T.C.

SELCUK UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

SOLUTION AND DEVELOPMENT OF THE TRAVELLING SALESMAN PROBLEM AND DATA ALLOCATION PROBLEM BY USING HEURISTIC ALGORITHMS

MOSTAFA MAHI

Ph.D. THESIS

Computer Engineering

October 2018
KONYA
All rights reserved

Approval of Thesis

SOLUTION AND DEVELOPMENT OF THE TRAVELLING SALESMAN PROBLEM AND DATA ALLOCATION PROBLEM BY USING HEURISTIC ALGORITHMS

Submitted by MOSTAFA MAHI in partial fulfilment of the requirements for the degree of **Doctor of Computer Engineering Department, Selcuk University by:**

Jury:	**Signature**
Prof. Dr. Ahmet ARSLAN	[illegible]
Assoc. Prof. Dr. Halife KODAZ	[illegible]
Assist. Prof. Dr. Ömer Kaan BAYKAN	[illegible]
Assist. Prof. Dr. Mehmet HACIBEYOĞLU	[illegible]
Assist. Prof. Dr. Sait Ali UYMAZ	[illegible]

I approve the results above.

Prof. Dr. Mustafa YILMAZ

Director of the Graduate School of Natural and Applied Science

Data of Approval: 15/10/2018

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

This thesis is the result of research and project undertaken by me, in cases where I have used the scientific and research achievements of others (including thesis, book and article, etc.). In accordance with the rules and procedures, I have included the name of the source used and other specifications in the relevant listing.

Mostafa MAHI

Signature: M.Mahi

Date: 15/10/2018

ÖZET

DOKTORA TEZİ

Gezgin Satıcı Problemi ve Veri Tahsis Probleminin Sezgisel Algoritmalar Kullanılarak Çözümü ve Geliştirilmesi

Mostafa MAHI

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Halife KODAZ

2018, 61 Sayfa

Bu tez çalışmasında, iki önemli optimizasyon problemi ele alınmıştır. İlk problem olarak Gezgin Satıcı Problemi (TSP) hibrit bir yöntemle çözülmeye çalışılmıştır. Önerilen yöntemde Karınca Kolonisi Optimizasyonu (KKO) algoritmasının parametreleri Parçacık Sürü Optimizasyonu (PSO) ile optimize edilmiştir. Daha sonra yerel minimumlardan kaçınmak için 3-Opt algoritması kullanılmıştır. İkinci problem olarak Veri Tahsis Problemi (VTP) literatürde daha önce uygulanmamış olan PSO algoritması ile çözülmeye çalışılmıştır. Ayrıca, VTP için açgözlü bir yöntem önerilmiştir.

TSP, standart optimizasyon problemlerinden biri olarak optimizasyon algoritmalarının verimliliğini ölçmek için kullanılmaktadır. KKO algoritması, ayrık optimizasyon problemlerini çözmek için kullanılmaktadır. Bu tez çalışmasında, TSP'nin çözümü için yeni bir hibrit yöntem önerilmektedir. PSO algoritması aracılığıyla KKO algoritmasının giriş parametreleri bulunmakta ve son olarak tur esnasında oluşan çapraz kenarları kaldırmak için 3-OPT algoritması kullanılmaktadır. KKO algoritmasında yer alan α and β parametrelerinin optimal değerleri PSO algoritması ile bulunmaktadır. KKO algoritması, kenarların kesişim noktasını çözmede başarısız olmasından dolayi tur sırasında seçilen şehirlerin seçimini iyileştirmek için 3-Opt algoritması kullanılmıştır. Önerilen yöntem ile literatürde yer alan algoritmaların sonuçlarını karşılaştırmak için 10 adet standart veri seti üzerinde test yapılmıştır.

VTP, optimizasyon algoritmalarının verimliliğini ölçmek için kullanılan bir başka optimizasyon problemidir. Bu tahsis işleminde, yürütme süresinin ve sorguların işlem maliyetinin minimize edilmesi hedeflenmektedir. Bu problemi çözmek için literatürde daha önce kullanılmamış olan PSO yöntemi kullanılmıştır. Ayrıca üçüncü bir yöntem olarak aç gözlü bir yöntem önerilmiştir. PSO tabanlı ve açgözlü yöntemlerin performansları 20 farklı test kullanılarak kıyaslanmıştır. Elde edilen sonuçlar sunulan yöntemlerin, literatürdeki yöntemlere göre uygulama süresi ve toplam maliyet bakımından daha iyi olduğunu göstermiştir.

Anahtar Kelimeler: Aç Gözlü Algoritma, Gezgin Satıcı Problemi, Parçacık Sürüsü Optimizasyonu, Karınca Kolonisi Optimizasyonu, Veri Tahsis Problemi, 3-Opt Algoritması.

ABSTRACT

Ph.D. THESIS

SOLUTION AND DEVELOPMENT OF THE TRAVELLING SALESMAN PROBLEM AND DATA ALLOCATION PROBLEM BY USING HEURISTIC ALGORITHMS

Mostafa MAHI

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELCUK UNIVERSITY
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN COMPUTER ENGINEERING

Advisor: Assoc. Prof. Dr. Halife KODAZ

2018, 61 Pages

In this thesis, two important optimization problems are discussed. The first problem was solved by a hybrid method. In the proposed method, the parameters of the Ant Colony Optimization (ACO) algorithm was optimized with Particle Swarm Optimization (PSO). The 3-Opt algorithm used to avoid local minimums. The second problem was the Data Allocation Problem(DAP), solved by the PSO algorithm, which was not used previously in the literature. In addition, a greedy method for DAP was proposed.

TSP is used as a standard optimization problem to measure the efficiency of optimization algorithms. The ACO algorithm is used to solve discrete optimization problems. In this thesis, a new hybrid method is proposed for the solution of TSP. The ACO algorithm has input parameters via the PSO algorithm. Finally, the 3-OPT algorithm is used to remove the crossed edges in the round. With the PSO algorithm, the optimal values of the α and β parameters in the ACO algorithm are decided. The 3-Opt algorithm is used to improve the selection of selected cities during the tour when the ACO algorithm fails to resolve this intersection point of the edges. 10 standard data sets were tested to compare the proposed method and the results of the algorithms in the literature.

The DAP is another optimization problem used to measure the efficiency of optimization algorithms. Main purpose of this allocation process is minimizing the execution time and the transaction costs of the queries. In order to solve this problem, the PSO method has been applied which is not used in the literature. A third method has also been proposed as a greedy method. The performances of the PSO-based and greedy methods have been examined in 20 different test problems. The results showed that the methods presented were better than the results of the methods in the literature in terms of execution time and total cost.

Keywords: Ant Colony Optimization, Data Allocation Problem, Greedy Algorithm, Particle Swarm Optimization, Travelling Salesmen Problem, 3-Opt Algorithm.

ACKNOWLEDGEMENTS

The sweet gift of my goddess who made my successor to finish this thesis. Thank you, Assoc. Prof. Dr. Halife KODAZ, because it was very difficult to get this thesis without guidance.

I am grateful to acknowledge and thank all of those who assisted me in my doctorate program at Selcuk University. Special thanks are given to my other doctorate committee members Prof. Dr. Ahmet ARSLAN, Assist. Prof. Dr. Ömer Kaan BAYKAN and Assoc. Prof. Dr. Mustafa Servet KIRAN. Special thanks to my best friend Fatih TITREK.

Acknowledgment is extended to my mother, father, my wife and my sisters for support and throughout this work.

Mostafa MAHI
KONYA-2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

SYMBOLS	EXPLANATION
n	The number of sites.
m	The number of fragments.
i	The index of sites.
j	The index of fragments.
S_i	The i^{th} site.
$SiteCap_i$	The storage capacity of site S_i.
$UC_{n\times n}$	The matrix denoting the cost of unit data transmission between each two sites.
uc_{i1i2}	The cost of sending a unit data item from site S_{i1} to the site S_{i2}.
f_j	The j^{th}fragment.
$fragSize_j$	The size of fragment.
L	The number of considered transactions.
t_k	The k^{th}transaction.
$FREQ_{n\times l}$	The matrix denoting the execution frequency of each transaction in each site.
$freq_{ik}$	The execution frequency of transaction t_k in sites$_i$.
$TRFR_{l\times m}$	The matrix denoting the direct transaction − fragment dependency.
$trfr_{kj}$	The volume of data items of fragment f_j that must be sent from site containing f_j to the site executing transactiont_k, for each execution
$Q_{\ l\times m\times m}$	The matrix denoting the indirect transaction − fragment dependency.
q_{kj1j2}	The volume of data items that must be sent from site containing fragment $f\ j1$ to the site storing $f_{j2,}$for each execution of transa
Ψ	The m element vector which denotes an allocation scheme.
Ψ_j	The site to which fragment t_k is assigned in the allocation schemeΨ.
$COST(\Psi)$	The cost of data transmission in an allocation schemeΨ.
$COST1(\Psi)$	The cost of data transmission in an allocation schemeΨ resulting from direct tr − fragment dependencies.
$COST2(\Psi)$	The cost of data transmission in an allocation scheme Ψ resulting from indirect fragment dependencies.
$STFR_{n\times m}$	The matrix denoting the site − fragment dependency.
$stfr_{ij}$	The volume of data items from fragment f_j time (according to the site − fragment dependency) which are accessed by site $s\ _i$ in unit.
$PARTIALCOST1_{nx}$	The matrix denoting the $COST1(\Psi)$ incurred by allocating each fragment to eac
$partialcost1_{ij}$	The cost incurred by $f\ j$ allocated to site s_i as a result of direct transaction fragment dependency.
$QFR_{l\times m\times m}$	The matrix denoting the indirect transaction fragment dependency taking the execution frequencies of the transactions into account.
qfr_{kj1j2}	The volume of data needed to be sent from site storing fragment f_{j1} to the site having fragment f_{j2} in unit time taking into account the transaction frequency of t_k.
$FRDEP_{m\times m}$	The matrix denoting the inter fragment dependency.
$frdep_{j1j2}$	The volume of data items needed to be sent from site having fragment f_{j1} to th site having fragment f_{j2} dependency in unit time due to the indirect transaction fragment.
$ParticleNumber$	The number of Particle.
V	The velocity of Particle.
X	The position of Particle.
$TotalCap_i$	The current capacity of site S_i.
$IterationNumber$	The number of iteration.
W	Inertia weight.

