...
42 31 28 46 10 37 13 50 5 53;29 32 51 38 8 37 43 44 1 14;...
98 38 26 58 68 76 88 1 78 37],ones(10,1);
```

Appendix B

DMKP Benchmarks

B.1 Original Benchmark

Weing1 benchmark from the OR library is with the following Matlab format and the optimal value of 141278.

```
v=[ 1898 440 22507 270 14148 3100 4650 30800 615 4975 ...
 1160 4225 510 11880 479 440 490 330 110 560 ...
 24355 2885 11748 4550 750 3720 1950 10500];
w=[45 0 85 150 65 95 30 0 170 0 ...
 40 25 20 0 0 25 0 0 25 0 ...
 165 0 85 0 0 0 0 100 ];
a=[ 30 20 125 5 80 25 35 73 12 15 ...
 15 40 5 10 10 12 10 9 0 20 ...
 60 40 50 36 49 40 19 150 ];

W=600;
A=600;
```

where v is the profits matrix including the profit of each item, w and a are the weighting matrices for the two constraints and W and A are their respective constraint capacities. The derived change cycle events will be with the same notions.

B.2 Increased Optimal Cost

It is true that MKP is a maximization problem; however as explained in Chapter 3, we minimize the total profit of deselected items (Eq. 3.8). In this sense increased optimal cost means that the total profit of deselected items is increased, and thus the profit of selected items is decreased. The known best value achieved by IEigenAnt is 133615 for this change cycle event.

```
v=[ 1898 440 22507 270 4952 3100 4650 30800 615 4975 ...
 1160 4225 510 11880 479 440 490 330 110 560 ...
 24355 2885 11748 4550 750 3720 1950 10500];
w=[45 0 85 150 65 95 30 0 170 0 ...
 40 25 20 0 0 25 0 0 25 0 ...
 165 0 85 0 0 0 0 100 ];
a=[ 30 20 125 5 80 25 35 73 12 15 ...
 15 40 5 10 10 12 10 9 0 20 ...
 60 40 50 36 49 40 19 150 ];
W=600;
A=600;
```

B.3 Emergence of a New Optimal Solution

The best value achieved by IEigenAnt for is 147277 this change cycle event.

```
v=[ 2164 792 22507 270 4952 3100 4650 30800 615 4975 ...
 1160 4225 510 11880 479 440 490 330 110 767 ...
 24355 2885 11748 4550 750 5208 1950 10500];
w=[45 0 82 150 65 95 30 0 170 0 ...
 40 25 4 0 0 25 0 0 25 0 ...
 165 0 85 0 0 0 0 100 ];
a=[ 10 20 125 5 80 25 35 73 12 15 ...
 15 40 5 10 10 8 10 9 0 20 ...
 60 40 50 36 49 40 19 150 ];
W=738;
A=750;
```

B.4 Radical Change

The known optimal value achieved by IEigenAnt for this problem is 125821.

```
v=[ 1898 440 22507 270 14148 3100 4650 24948 615 4975 ...
 1160 4225 510 11880 479 440 490 330 110 560 ...
 24355 2885 11748 4550 750 3720 1950 10500];
w=[45 0 85 150 65 95 30 0 170 0 ...
 40 25 20 0 0 25 0 61 25 0 ...
 165 0 85 0 0 0 0 100 ];
a=[ 30 20 125 5 80 25 35 73 12 15 ...
```

```
 15 40 5 10 10 12 10 9 0 20 ...
 60 40 50 36 49 40 19 150 ];
W=600;
A=503;
```