S	The count number of plus signs.
X_s	The mean of the binomial distribution.
σ_s	The standard deviation of the binomial distribution.
Z	Test statistic.
H_0	There is no significant difference between the two algorithms.
H_1	There is a significant difference between the two algorithms.
C	*Approximation of the average fragment size*
UCN	*Unit transmission cost between two neighbor sites*
L	*Number of transactions*
RPT	*Probability of a transaction being requested at a site*
APF	*Probability of a fragment being accessed by a transaction*
$APFS$	*Probability of a transaction necessitates data transaction between two si (other than the originating site)*
P	*Number of particle*
c_1, c_2	*Learning factors*
K	*Number of iteration*
W	*Inertia weight*
V_{max}	*Maximum velocity*
$rand_1, rand_2$	*Generate random number*
Avg	*Average*
TSP	*Travel Salesman Problem*
DAP	*Data Allocation Problem*
PSO	*Particle Swarm Optimization*
ACO	*Ant Colony Optimization*
GA	*Genetic Algorithm*
ANN	*Artificial Neural Network*
ABC	*Artificial Bee Colony*
SA	*Simulated Annealing*
DMD-ATA	*Dynamic Multi – Dimensional Anamorphic Travelling Ants*
TODMA	*Through data migration algorithm on task level*
RFID	*Radio Frequency Identification*
BBO	*Biogeography Optimization*
NN	*Nearest Neighbour*
DNN	*Dual Nearest Neighbour*
RABNET	*Real – valued Antibody Network*
POPMUSIC	*Partial Optimization Metaheuristic Under Special Intensification Conditic*
RVND	*Randomized Variable Neighbourhood Descent*
$PACO - 3Opt$	*Parallel Cooperative hybrid algorithm*
RTS	*Robust Tabu Search*

1. INTRODUCTION

The majority of real-world problems are solvable in various scientific fields using heuristic algorithms. Optimization can be defined as the selection of the best element according to some criteria from a set of available alternative elements. In order to solve scientific problems such as mathematical sciences, medicine, various types of engineering disciplines are used for optimization. Optimization problems are generally divided into two groups, continuous and discrete. Optimization problems in computer and mathematics science offer the best solution among scientific solutions. The optimization systems are divided into continuous and discrete categories. Optimization problems are a mix of optimization problems with discrete variables. In a hybrid optimization algorithm, we are looking for a set of objects such as integers, permutations or graphs whose number of members is finite (or unlimitedly counting). The example of Travelling Salesman Problem (TSP) (Dorigo and Gambardella, 1997; Dorigo and Stützle, 2009; Taillard and Helsgaun, 2019) and Data Allocation Problem (DAP) (Adl and Rankoohi, 2009) is a discrete optimization type. Given the fact that the TSP is the type of graph and in the DAP, there is a set of objects such as fragmentations and sites are characterized by discrete variables as a discrete optimization problem. In the meantime, due to this fact that in the TSP and DAP, the variables of the problem at a given point have discrete variables. So we can classify the TSP and DAP in a discrete category. Considering the variables in the TSP and DAP as discrete ones, we can solve the heuristic algorithms or a combination of them, which is based on combining the algorithms of Particle Swarm Optimization (PSO), Ant Colony Optimization algorithm (ACO) and 3-Opt, as well as the Greedy Algorithm. Traditional optimization techniques such as linear programming (LP), nonlinear programming (NLP) and dynamic programming (DP) have played a major role in solving these problems. Classic optimization methods are categorized into three linear, quadratic and nonlinear sections. Liner program is divided into five sections display, simplex, large scale, algorithm (active-set, interior-point and simplex) and diagnostics. While optimization methods and heuristic algorithms are used to solve optimization problems, practical methods cannot be used in practice, because it requires a long time to find the desired result and to solve multidimensional and large-scale problems in the real world. On the other hand, heuristic algorithms are often used in practice, because they find the desired (optimal) result or result close to the desired (optimal) result at a reasonable time solving multi-dimensional and large-scale problems in the real world.

To solve optimization problems, metaheuristic and new algorithms are recommended. The heuristic algorithms are classified into two categories of evolutionary computing and swarm intelligence. Swarm intelligence methods include PSO (Eberhart and Kennedy, 1995), ACO (Dorigo and Stützle, 2009), Artificial Bee Colony algorithm (ABC) (Karaboga and Basturk, 2007), Grey Wolf Optimization (GWO) (Mirjalili et al., 2014), Genetic Algorithms (GA) (Goldberg, 1989), Memetic Algorithms (MA) (Neri et al., 2012) and Gene Expression algorithms (GE) (Ferreira, 2006).

1.1 Thesis Aim And Literature Contribution

In this thesis, three methods are proposed for solving TSP and DAP. In the first method, a hybrid method for PSO, ACO and 3-Opt algorithms for TSP are suggested. The proposed method was tested on Eil51, Berlin52, Rat99, Eil76, St70, Kroa105, Kroa200, CH150 and Eil101 datasets in website TSPLIB (Pham et al., 2018). The obtained results are compared with the results of the studies done in the literature on these data sets. Often, the results of the proposed method are better than those of the literature. Considering the comparison of the proposed method with the previous and similar methods, we have obtained better results, which indicates better performance of the proposed method. Another problem that is used in the thesis study is DAP. For the DAP, the PSO algorithm, which was not applied previously, is used. We compare the results of our proposed work with the results of previous work in literature. Most of our work results are better than the previous literature. As a third method, a greedy method has been proposed for DAP. We compare the results of our proposed work with the results of previous alternatives in literature.

1.2 Related Works

In this section, previous work is being done to solve TSP and DAP problems. For decision making, the shortest tour between all cities, which is the type of theory of computational complexity and NP-complete problems, is considered. The runtime of the worst case for the TSP algorithm increases by increasing the number of graph cities as an exponential. For this reason, for solving the TSP, which is the complexity of higher computing time, there are many heuristic algorithms for solving it. Grefenstette et al. have used the GA algorithm and various meta-cognitive methods to solve the TSP problem (Grefenstette et al., 1985). Shi et al. used the PSO algorithm to solve the TSP to reduce its execution time (Shi et al., 2007). Geng et al. provided a local-based search algorithm with greedy search techniques based on Simulated Annealing (SA) to solve TSP (Geng et al., 2011).

To obtain more precise solutions, the SA algorithm is based on mutations with different probabilities during the search. Jolai and Ghanbari have been using the Artificial Neural Network (ANN) to solve the TSP (Jolai and Ghanbari, 2010). To make results more accurate and to get shorter tours of Hopfield Neural Networks and data transfer techniques. Pedro et al. Also used Tabu Search algorithms to solve TSP (Pedro et al., 2013). Dorigo et al. proposed an Ant System for solving TSP (Dorigo and Gambardella, 1997). Dorigo and Gambardella determined an ACO to solve the TSP (Dorigo and Gambardella, 1997). They prove that the ACO algorithm provides the best solutions for symmetric and asymmetric TSP. Mavrovouniotis and Yang provided an ecosystem for dynamic environments (Mavrovouniotis and Yang, 2013). Their frameworks include random immigrants, various immigrant schemes, migrant-based memory and elite-based immigrants. Karaboga and Gorkemli provided a new algorithm for Artificial Bee Colony (ABC) to name combinatory ABC for TSP (Karaboga and Gorkemli, 2011). They proved that the ABC algorithm is used for solving hybrid optimization problems.

To solve TSP, heuristic hybrid procedures relying on SA, ABC, ANN, PSO, ACO and others were exerted. Bountoux and Feillet applied combination algorithm to untangle the TSP (Bontoux and Feillet, 2008). Their algorithm includes the ACO algorithm crossbreed with local search methods. They are named Dynamic Multi-Dimensional Anamorphic Travelling Ants (DMD-ATA). Tsai et al. provided a metaheuristic method named ACOMAC algorithm to untangle the TSP They introduce several concepts from ant colony from a parallel genetic algorithm so that local search space can be used by different islands to prevent the minimum local, until finding the global minimum answer to solve the TSP. They provided two new solutions k Nearest Neighbours (K-NNs) and the Dual Nearest Neighbour (DNN) to ACOMAC to solve the big problems of TSP. Pasti and Castro proposed method on a trained neural network using the immune system ideas to a meta-heuristics solution of TSP (Pasti and De Castro, 2006). In an organized structure, they apply a learning algorithm on a network cell to solve the TSP problem. Their network has according to a Real-valued Antibody Network (RABNET). Masutti and Castro with a structured neural network used the modified RABNET-TSP problem to solve the TSP (Masutti and de Castro, 2009a). Combined with the ACO algorithm and beam search called Beam-ACO, it is used to solve TSP (López-Ibáñez and Blum, 2010). Cheng and Mao changed the ACO algorithm and called it Ant Colony System-TSP with Time Windows (ACS-TSPTW) to solve the TSP (Rodríguez Vásquez, 2016). To solve the Maxim and Minim problems in the optimization problems, Krohling and Coelho provided a PSO-based a co-evolutionary method (Krohling and dos Santos Coelho, 2006). By combining the methods of evolutionary learning algorithm and neural fuzzy network based on Link function, Lin et al. designed an evolutionary neural fuzzy network (Lin et al., 2009). Their evolutionary learning algorithm designed for prediction problems is combined from the sequential and PSO algorithm. Chen and Chen using the ideas of SA, ACO and PSO algorithms, presented an method for solving TSP (Chen and Chien, 2011a). A hybrid ant colony algorithm (HACO) is a combination of the ACO algorithm and the delete-cross method which is used for local search convergence and eliminates the slow-moving slowdown of the ACO algorithm (Junqiang and Aijia, 2012). Dong et al. in collaboration with the GA and the ACO, Which makes the ACO algorithm more effective for solving TSP and also called their method Cooperative Genetic Ant System (CGAS) (Dong et al., 2012a). Packer et al. Using the ACO algorithm and optimizing the Taguchi method parameters, provided a solution to the TSP (Peker et al., 2013b). Gunduz and Kiran proposed an intelligent movement-based algorithm for particles to solve the TSP (Gündüz et al., 2015).

Osaba et al. the discrete algorithm improved for symmetric and asymmetric in introduced and applied to TSP (Osaba et al., 2016). Zhong et al. the PSO algorithm for discrete learning, which uses the acceptance criteria of the SA algorithm, is presented for TSP. New flight equation, which can both capture both the best features of each particle and the features of the problems, is designed for the TSP (Zhong et al., 2018). Khan et. al. proposed a modified PSO algorithm to solve TSP with an inappropriate cost matrix

(Khan et al., 2018). Eric et. al. introduced a Partial Optimization Metaheuristic Under Special Intensification Conditions (POPMUSIC) for solving large data set of the TSP (Taillard and Helsgaun, 2018). Freitas and Penna introduced a Randomized Variable Neighbourhood Descent (RVND) heuristic condition (POPMUSIC) to solve the TSP of applying cities where flights take place(de Freitas and Penna, 2018). Gulcu et al. proposed the parallel cooperative hybrid method based on ACO and 3-Opt algorithm to solve the TSP (Gülcü et al., 2018).

Recently, one of the most attractive applications is distributed database called Data Allocation Problem (DAP). DAP is targeted to determine fragments placement in various sites to alleviate the transaction cost. DAP is a standard test problem which is used in optimization algorithms of performance analysis with special constraints (Adl and Rankoohi, 2009; Tosun et al., 2013b; Tosun, 2014a). Data allocations to sites are crucial. In reality, search engines or mail servers used data are big and disorganized. Fragments locations which request data can be changed. In such situations, data organization become more important. For instance, some items such as parallel query executions, network and servers load balancing are needed to be managed. DAP is NP-hard problem without considering of mentioned problems. DAP can be solved by two types of dynamic and static algorithms. Static algorithms based on the allocation of data are implemented on the static transaction execution model in the target environment. These templates are converted into dynamic algorithms (Gu et al., 2006; Adl and Rankoohi, 2009; Mashwani and Salhi, 2012). This part of the thesis refers to some studies which are reviewed DAP solution. DAP is NP-hard problem and several algorithms such as GA (Tosun et al., 2013a; Barbalios and Tzionas, 2014), ACO (Adl and Rankoohi, 2009; Tosun, 2014a) and metaheuristic methods are proposed to solve it.

The rest of the thesis surveyed other studies about DAP such as:

DAP problems can be solved by various solutions. Among the heuristic algorithms, the PSO algorithm is used for solving problems considering convergence rate, precision solution, robustness and easy compatibility. PSO algorithm doesn’t contain any overlapping and mutation calculation. To replace fragments on sites, the PSO-DAP method determines the optimal vector for particle velocity at each step. and sends the optimal result to the next iteration, until the iteration interval is reached and the final result will be optimized. In such situations researching speed is so high. This thesis is focused on solving DAP based on usage and conformity of PSO. PSO has lower control parameters, speed convergence and low power consumption characteristics. The strength of the problem solution space is not optimized to solve the DAP, but because of the features mentioned, it can be used to solve optimization problems with different features (Bai, 2010). This thesis is targeted to solve DAP based on PSO (Kiran and Gunduz, 2013; Mahi et al., 2015) and its performance is compared with genetic algorithm (Tosun, 2014a), Tabu Search (Tosun, 2014a), Ant Colony (Tosun, 2014a) and Simulated Annealing (Tosun et al., 2013b) on solving 20 problems with various dimensions. Execution time is a crucial factor for total cost. Presented algorithm has suitable and comparable results in considered time and it can be inferred that Greedy DAP execution time in comparison with other algorithms is the best. It is worth to note that the new dataset is prepared for the comparison of future studies. A met heuristic algorithm based on separating a graph database among nodes through defining all information on the same or adjacent nodes is proposed by Anita Brigit Mathew. A met heuristic algorithm including Best Fit Decreasing with ACO refers to data allocation in distributed architectures of NoSQL database graph (Sanchez et al., 2018). An effective data allocation method that contemplates static and dynamic specifications of data centers to make more effectual datacenter resizing is proposed by Wuhui Chen et al. Additionally, in order to alleviate the cost of communication in datacenter resizing, a generic model is proposed by them. their methods is a heuristic algorithm that contemplates traffic current in the network of data centers through first transmitting of DAP into a chunk distribution tree (CDT) which relate to the feasible solution in polynomial time and decreasing the CDT construction to a graph separation problem (Guo et al., 2017). An improved heuristic method based on division and allocation has been proposed by Amer and Abdalla all of the methods that are mentioned earlier are combined into single, efficient method that their effectual solution for DDBS productivity promotion is apparently completed. The outstanding point is that internal and external evaluations are widely demonstrated (Amer and Abdalla, 2012). Niamir et al. in 2018 has been proposed a method based on the complete taxonomy of the accessible division and allocation in the distributed database schema. Also, additional studies on instances of these methods to distinguish their limitations and achievements have been done (Niamir et al., 2018). Sivakumar and Basheer, 2017) evaluates several methods to distinguish an agent that was leaked to any part of the owner’s data and detect duplicate data in the cloud storage service (Sivakumar and Basheer, 2017). Data division in the distributed database system has been surveyed by Sanhani et al. their survey relates to the distributed database environment, fragmentations and distributed database design. Their compartment was among horizontal, vertical and mixed fragmentation. The correct rules and orders of fragmentation and the best way of data fragmentation in distributed data environments are discussed in (Al-Sanhani et al., 2017). An improved data allocation through data migration algorithm on task level (TODMA) has been presented by Jiayi Du et al. data migration and dynamic programming were coherent and were combined to allocate data in TODMA (Du et al., 2017; Mayne and Satav, 2017). Cost, performance and accessibility of large data application's one cloud are the most outstanding points that are analyzed by Mayne et al and consequently, three models have been built. These models relate to BRA algorithm to achieve all of the requirements of improved solution meeting. As a result, complete methods to allocating resources of large data application on the cloud has been implemented (Mayne and Satav, 2017). A SA to solve the DAP is done by Sen et al. Their thesis targeted to analyze the SA with benchmarks that are achieved through CPLEX benchmarks (Sen et al., 2016). Radio Frequency Identification (RFID) tag oriented DAP as a nonlinear knapsack problem has been modeled by Wang et al (Wang et al., 2015). Their work has been focused on artificial immune network (DA aiNet) utilization to address it. The effect of memory capacity and correlation matrix have been done by numerical assessments. Additionally, their work has been compared with its alternatives. Singh et al have proposed an algorithm for replicated fragment allocation in distributed database design for the static environment based on Biogeography Optimization (BBO). Their algorithm refers to replicated fragments method to alleviate the total cost of data transmission and storage cost of fragments. To evaluate the effectiveness of this method, its results have been compared with GA (Singh et al., 2014). Tosun presented recombination operator based on Order 1 crossover algorithm that is executed the quicksort partitioning algorithm to create several chromosomes of partitions. Other chromosomes with offspring are produced with the least cost of partition. Tosun et al. has been used GA, fast ACO and Robust Tabu Search (RTS) to solve DAP (Tosun et al., 2013b).

Time series modeling has been used by Li and Wong to predict short-term load (Li and Wong, 2013). This method is targeted to the number of node adjustment and fragment reallocation to clear node overloading and fragment reallocation of fragment mitigations. Tosun et al in 2013 have presented a set of SA, GA and fast ACO to solve DAP (Tosun et al., 2013a).A novel data reallocation model for replicated and non-replicated limited Distributed Database Systems (DDBSs) through data access pattern changing is proposed by Abdalla. In this method, distribution of fragments over network sites according to a proper predicted set of query frequency has been done over sites which are considered site restrictions in the reallocation phase. The mentioned method is so effective in data fragments reallocation over sites based on communication and update cost values respectively. Reallocation phase is based on selecting maximal update cost value for each fragment and consequently execute reallocation. Obtained results have shown that mentioned algorithm in solving fragments reallocation problem in a dynamic distributed relational database environment (Abdalla, 2012). Data reallocation model for DDBS based on data access over sites has been proposed by Amer and Abdalla. This method considered fragments scattering over network sites based on forecasted query frequency values over sites. Data fragments reallocation considering communication costs between sites and update cost values for each fragment is crucial in their algorithm. The reallocation phase has been done considering maximum update cost values selected for each fragment which is related to reallocation determination. The presented method in solving dynamic fragments reallocation problem in terms of distributed relational database systems is so effective (Amer and Abdalla, 2012).

ACO-DAP (Adl and Rankoohi, 2009) model based on ACO and local search have been presented by Adl and Rankoohi in 2009. In this method overcoming RAPs was targeted. Genetic algorithms were considered in their method and simulation results demonstrate that its performance was good. Ulus and Uysal have been presented a new dynamic DDBS called threshold algorithm in 2003 (Ulus and Uysal, 2003). In this method, data reallocation has been done by changing the data access pattern. Simulation results demonstrate that the threshold algorithm during changing data access pattern dynamically performs well in DDS. The mentioned method has been done based on evolutionary algorithms to allocate data in distributed database systems. Alternative reviews demonstrate that new methods are needed to increase the time and cost efficiency of the methods presented for DAP. It is known that PSO has not been used to solve DAPs. This thesis is targeted to solve DAP through PSO utilization and adaptation. Execution times and fragment allocation quality are investigated experimentally by PSO. Simulation results demonstrate that PSO-DAP’s (Mahi et al., 2015; Mahi et al., 2018) execution time in comparison with other algorithms, has suitable performance in determined time.

1.3 Thesis Organization

In the first section, the thesis study was introduced. Information on the aim and literary contributions of the thesis study have been given. In Section 2, detailed information is given about the studies in the literature to solve TSP and DAP. Section 3 contains information on the materials and methods used in the thesis study. In this section, PSO, ACO, 3-OPT and Greedy algorithms are explained after information about TSP and DAP. Also, 3 proposed methods (PSO-ACO-3-Opt, PSO-DAP and Greedy DAP) in the thesis are explained in this section. Comparisons and experimental results are in Section 4. The results of each proposed method are given in this section and compared with the results of the studies in the literature. Finally, Section 5 summarizes the conclusions of this thesis.

2. MATERIAL AND METHODS

This section is aimed at providing information about PSO, ACO, 3-Opt algorithms, TSP, DAP, a proposed method based on PSO, ACO and 3-Opt algorithms for TSP, the new method based on PSO for solving DAP and a new method based on Greedy Algorithm for solving DAP. The hybrid method developed by using these algorithms is explained in-depth.

2.1 Travelling Salesman Problem (TSP)

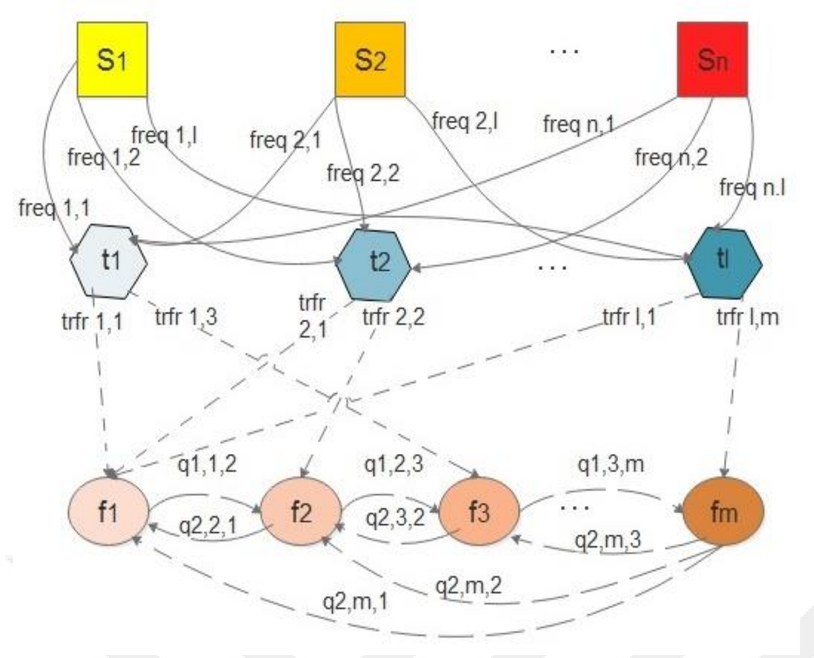
TSP is one of the standard combinatory distinct optimization problems which is a set of vertices (cities) and edges (the distance between these paths). The Salesmen randomly selects a city as the root, chooses routes that visit all the cities and eventually return to the chosen root, so that it will get the minimum tour during the journey. So the purpose of this problem is to find the shortest tour of a set of cities that cross each city just once, except the city of origin. This is an NP-hard problem type, it cannot exactly be solved using a polynomial algorithm. To solve the optimization problem of heuristic algorithms and their combination and novelty, clustering and parallelization discussions are used (Khan and Maiti, 2018).

2.2 Data Allocation Problem (DAP)

Finding the location of fragments at the best sites to alleviate the total cost of the transaction when a site sends a query to other sites (Adl and Rankoohi, 2009; Mamaghani et al., 2010; Tosun et al., 2013b; Mahi et al., 2018) is the purpose of DAP. Table 2.1 refers to notations. Figure 2.1 is shown the dependencies among sites, fragments and transactions. It is taken directly from (Adl and Rankoohi, 2009). For example, to get the query from S1 to S2 to obtain fragment j, transaction k is necessary. Transaction access to the website is done through Site Fragment Frequency (FREQ) matrix that is included frequency values between sites transactions. Transactions to fragmentations achievement are done through Transactions to fragmentations (TRFR) matrix. Evaluating transactions amount data relationship for fragments dependency is done through TRFR matrix which has some parameters. Two fragments of a transaction's data have existed in Q matrix. Each fragments size has selected randomly at interval $\frac{c}{10}$ whereas, in this interval $\frac{20+c}{10}$,c is an amount which is between $[10, 1000]$ (Adl and Rankoohi, 2009).

Table 2.1 Description of notations (Adl and Rankoohi, 2009).

Symbol	***Description***
n	The number of sites.
m	The number of fragments.
i	The index of sites.
j	The index of fragments.
S_i	The i^{th} site.
$SiteCap_i$	The storage capacity of site S_i.
$UC_{n\times n}$	The matrix denoting the cost of unit data transmission between each two sites.
uc_{i1i2}	The cost of sending a unit data item from site S_{i1} to the site S_{i2}.
f_j	The j^{th}fragment.
$fragSize_j$	The size of fragment.
L	The number of considered transactions.
t_k	The k^{th}transaction.
$FREQ_{n\times l}$	The matrix denoting the execution frequency of each transaction in each site.
$freq_{ik}$	The execution frequency of transaction t_k in sites_i.
$TRFR_{l\times m}$	The matrix denoting the direct transaction − fragment dependency.
$trfr_{kj}$	The volume of data items of fragment f_j that must be sent from site containing f_j to the site executing transactiont_k, for each execution of t_k.
$Q_{\,l\times m\times m}$	The matrix denoting the indirect transaction − fragment dependency.
q_{kj1j2}	The volume of data items that must be sent from site containing fragment $f\ j1$ to the site storing $f_{j2,}$for each execution of transactiont_k.
Ψ	The m element vector which denotes an allocation scheme.
Ψ_j	The site to which fragment t_k is assigned in the allocation schemeΨ.
$COST(\Psi)$	The cost of data transmission in an allocation schemeΨ.
$COST1(\Psi)$	The cost of data transmission in an allocation schemeΨ resulting from direct transaction fragment dependencies.
$COST2(\Psi)$	The cost of data transmission in an allocation scheme Ψ resulting from indirect transaction fragment dependencies.
$STFR_{n\times m}$	The matrix denoting the site − fragment dependency.
$stfr_{ij}$	The volume of data items from fragment f_j time (according to the site fragment dependency) which are accessed by site $s\ _i$ in unit.
$PARTIALCOST1_{nxm}$	The matrix denoting the $COST1(\Psi)$ incurred by allocating each fragment to each site.
$partialcost1_{ij}$	The cost incurred by $f\ j$ allocated to site s_i as a result of direct transaction fragment dependency.
$QFR_{l\times m\times m}$	The matrix denoting the indirect transaction fragment dependency taking the execution frequencies o the transactions into account.
qfr_{kj1j2}	The volume of data needed to be sent from site storing fragment f_{j1} to the site having fragment f_{j2} in unit time taking into account the transaction frequency of t_k.
$FRDEP_{m\times m}$	The matrix denoting the inter fragment dependency.
$frdep_{j1j2}$	The volume of data items needed to be sent from site having fragment f_{j1} to the site having fragment dependency in unit time due to the indirect transaction fragment.
$ParticleNumber$	The number of Particle.
V	The velocity of Particle.
X	The position of Particle.
$TotalCap_i$	The current capacity of site S_i.
$IterationNumber$	The number of iteration.
W	Inertia weight.
S	The count number of plus signs.
X_s	The mean of the binomial distribution.
σ_s	The standard deviation of the binomial distribution.
Z	Test statistic.
H_0	There is no significant difference between the two algorithms.
H_1	There is a significant difference between the two algorithms.



→ $freq_{i,k}$	$freq_{i,k}$ shows frequency value between transaction k and site i
⇢ $trfr_{k,j}$	$trfr_{k,j}$ shows the amount of required data to direct dependency of transaction k on fragment j.
⇢ $q_{k,j1,j2}$	$q_{k,j1,j2}$ shows amount of data from the site containing j_1 to the one containingj_2, for each execution of transaction k.

Figure 2.1 The dependences among sites, transactions and fragments (Adl and Rankoohi, 2009).

The size of each fragment withrf_i is calculated through Eq. 2.1. according to the formula below, provided that

$$(\sum_{i=1}^{n} p_i = m), rf_i = m - \sum_{q=1}^{i} p_q \qquad (2.1)$$

The capacity of each site is calculated by Eq. (2.2). (Adl and Rankoohi, 2009).

$$siteCap_i = p_i * max_{1\le j\le m}(fragSize_j) \qquad (2.2)$$

The site capacity should not be more than Eq. 2.3 during fragments replacement on the site.

$$\sum_{j=1}^{m} fragSize_j * x_{ij} \le siteCap_i \quad i = 1,2,\dots,n \qquad (2.3)$$

COST 1 is calculated from the allocation of fragments on sites according to Eq. 2.4. According to fragment size and site capacity, fragments allocated to sites and the vector is produced.

$$\mathrm{partialcost1}_{ij} = \sum_{q=1}^{n} \mathrm{uc}_{iq} * \mathrm{stfr}_{qj} \qquad (2.4)$$

Vector ψ (Eq. (2.5) is related to COST 1 calculation.

$$\mathrm{COST1}(\psi) = \sum_{j=1}^{m} \mathrm{partialcost1}_{\psi_j j} \qquad (2.5)$$

COST2 parameter is calculated as the query from site j1 to j2 has been taken. COST parameter is calculated through matrix q. matrix qfr_{kj1j2} is calculated through multiplying total of the k th column by the k th element of freq matrix to q_{kj1j2} matrix Eq. 2.6.

$$\mathrm{qfr}_{kj1j2} = \mathrm{q}_{kj1j2} * \sum_{r=1}^{n} \mathrm{freq}_{kr} \qquad (2.6)$$

FRDEP matrix is calculated based on an accumulation of transaction cost between fragments Eq. (2.7)

$$frdep_{j1j2} = \sum_{k=1}^{l} qfr_{kj1j2} \qquad (2.7)$$

COST 2 is achieved through FRDEP matrix and vector UC as Eq. (2.8) multiplying.

$$\mathrm{COST2}(\psi) = \sum_{j1=1}^{m} \sum_{j2=1}^{m} \mathrm{frdep}_{j1j2} * \mathrm{uc}_{\psi_{j1}\psi_{j2}} \qquad (2.8)$$

Finally, the sum of the COST 1 and COST 2 is related to COST according to the produced vector as Eq. 2.9 The vector which is created to allocate fragments on sites to get the query for the algorithm in this thesis is the best. Mentioned aim will be followed by our new method in the next section.

$$COST(\psi) = COST1(\psi) + COST2(\psi) \qquad (2.9)$$

2.3 Particle Swarm Optimization (PSO) Algorithm

PSO is designed by Kennedy and Eberhart, a population-based optimization algorithm, inspired by the collective behaviour of particle motion to find food, such as birds (Kennedy and Eberhart, 1995). Every single who is raised is called a particle and refers to a solution in the search space. Problem parameters specify the dimensions of the Particles for the desired problem. At first, the particles are distributed Completely random in the search space within the specified range. The motion dependency of each particle is in the fitness value of the function that moves at each iteration relative to the target function. The particle positions are updated as in Eq. (2.10) and Eq. (2.11) (Kennedy and Eberhart, 1995).

$$v_i^{t+1} = v_i^t + c_1 r_1^t (Pbest_i^t - X_i^t) + c_2 r_2^t (gbest - X_i^t) \tag{2.10}$$

$$X_i^{t+1} = X_i^t + v_i^{t+1} \tag{2.11}$$

Where X_i^t indicates that the point of the particle ith in the repetition, X_i^{t+1} indicates ith particle's position in $t+1$ iteration and v_i^{t+1} shows that the velocity vector ith particles. c1 and c2 determine the effect of the specific best particle solution ($PBest\ i$) and best system solution ($GBest$) on the velocity vector and r1 and r2 are random numbers at intervals [0-1]. In ameliorated versions of the PSO, the inertia weight w parameter, which determines the effect of the old velocity vector on the new velocity vector, was added to Eq. (2.10) (Kennedy and Eberhart, 1995). An algorithm operates to determine the number of iterations or error values obtained.

2.4 Ant Colony Optimization (ACO) Algorithm

The ACO algorithm is constructed by Dorigo and Gambardella as an inspiration from the actual behaviour of the ant colonies (Dorigo and Gambardella, 1997). By examining the behaviours of ants in real life, it was beholding that ants were able to find the shortest route between their nests and food sources. Impact Parameters to find the shortest path, substance and chemical are the pheromone that ants leave in the direction they use. Ants in a colony usually find a pathway that focuses on pheromone matter. The amount of pheromone is used on a frequent basis (Dorigo and Stützle, 2009). The algorithm that proposes a solution for TSP, which is a discrete test problem (hybrid), by Colorni et al. are used this feature of ants (Colorni et al., 1991). At TSPs, the Travelling salesman plans to make a tour of length minimum, given that they visit one city Every once. In this proposed algorithm, it has been avowed that ants leave pheromones on the paths between the cities they are using and pheromone escapes in a particular ration. Given the distance and volume of pheromones between cities, the choice of cities where ants go is greedily performed depending. This algorithm operates repetitively and the shortest path is considered to be the best solution. The choice of city j, which will be an ant in city i, will be iteration in t, based on Eq. (2.12).

$$P_{ij}{}^{k} = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}, if\ j\ is\ allowed\ city \\ 0, \quad otherwise \end{cases} \tag{2.12}$$

In Eq.(2.12), τ_{ij} shows amount the pheromones between cities i and j, τ_{ij} shows the indicates information ($\frac{1}{d_{ij}}$) corresponding to the distance between i and j cities and j displays the cities where the ant kth can go. An ant selects the city with the highest proportion of P_{ij} with a greedy choice. The parameters α and β are used to determine the importance of the pheromone value and the distance between the cities. *k*th ant will do a one total tour using an Eq. 2.12. The upper operations are said in repetition for all of the ants that are ready in the colony. The value of pheromones that is used by an ant in the path which was resolute matching by Eq. (2.13).

$$\Delta\tau_{ij}^{k}(t,t+1) \qquad = \begin{cases} \frac{Q}{L^{k}}, if\ (i,j) \in route\ performed\ by\ the\ k^{th}\ ant \\ 0\ , otherwise \end{cases} \tag{2.13}$$

Where L^k represents the tour distance, Q displays a constant number and k display kth of the ant in the colony. The total value of pheromones used by the ants that are provided in the colony and used the path between cities i and j are calculated using Eq. (2.14).

$$\Delta\tau_{ij}^{k}(t,t+1) = \sum_{k=1}^{n} \Delta\tau_{ij}^{k}(t,t+1) \tag{2.14}$$

The value of the pheromones found in the interurban routes in the iteration $(t+1)$ Eq. (2.15) also relevant to the effect of vaporization.

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \Delta\tau_{ij}^{k}(t,t+1) \tag{2.15}$$

In Eq. (2.15), ρ is the evaporation coefficient and takes the amount at intervals [0-1]. When the maximum number of iterations is reached, the shortest length of the tour is obtained as a solution to the problem.

2.5 3-Opt Algorithm

3-Opt is a simple local search algorithm for optimizing TSP. The 3-Opt algorithm is a specific instance of the k-opt algorithm (Dorigo and Stutzle, 2004; Gülcü et al., 2018; Taillard and Helsgaun, 2018). In the 3-Opt algorithm, the three edges of the tour are broken, reconnecting to the tour until the optimal solution is improved and reaches the result. Then this process is repeated for a different set of three connections (Dorigo and Stutzle, 2004). In this way, the graph is seen as edges, as shown in Figure 2.2 and the overlapping edges in the tour are plotted. By replacing short edges with long edges in a graph with a three-edged overlap, the solution TSP is optimized shown in Figure 2.2. To optimize the length of the tour, different algorithms such as GA, PSO, ACO and ABC are suggested. When these algorithms try to find the best tour, they stuck in the local minimum and this makes the best tour not to be found. To remove the minimum local positions, they are applied in the k-opt algorithm (Dorigo and Stutzle, 2004) and one of them is an algorithm of 3-Opt.

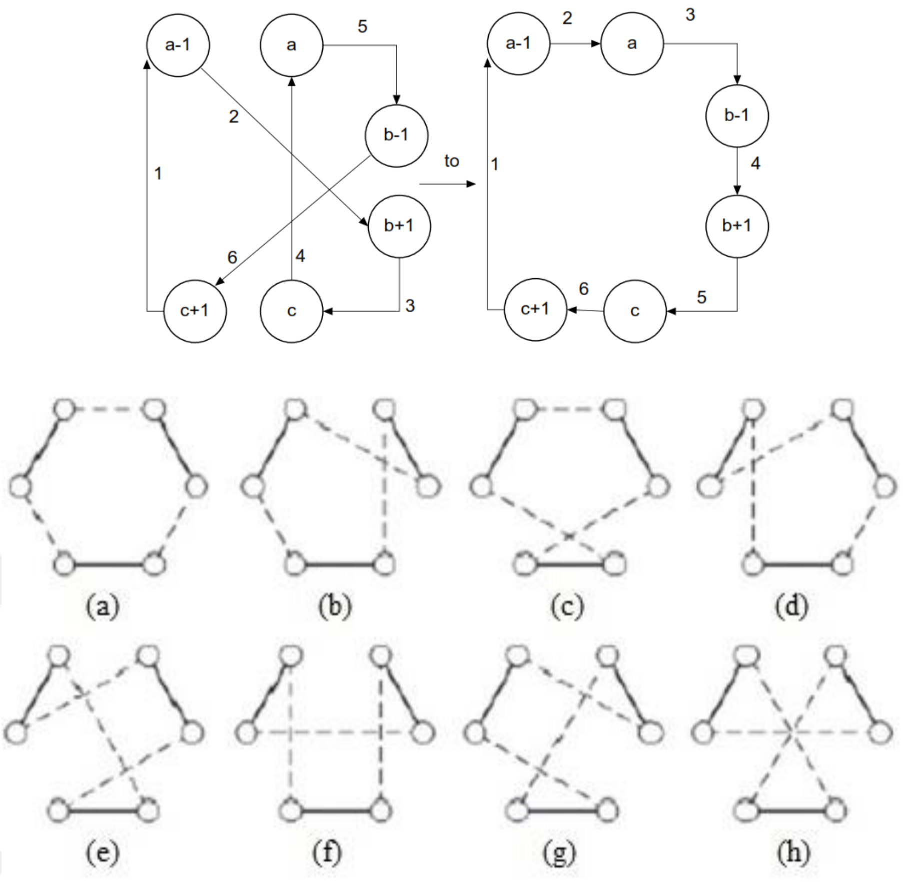


Figure 2.2 3-Opt Algorithm representation (Dorigo and Stutzle, 2004)

2.6 Greedy Algorithm

The base information of Greedy algorithm, optimal solution information for DAP and Greedy DAP is described in this section. A greedy algorithm is a simple and intuitive algorithm that is used in optimization problems. Its function is in a way that, the optimal choice is made at each step along with finding the overall optimal method to solve the entire problem. Huffman encoding that is used to compress data or Dijkstra's algorithm which is applied to find the shortest route through a graph are the examples of problems which can be solved successfully with Greedy algorithms. The greedy algorithm operates such as taking all of the data to a certain problem and consequently set a rule for elements to add solution at each step of the algorithm (Astrachan et al., 2002; Bouchaud, 2018; Gao et al., 2018).

3. PROPOSED METHOD

In this section, the proposed methods in the thesis are fully explained in three sections. In the first section, we outline the PSO-ACO-3-Opt hybrid method for TSP. In the second section, we explain how to solve the PSO method (PSO-DAP) on the DAP. Finally, Section 3 describes the greedy algorithm for the DAP.

3.1 Proposed Method Based On PSO, ACO and 3-Opt Algorithms For TSP

In general, consider the number of ants with the same number of cities to solve TSPs through ACO. In the TSP, the number of ants increases with the number of calculations and the complexity of the problem increases. Also, the parameters α and β are specified based on experience. In this thesis, a hybrid method is proposed based on the PSO, ACO and 3-Opt algorithms based on the impoundment of the performance of TSP solutions (Mahi et al., 2015). First, the ants are randomly divided into cities. In the following, pheromones are allocated to all the routes within the cities, the same amount as calculated by the formula in Eq. 3.1.

$$Amount\ of\ pheromone = 1/(number\ of ant\ * number\ of\ city) \tag{3.1}$$

All ants make their first trips only by taking the distance between the cities. The length of the tour is specified for all ants and the pheromone is updated in accordance with Eq. 2.13-2.15. The value of the parameters α and β in Eq. 2.12 are determined using the PSO. The target function of the PSO algorithm is the tour length. The ant $gbest$ represents the parameters α and β, which produces the minimum tour length for each ant in the PSO algorithm. Ant route and parameters that give the smallest length of the tour can be proposed as a good solution to the system. The Pheromone that changes according to Eq. 2.15 can be taken using the roads of the ants. As the number of iterations that have been chosen to the ACO algorithm has been achieved, mean that the iteration levels of the PSO-ACO is finished. In case of 3-Opt algorithm, the PSO-ACO is proposed such that if two branches cross each other, these branches should be broken such the path with the longest length is removed and find the shortest tour in the graph. We propose to use the 3-Opt algorithm to find a better solution. Figure 3.1 shows the Pseudo-code for the proposed method. Figure 3.2 shows the simplified flowchart.

```
- Initialization (number of ant, number of city, pheromone, iteration number)
- First tour length calculation using nearest city
- Calculation of new pheromone values
- Optimization of ACO parameters (α, β) with PSO
        While (ant ≤ number of ant)
                Initialize parameters for PSO,(particle=number of ant, $c_1$,$c_2$)
                While (t< iteration number)
                        While (particle ≤ number of ant)
                                $p_{ij}^{k}=\begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{r\notin \Gamma}^{n}\tau_{ij}^{\alpha}\eta_{ir}^{\beta}}, if\ j \notin \Gamma \\ 0, \quad otherwise \end{cases}$
                                Send parameters to ACO from PSO find the best tour
                                Calculation new α and β values
                                        $V_i(k + 1) = wV_i(k) + c_1r_1 (Pi - x_i(k))+ c_2r_2 (Pg - x_i(k))$ ,
                                        $X_i(k + 1) = X_i(k) + V_i(k + 1)$,
- Update pheromone values
- Execution algorithm 3-OPT
```

Figure 3.1 Pseudo-code of proposed hybrid method PSO,ACO,3-Opt(Mahi et al., 2015).

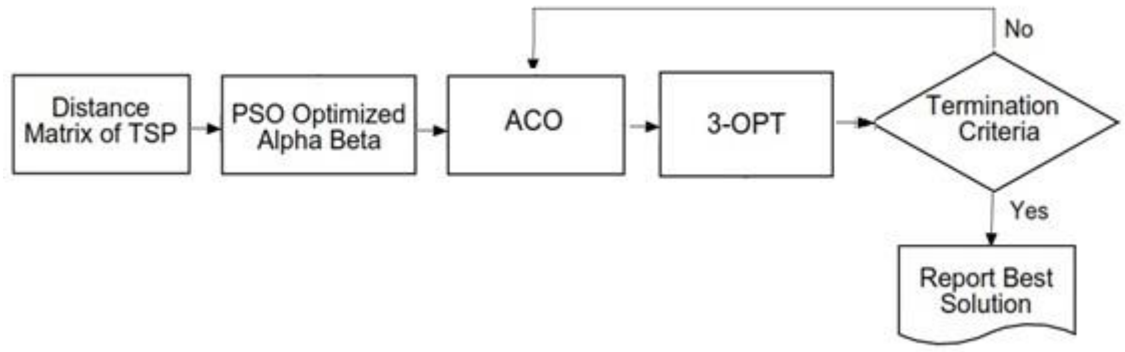


Figure 3.2 The flowchart of the proposed hybrid method PSO, ACO, 3-Opt.

3.2 Proposed Method Based On PSO For Solving DAP

In this thesis, we centred on making DAP decisions using and adapting PSO. Conforming to previous works, PSO has not exerted to solve DAPs. Because PSO has lower control parameters, features of isotropy speed and little consumption time and Power versus solution to optimization problems which is often used by doctors and researchers to solve optimization problems with various features (Bai, 2010; Deng et al., 2012).

In the proposed procedure for solving DAP, we benefit the PSO algorithm. We have identified on each site which fragments are located. Considering the fact that the whole cost should be low in this deal and the fragments are placed in the best sites in order to reduce the total transaction cost. In this thesis, the PSO algorithm is used to specify how to place fragments on sites. Fragment replacement in DAP is calculated using the cost formula. In PSO-DAP(Mahi et al., 2018), the vector particle size is 1 x m and its structure is as follows:

$$\begin{array}{ccccccccccccc} & & & 1 & 2 & 3 & 4 & 5 & . & . & . & m & & m \rightarrow \textit{fragment number} \\ P_k & = & [& 3 & 2 & 1 & 3 & 3 & . & . & . & n &] & n \rightarrow \textit{site number of the fragment} \end{array}$$

P_k is kth particle, $P_k[j] = i$ indicates that the jth fragment is located in the ith site ($i = 1,2,..,n. j = 1,2,..,m$). Instance, through $P_k[1] = 3$, fragment 1 is located in the site 3; through $P_k[2] = 2$, fragment 2 is located in the site 2 and etc. Particles indicate which fragment is located in which place.

To solve the DAP; dividing the components in best site along with minimizing the whole cost of the transaction is our goal. In this thesis, the procedure of allocating fragments to sites was evaluated using the PSO algorithm, taking into account the summation cost. For the PSO algorithm, the fitness is calculated using the cost function specified in Section 2.1 $Pbest$ and $gbest$ with the cost of the particle. In PSO-DAP, each site has the capacity to accept the size of the fragments. A counter is set For each site to review the site's capacity. The results of particle production and site capacity are reviewed. If there is extra capacity at each site, the starting location of the particles is re-generated. The description of the symbols is given in (Adl and Rankoohi, 2009).

If site capacity is not sufficient when replacing a fragment on the site, the position of the new particles is recalculated to the initial speed. The initial velocity of particles is randomly generated between [1, n]. If the location of the particle is greater than the distance of [1, n], these positions are randomly reproduced. As the particle location updating the continuous amounts which obtained the decimal numbers are rounded. At the end of each repetition, the obtained results of new particle position, their velocity and the pbest and gbest are updated in the PSO algorithm. PSO-DAP ends with a limited number of iterations. The PSO-DAP pseudo-code is shown in Figure 3.3 and its flowchart is shown in Figure 3.4.

```
- Initialization (number of particle, number of fragment and site, iteration number, length of the particle vector , site capacities, $k = 1$ )
Generate data set
Determination of the starting positions of the particles
While ($k <=$ iteration number)
        - Calculation of cost
                $COST(\psi) = COST1(\psi) + COST2(\psi)$
        - obtaining the value of $pbest$ and $gbest$
        - calculating particles velocity .
            $V_i(k+1) = w * V_i(k) + c_1 * rand_1 * (pbest_i - x_i(k)) + c_2 * rand_2 * (gbest - x_i(k))$
        -updating the location of particles
            $X_i(k+1) = X_i(k) + V_i(k+1)$
        If site capacities are overflow then generate new particle
            $k = k + 1$
End while
```

Figure 3.3 Pseudo-code provided for the PSO-DAP algorithm(Mahi et al., 2018).

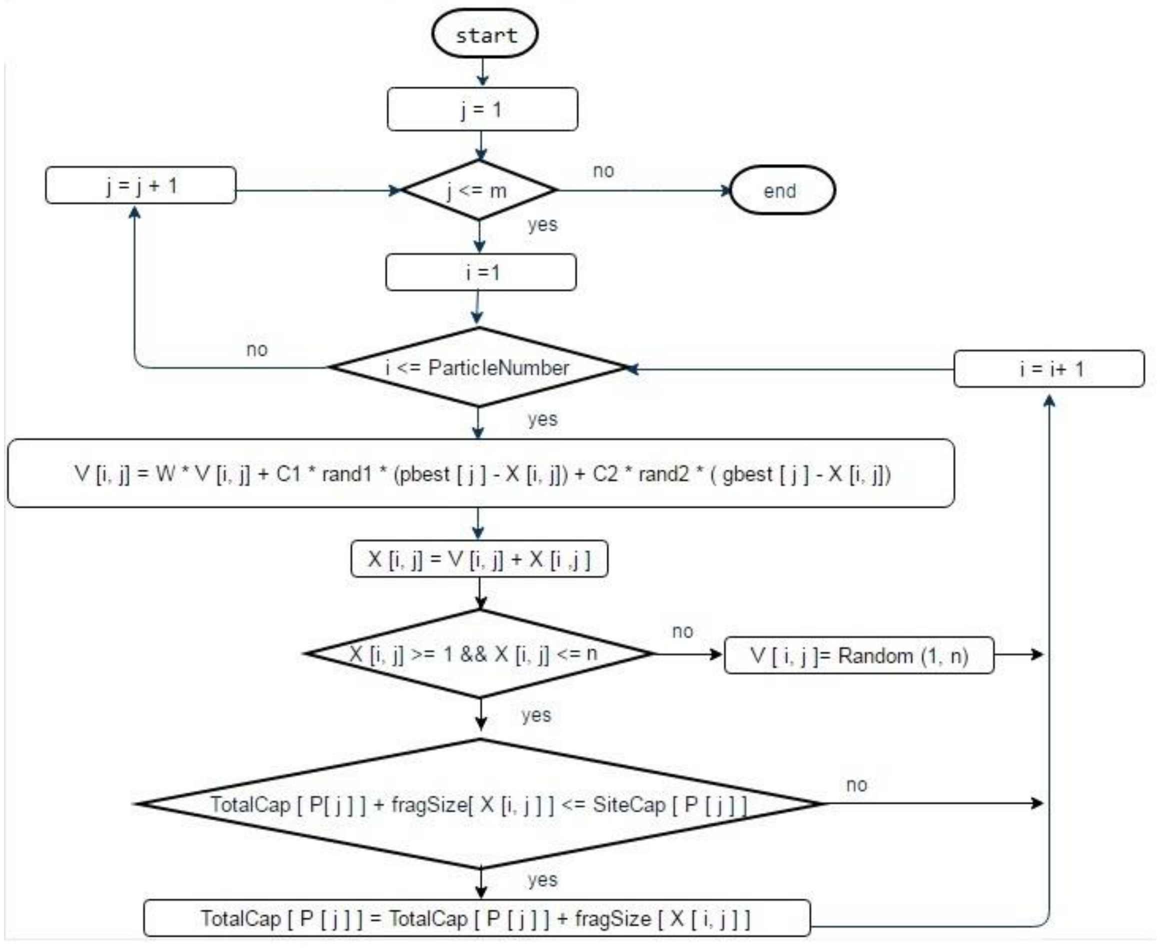

start
j = 1
j = j + 1
j <= m
no
end
yes
i =1
no
i <= ParticleNumber
i = i+ 1
yes
V [i, j] = W * V [i, j] + C1 * rand1 * (pbest [j] - X [i, j]) + C2 * rand2 * (gbest [j] - X [i, j])
X [i, j] = V [i, j] + X [i ,j]
X [i, j] >= 1 && X [i, j] <= n
no
V [i, j]= Random (1, n)
yes
TotalCap [P[j]] + fragSize[X [i, j]] <= SiteCap [P [j]]
no
yes
TotalCap [P [j]] = TotalCap [P [j]] + fragSize [X [i, j]]

Figure 3.4 The flowchart of the PSO-DAP (Mahi et al., 2018).

3.3 Proposed Method Based On Greedy Algorithm For Solving DAP

DAP solution based on Greedy algorithm utilization and adaptation is the aim of this study. The presented method can be introduced as Greedy DAP. It is a novel method to solve DAP. Due to the fewer control parameters of Greedy, some parameters such as speed convergence specifications, low consuming time and the Strength against the solution hops of the optimization problems are rarely applied to solve optimization problems with other characteristics by the vector p and scholars (Bai, 2010; Deng et al., 2012; Gao et al., 2018). p Greedy algorithm is used to solve DAP in this thesis. Fragments placements are determined in each site. To achieve this goal, the cost must be low so to reduce the cost of transactions, fragments must be located in the sites. Fragments placement in each site is done based on the Greedy algorithm in this thesis. Vector p is used to compute the cost of fragments replacement in the DAP. In the Greedy DAP, the size of vector p is 1 x m and its construction is as follow:

$$\begin{array}{lcccccccl} & 1 & 2 & \ldots & j & \ldots & m & \rightarrow fragment\ number \\ \text{vector } p: & 4 & 5 & \ldots & i & \ldots & n & \rightarrow site\ number\ of\ the\ fragment \end{array}$$

Vector p is an array, $p[\mathrm{j}] = \mathrm{i}$ indicates that the j^{th} fragment is located in the i^{th} site $(\mathrm{i} = 1,2,..,\mathrm{n}.\ \mathrm{j} = 1,2,..,\mathrm{m})$. For example, because of $p[1] = 4$, fragment 1 is located in the site 4; because of $p[2] = 5$, fragment 2 is located in the site 5 and etc. The p vector shows which fragment is placed on the site. Fragments placements on the best site to alleviate total transaction cost is the target of this thesis to solve DAP. Greedy algorithm considering the total cost is used to evaluate the steps to do the job of fragments allocation to sites. Fitness calculation is done by the cost function in the Greedy algorithm which is described in section 2.2. $COST(\psi)$ determination is done by the cost values of vector p. To generate the vector p as shown above, we calculate the Eq. (3.2).

$$Cost[i,j] = ((partialCost[i,k] * (frdep[k,j]) * (Uc[k,j])) \qquad (3.2)$$

Site capacity is one of the parameters in Greedy DAP; therefore, a counter for each site is determined to check the capacity of the site. As p vector generation, site capacities are checked. Any overcapacity in sites relates to the vector p preproduction. The cost matrix is given in Table 3.1. This matrix contains fragment m and site n, showing their indexes in j and i respectively. First, the minimum amount of cost matrix in terms of the site capacity and the fragment size is found and if we can replace the fragment on the site, we can update the site capacity and vector p and column j from cost matrix with the maximum numbers.

Table 3.1 Cost Matrix representation.

$$Cost\ Matrix = \begin{array}{c|cccccc|} & 1 & 2 & \cdots & j & \cdots & m \\ 1 & & & & & & \\ 2 & & & & \vdots & & \\ \vdots & & & & \vdots & & \\ I & \cdots & \cdots & \cdots & min & & \\ \vdots & & & & & & \\ n & & & & & & \end{array}_{n \times m}$$

Then we will find the next minimum value of the matrix and we will continue this process until all the matrices in the matrix are filled up to a maximum value. Replace the values of the column j in a maximal matrix $\mathrm{Cost}(i, j)$. An updated cost matrix is given in Table 3.2 Due to site capacity, some fragment values which are obtained from cost matrix replaced fragment j to site i and if the capacity of site i filled again, the minimum value from the cost matrix will be selected. This process will be continued until the fragment replacement.

Table 3.2 Updated cost matrix representation.

$$Cost\ = \begin{array}{c|cccccc|} & 1 & 2 & \cdots & j & \cdots & m \\ 1 & & & & Max & & \\ 2 & & & & Max & & \\ \vdots & & & & Max & & \\ i & \cdots & \cdots & \cdots & Max & & \\ \vdots & & & & Max & & \\ n & & & & Max & & \end{array}_{n \times m}$$

Greedy DAP pseudo code and consequently scripting chart of it is shown in Figure 3.5 and 3.6 respectively.

```
- Initialization ( number of fragment(m) and number of site(n), iteration number, length of the P, site
capacities )
Generate data set standard:
-Create Cost
      $Cost = frdep * partial\ cost$
-Determination of the allocation of fragment on sites of the Cost
                    find min cost from Cost$Matrix$
                If site capacities are overflow
                    Fragment does not replace on that site.
                Else allocate fragments to the sites And Update Site Capacity j
                    The values of column j are replaced by $Max$ value.
         - find again minimum from Cost
        - Specify the vector to allocate fragments to the sites
        -updating the location $Array$ of Cost
```

Figure 3.5 Pseudo of the Greedy DAP.

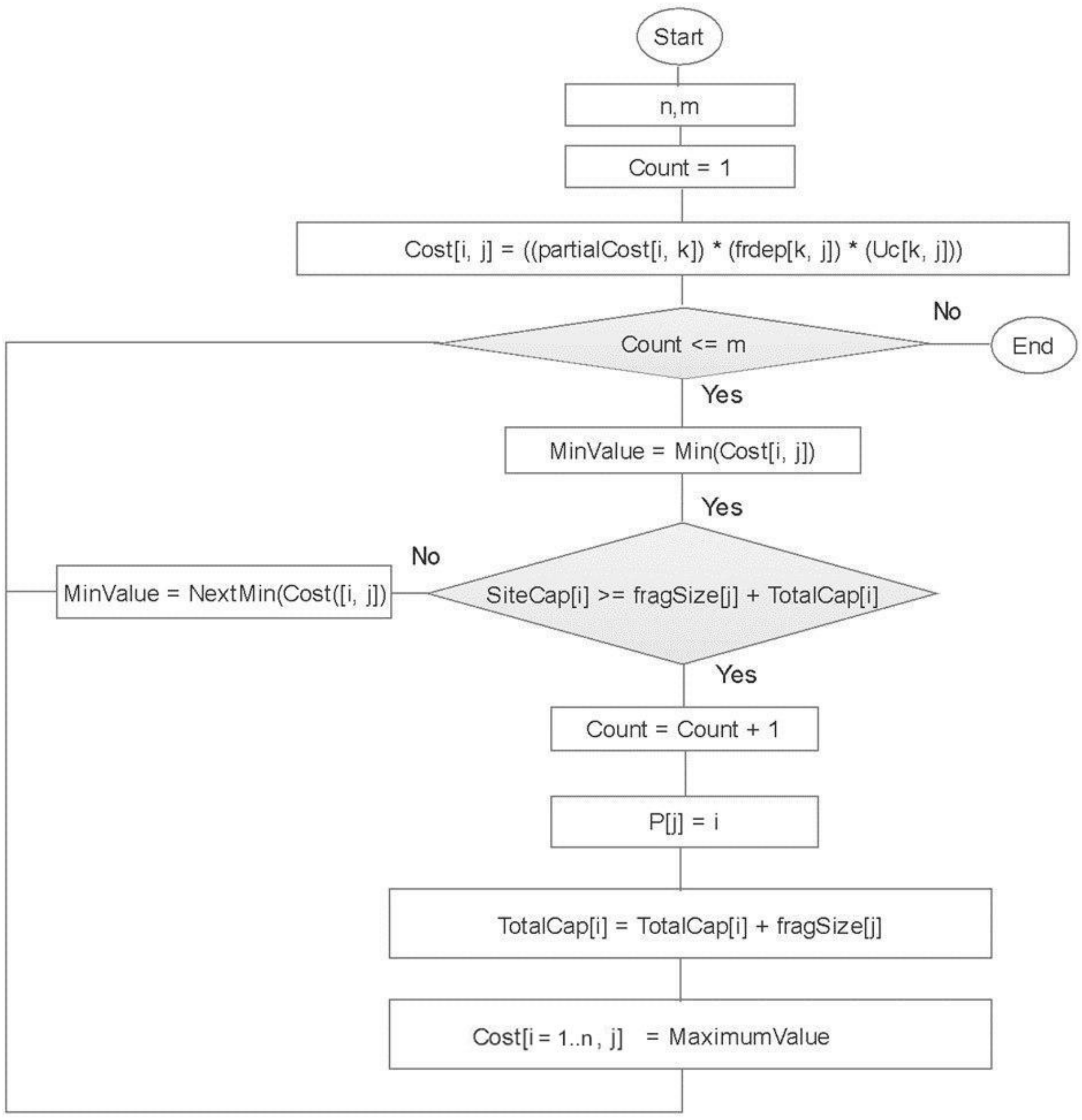

Start
n,m
Count = 1
Cost[i, j] = ((partialCost[i, k]) * (frdep[k, j]) * (Uc[k, j]))
Count <= m
No
End
Yes
MinValue = Min(Cost[i, j])
Yes
No
MinValue = NextMin(Cost([i, j])
SiteCap[i] >= fragSize[j] + TotalCap[i]
Yes
Count = Count + 1
P[j] = i
TotalCap[i] = TotalCap[i] + fragSize[j]
Cost[i = 1..n , j] = MaximumValue

Figure 3.6 The scripting chart of the Greedy DAP.

4. EXPERIMENTAL RESULTS

In this section, we show the results of the three proposed methods in the table and figure. Section 3.1 investigate proposed method based on PSO, ACO, 3-Opt algorithms for TSP results and compared it with other studies in the literature. Section 3.2 conclude method based on PSO algorithm for DAP results and compared it with other studies in the literature. Section 3.3 include method based on the Greedy algorithm for DAP results and compared it with other studies in the literature. For a different case of DAP, parts and results of a different number of sites and the tests performed are compared.

4.1 Experimental Results of Proposed Method Based on PSO, ACO and 3-Opt Algorithms for TSP

The implementation of proposed method for TSP was determined using standard deviations and average circuit length of the tour on ten different test problem taken from TSPLIB (Reinelt, 1991). For all experiments, the number of ants and particles is 10. The effect of the number of ants with different numbers is given in Table 3.1 for Eil51, Eil76, Rat99, Ch150 and Kroa200 tests in order. As shown in Table 3.1, the runtime increases and the performance becomes worse due to the number of ants. To better see the top results, they are shown in bold.

For *α, β*, the best value is selected in the corresponding range of $0 \leq \alpha , \beta \leq 2$. For each particle, two dimensions of *α, β* are considered for the PSO algorithm. parameters C1 and C2, in the PSO algorithm are set to value 2. Table 4.2 lists the best values for the input parameters of the ACO algorithm. Both ACO and PSO algorithms were executed in 1000 replays. The tests were repeated 20 times for each test. The percentage of pheromone vaporization in the ACO algorithm was selected value 0.1. The amount of evaporation ratio was resolved to be the best value after try and error.

The 3-opt algorithm solves the problem of intersecting edges in the graph, Figure 4.1, showed graph states before and after the algorithm execution. As shown in Figure 4.1, the PSO-ACO algorithm has not solved the intersection of edges in the graph, however, this problem has been solved with the application of the 3-OPT algorithm.

Table 4.1 Effect of ant number for efficiency. **Avg** is the average route length; SD is the standard deviation; Error(%) is relative error; Time(s) is run time in seconds (Mahi et al., 2015)**.**

Data Set		Ant Number			
		10	20	30	City Number
Eil51	Avg.	**426.45**	427.50	429.25	432.75
	SD	0.61	0.53	1.16	1.49
	Error(%)	0.11	0.35	0.76	1.58
	Time(s)	140.50	141.29	149.52	160.05
Berlin52	Avg.	**7543.20**	7580.38	7586.63	7598.63
	SD	2.37	22.03	22.93	30.98
	Error(%)	0.02	0.51	0.59	0.75
	Time(s)	170.46	173.32	174.10	180.90
Rat99	Avg.	**1227.40**	1240.37	1251.38	1254.13
	SD	1.98	6.41	6.00	7.06
	Error(%)	0.28	1.34	2.24	2.46
	Time(s)	284.09	294.77	305.98	326.58
Eil76	Avg.	**538.30**	540.38	546.75	549.73
	SD	0.47	1.77	1.98	2.87
	Error(%)	0.06	0.44	1.63	2.18
	Time(s)	220.68	239.95	279.17	283.65
St70	Avg.	**678.20**	680.875	681.56	683.50
	SD	1.47	1.73	3.13	1.77
	Error(%)	0.47	0.87	0.97	1.26
	Time(s)	256.89	260.15	273.20	291.61
Kroa100	Avg.	**21445.10**	21709.63	21816.44	21974.00
	SD	78.24	113.11	218.15	115.88
	Error(%)	0.77	2.01	2.51	3.25
	Time(s)	301.32	303.06	310.17	332.40
Lin105	Avg.	**14379.15**	14593.75	14664.88	14690.63
	SD	0.48	66.04	58.36	72.28
	Error(%)	0.00	1.49	1.99	2.17
	Time(s)	294.35	303.24	310.29	349.16
Kroa200	Avg.	**29646.05**	30357.63	30504.25	30662.75
	SD	114.71	51.00	148.50	202.56
	Error(%)	0.95	3.37	3.44	4..41
	Time(s)	302.15	308.43	380.42	1179.46
CH150	Avg.	**6563.95**	6727.25	6779.13	6800.13
	SD	27.58	32.69	7.55	23.93
	Error(%)	0.55	3.05	3.85	4.17
	Time(s)	286.90	300.93	329.83	346.37
Eil101	Avg.	**632.70**	646.00	647.25	647.75
	SD	2.12	1.77	2.49	1.83
	Error(%)	0.59	2.70	2.90	2.98
	Time(s)	302.15	302.32	302.70	330.40

Table 4.2 The best input parameters for the ACO algorithm

Problem	α	*β*
Eil51	1.11	1.44
Berlin52	0.95	1.05
Rat99	0.99	1.07
Eil76	0.88	1.50
St70	0.94	1.05
Kroa100	1.01	1.10
Lin105	1.20	0.65
Kroa200	0.75	1.15
Ch150	0.75	1.20
Eil101	1.20	0.75

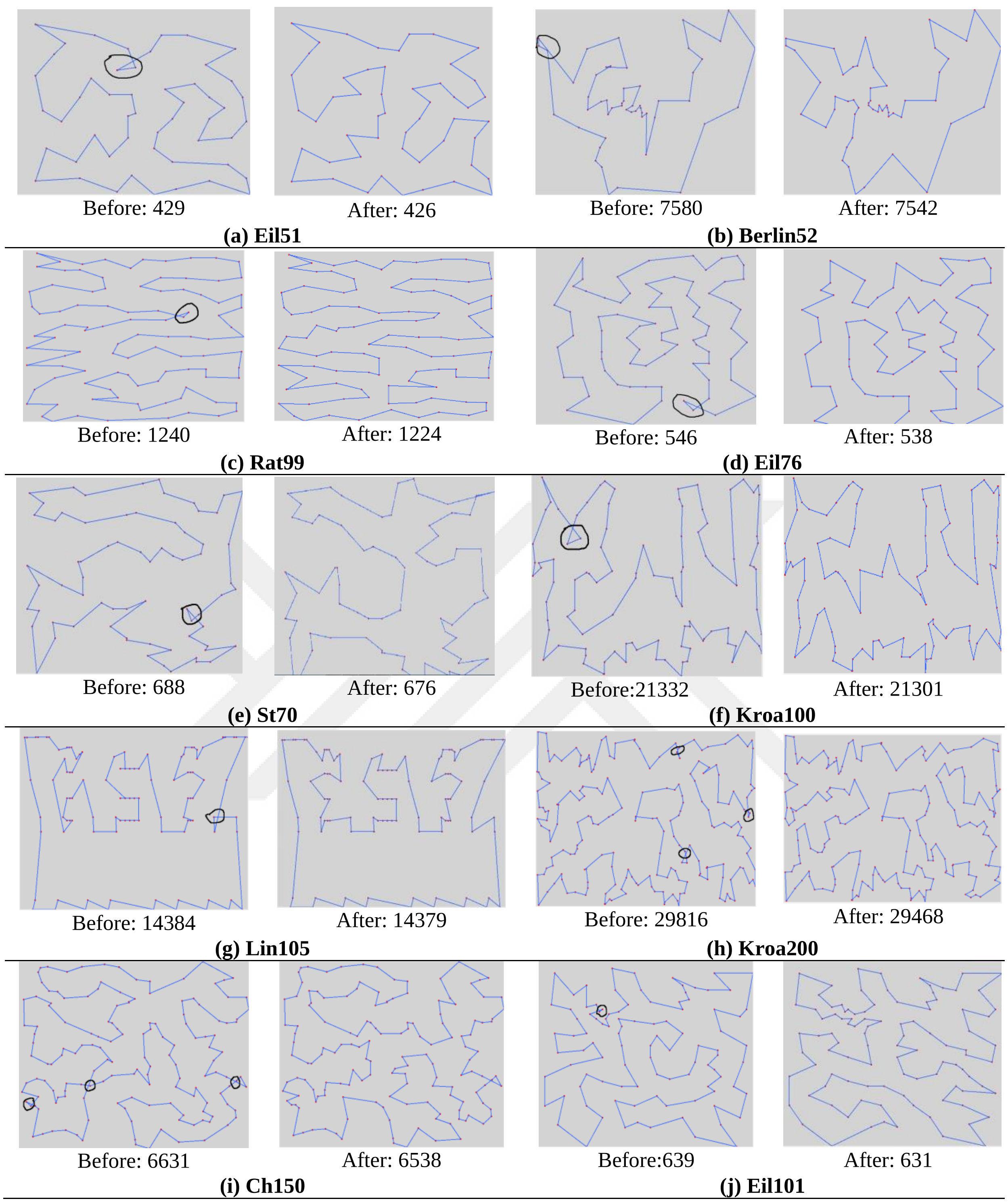


Figure 4.1 The best paths are specified before and after the 3-Opt implementation by the proposed hybrid method PSO, ACO and 3-Opt (Mahi et al., 2015).

To calculate the average, standard deviation and error percentage, the proposed method has been implemented 20 times. The error percentage is specified as Eq. 4.1 (Chen and Chien, 2011b). The obtained results of the proposed method are presented in

Table 4.3. The results of method assessment are introduced in Table 4.3 in comparison with previously worked studies. The best results are included in bold.

$$Error\ percentage = \frac{(average\ solution - best\ known\ solution)}{best\ known\ solution} \times 100 \tag{4.1}$$

Table 4.3 The results of the proposed hybrid method PSO, ACO and 3-Opt for test problems (Mahi et al., 2015)**.**

Problem	BKS*	Best	Worst	Average	SD**	Err(%)***	Time(s)
Eil51	426	426	428	426.45	0.61	0.11	140.50
Berlin52	7542	7542	7548	7543.20	2.37	0.02	170.46
Rat99	1224	1224	1230	1227.40	1.98	0.28	284.09
Eil76	538	538	539	538.30	0.47	0.06	220.68
St70	675	676	681	678.20	1.47	0.47	256.89
Kroa100	21282	21301	21554	21445.10	78.24	0.77	301.32
Lin105	14379	14379	14381	14379.15	0.48	0.00	294.35
Kroa200	29368	29468	29957	29646.05	114.71	0.95	303.23
Ch150	6528	6538	6622	6563.95	27.58	0.55	286.90
Eil101	629	631	638	632.70	2.12	0.59	302.15

* Best Known Solution ** Standard Deviation

*** Relative error for the results taken by 20 runs

Table 4.4 reviewed demonstrate that the proposed method has the nearby results for an optimal solution by a minimum standard deviation for Eil51, Rat99, Eil76, St70, Kroa200 and Eil101 problems. The average results acquired for Eil51, Rat99, Eil76, St70, Kroa200 and Eil101 are 426.45, 1227.40, 538.30, 678.20, 29646.05 and 632.70, respectively. As will be seen from Table 3.4, these results are better than the results of studies in the literature. It has also been observed that the results are close to the optimal solution for Berlin52, Lin105 and Ch150, as well as reasonable results in Kroa100. For dataset CH150, the previous literature has gained 6563.70. While our proposed method is 6563.95, it is close to the previous values. We obtained a minimum error percentage (0.55%) for the CH150 dataset. By examining the results in the literature, the results obtained by the proposed method have better results.

Table 4.4 The computational results of the PSO, ACO and 3-Opt and other methods in the literature. BKS is the best known solution; **Avg** is the average route length; SD is the standard deviation; Error(%) is relative error (Mahi et al., 2015).

Method	Problem	Eil51	Berlin52	Rat99	Eil76	St70	Kroa100	Lin105	Kroa200	Ch150	Eil101
	BKS	426	7542	1224	538	675	21282	14379	29368	6528	629
ACOMAC (2004) (Tsai et al., 2004)	Avg.	430.68	-	-	555.70	-	21457.00	-	-	-	-
	SD	-	-	-	-	-	-	-	-	-	-
	Error(%)	1.10	-	-	3.29	-	0.82	-	-	-	-
ACOMAC+NN (2004) (Tsai et al., 2004)	Avg.	430.68	-	-	555.90	-	21433.30	-	-	-	-
	SD	-	-	-	-	-	-	-	-	-	-
	Error (%)	1.10	-	-	3.33	-	0.71	-	-	-	-
RABNET-TSP (2006) (Pasti and De Castro, 2006)	Avg.	438.70	8073.97	-	556.10	-	21868.47	14702.17	30257.53	6753.20	654.83
	SD	3.52	270.14	-	8.03	-	245.76	328.37	342.98	83.01	6.57
	Error(%)	2.98	7.05	-	3.36	-	2.76	2.25	3.03	3.45	4.11
Modified RABNET-TSP (2009) (Masutti and de Castro, 2009b)	Avg.	437.47	7932.50	-	556.33	-	21522.73	14400.7	30190.27	6738.37	648.63
	SD	4.20	277.25	-	5.30	-	93.34	44.03	273.38	76.14	3.85
	Error(%)	2.69	5.18	-	3.41	-	1.13	0.15	2.80	3.22	3.12
SA ACO PSO (2011) (Chen and Chien, 2011b)	Avg.	427.27	**7542.00**	-	540.20	-	21370.30	14406.37	29738.73	**6563.70**	635.23
	SD	0.45	0.00	-	2.94	-	123.36	37.28	356.07	22.45	3.59
	Error(%)	0.30	0.00	-	0.41	-	0.41	0.19	1.26	0.55	0.99
IVRS+2opt (2012) (Jun-man and Yi, 2012)	Avg.	431.10	7547.23	-	-	-	21498.61	-	-	-	648.67
	SD	-	-	-	-	-	-	-	-	-	-
	Error(%)	1.20	0.07	-	-	-	1.02	-	-	-	3.13
ACO+2opt (2012) (Jun-man and Yi, 2012)	Avg.	439.25	7556.58	-	-	-	23441.80	-	-	-	672.37
	SD	-	-	-	-	-	-	-	-	-	-
	Error(%)	3.11	0.19	-	-	-	10.15	-	-	-	6.90
HACO (2012) (Junqiang and Aijia, 2012)	Avg.	431.20	7560.54	1241.33	-	-	-	-	-	-	-
	SD	2.00	67.48	9.60	-	-	-	-	-	-	-
	Error(%)	1.22	0.23	1.42	-	-	-	-	-	-	-
CGAS (2012) (Dong et al., 2012b)	Avg.	-	7634.00	-	542.00	-	21437.00	-	29946.00	-	-
	SD	-	-	-	-	-	-	-	-	-	-
	Error(%)	-	1.22	-	0.74	-	0.73	-	1.97	-	-
WFA with 2-Opt (2013) (Othman et al., 2013)	Avg.	426.65	**7542.00**	-	541.22	-	**21282.00**	**14379.00**	29654.03	6572.13	639.87
	SD	0.66	0.00	-	0.66	-	0.00	0.00	151.42	13.84	2.88
	Error(%)	0.15	0.00	-	0.60	-	0.00	0.00	0.97	0.68	1.73
WFA with 3-Opt (2013) (Othman et al., 2013)	Avg.	426.60	**7542.00**	-	539.44	-	21282.80	14459.40	29646.50	6700.10	633.50
	SD	0.52	0.00	-	1.51	-	0.00	1.38	110.91	60.82	3.47
	Error(%)	0.14	0.00	-	0.27	-	0.00	0.56	0.95	2.64	0.72
ACO with Tagushi Method (2013) (Peker et al., 2013a)	Avg.	435.40	7635.40		565.50		21567.10	14475.20			655.00
	SD	-	-	-	-	-	-	-	-	-	-
	Error(%)	2.21	1.24	-	5.11	-	1.34	0.67	-	-	4.13
ACO with ABC (2014) (Gündüz et al., 2014)	Avg.	443.39	7544.37	-	557.98	700.58	22435.31	-	-	6677.12	683.39
	SD	5.25	0.00	-	4.10	7.51	231.34	-	-	19.30	6.56
	Error(%)	4.08	0.03	-	3.71	3.79	5.42	-	-	2.28	8.65
Proposed Method **PSO-ACO-3Opt**	Avg.	**426.45**	7543.20	**1227.40**	**538.30**	**678.20**	21445.10	14379.15	**29646.05**	6563.95	**632.70**
	SD	0.61	2.37	1.98	0.47	1.47	78.24	0.48	114.71	27.58	2.12
	Error(%)	0.11	0.02	0.28	0.06	0.47	0.77	0.00	0.95	0.55	0.59

4.2 Experimental Results of Proposed Method Based On PSO For Solving DAP

To compare PSO-DAP with the Alternative algorithms, the original dataset is first generated using the cost formulation in Section 2.6. In the article (Tosun, 2014b) the number of fragments and sites is the same for comparing the results with the previous literature. Due to the fact that other algorithms use the same method, twenty different sites or the number of a fragment taken from 5 to 100 are used. The PSO-DAP input parameters are shown in Table 4.5, as with the parameters of the (Adl and Rankoohi, 2009) it is considered.

To get the average results, the program has been executed 20 times. The number of iteration of the PSO-DAP algorithm was considered to be 500 times. To compare the results of the PSO-DAP algorithm with previous literature for the DAP problem. The program runs on a computer with specifications of 1.6 GHz processor, 4 GB of memory and Windows 7 operating system in the C # program.

Table 4.5 Input parameters for PSO-DAP.

Parameter description	**Parameter Name**	**Value**
Approximation of the average fragment size	C	*10*
Unit transmission cost between two neighbour sites	UCN	*[0-1]*
Number of transactions	L	*20*
Probability of a transaction being requested at a site	RPT	*0.7*
Probability of a fragment being accessed by a transaction	APF	*0.4*
Probability of a transaction necessitates data transaction between Two sites (other than the originating site)	$APFS$	*0.025*
Number of particle	P	*30*
Learning factors	c_1, c_2	*2*
Number of iteration	K	*500*
Inertia weight	W	*0,5*
Maximum velocity	V_{max}	*N*
Generate random number	$rand_1, rand_2$	*[0-1]*

We have generated random data according to the data formulas and we show the cost and time cost in Table 4.6. The results of the PSO-DAP method in compare with other literature methods, ACO, RTS, GA and Hybrid Genetic Multi-Start Tabu Search Algorithm (HG-MTS) (Tosun, 2014b). A comparison that cover s cost and time values are shown in Table 4.7 and Table 4.8, respectively. The best results are expressed in bold.

Table 4.6 Generate first cost and execution time for increasing DAP instance sizes.

Size	Cost×10^6	Time (s)	Size	Cost×10^6	Time (s)
5	*0.07*	*0.14*	*55*	*145.23*	*25.18*
10	*0.21*	*0.20*	*60*	*212.56*	*55.97*
15	*0.55*	*0.75*	*65*	*290.99*	*147.06*
20	*2.49*	*0.89*	*70*	*319.86*	*166.73*
25	*8.91*	*1.56*	*75*	*430.69*	*197.95*
30	*9.65*	*2.28*	*80*	*594.30*	*253.48*
35	*25.67*	*3.17*	*85*	*771.51*	*243.56*
40	*43.91*	*6.13*	*90*	*1047.78*	*323.75*
45	*73.52*	*10.83*	*95*	*1274.53*	*331.08*
50	*112.09*	*21.04*	*100*	*1487.04*	*337.76*

Table 4.7 Cost comparison of methods for increasing DAP instance sizes.

Size	ACO (Tosun, 2014b) $x10^6$	RTS (Tosun, 2014b) $x10^6$	GA1 (Tosun, 2014b) $x10^6$	GA2 (Tosun, 2014b) $x10^6$	GA3 (Tosun, 2014b) $x10^6$	HG-MTS (Tosun, 2014b) $x10^6$	SA (Tosun et al., 2013b) $x10^6$	PSO-DAP (Proposed Methods)	
								Average X10^6	Standard Deviation
5	*0.04*	*0.04*	*0.04*	*0.04*	*0.04*	*0.04*	*0.04*	***0.02***	*0.0007*
10	*0.31*	*0.32*	*0.31*	*0.31*	*0.31*	*0.31*	*0.31*	***0.05***	*0.0130*
15	*0.98*	*0.99*	*0.98*	*0.98*	*0.98*	*0.98*	*0.98*	***0.41***	*0.0532*
20	*2.61*	*2.63*	*2.64*	*2.64*	*2.61*	*2.61*	*2.61*	***0.77***	*0,0816*
25	*5.19*	*5.25*	*5.26*	*5.24*	*5.19*	*5.15*	*5.15*	***3.74***	*1.2826*
30	*10.27*	*10.39*	*10.42*	*10.41*	*10.27*	*10.27*	*10.27*	***3.19***	*1.0470*
35	*16.39*	*16.64*	*16.61*	*16.66*	*16.39*	*16.41*	*16.41*	***9.04***	*4.5046*
40	*25.9*	*26.28*	*26.33*	*26.21*	*25.92*	*26.02*	*26.02*	***19.24***	*8.9307*
45	*37.26*	*37.73*	*37.8*	*37.82*	*37.27*	*37.40*	*37.40*	***27.04***	*16.7220*
50	*53.89*	*54.76*	*54.63*	*54.69*	*53.88*	*54.08*	*54.08*	***34.43***	*25.6230*
55	*71.19*	*72.72*	*72.40*	*72.13*	*71.21*	*71.40*	*71.40*	***51.38***	*37.4836*
60	*90.16*	*91.76*	*91.49*	*91.56*	***90.20***	*90.50*	*90.50*	*97.78*	*59.9261*
65	*112.13*	*113.59*	*113.75*	*113.84*	***112.08***	*112.49*	*112.49*	*125.01*	*91.0824*
70	*146.19*	*148.48*	*148.8*	*148.18*	*146.15*	*146.73*	*146.73*	***138.69***	*109.8225*
75	*177.7*	*180.04*	*180.75*	*180.63*	*177.65*	*178.16*	*178.16*	***171.47***	*139.8139*
80	*219.26*	*223.10*	*222.80*	*222.96*	***219.18***	*219.81*	*219.81*	*260.86*	*160.2445*
85	*261.88*	*267.04*	*266.15*	*266.19*	*261.99*	*262.89*	*262.89*	***260.63***	*240.2334*
90	*315.86*	*320.88*	*320.93*	*320.58*	*315.86*	*316.81*	*316.81*	***287.09***	*302.8202*
95	*369.92*	*375.49*	*375.85*	*375.29*	*369.91*	*371.14*	*371.14*	***365.06***	*392.3170*
100	*428.28*	*436.19*	*436.15*	*434.45*	***427.98***	*429.10*	*429.10*	*481.58*	*459.9464*

Table 4.8 Execution time (s) comparison of methods for increasing DAP instance sizes.

DAP Size	ACO (Tosun, 2014b)	RTS (Tosun, 2014b)	GA1 (Tosun, 2014b)	GA2 (Tosun, 2014b)	GA3 (Tosun, 2014b)	HG-MTS (Tosun, 2014b)	SA (Tosun et al., 2013b)	PSO-DAP (Proposed Method)
5	9.26	0.83	76.27	56.11	88.11	1.44	130.29	**0.74**
10	14.52	2.73	87.80	60.37	94.91	2.45	143.84	**1.35**
15	13.74	5.66	90.76	66.22	104.13	2.65	214.30	**2.17**
20	17.91	8.89	123.79	84.13	167.22	4.17	243.30	**3.65**
25	25.86	14.52	131.98	81.96	125.30	5.21	351.23	**4.25**
30	31.17	20.89	132.46	104.64	137.02	7.38	461.89	**6.45**
35	43.31	29.06	150.06	111.87	151.02	10.73	393.73	**6.81**
40	56.59	37.05	166.80	128.75	173.21	15.60	420.65	**8.85**
45	80.92	48.67	191.93	159.10	202.10	20.80	437.74	**8.42**
50	105.33	62.74	471.98	207.56	359.57	26.80	511.40	**9.60**
55	126.00	76.07	268.31	201.43	261.71	27.22	516.86	**13.74**
60	166.55	91.79	315.31	208.37	290.46	39.56	828.14	**16.09**
65	204.35	109.20	421.93	284.08	336.01	48.92	1090.77	**16.09**
70	320.62	131.54	536.15	344.20	358.03	63.13	1303.21	**17.34**
75	309.51	155.31	609.77	379.07	380.81	73.41	976.97	**17.01**
80	396.18	193.63	464.17	331.17	416.18	87.84	1234.48	**16.29**
85	807.43	195.80	532.05	364.71	586.21	102.79	898.11	**18.31**
90	621.55	215.58	563.15	400.37	531.13	123.19	1336.74	**20.98**
95	725.93	250.72	629.55	974.24	569.92	143.16	1128.08	**18.15**
100	1203.99	278.63	1236.30	568.73	808.82	179.07	1389.19	**20.97**

In order to show the proposed methods performance difference from the performance of other algorithms in the literature sign test has been used (Lurie et al., 2011; Mann, 2013). There is no significant difference between the two algorithms as the H0 hypothesis. There is a significant difference between the two algorithms as the H1 hypothesis. The calculations were carried out at the 5% significance level. The results of the entrance exam are accurate in Table 4.8. In Table 4.9. each algorithm is compared with the proposed method. The statistical comparison of the proposed method and the other method is presented in six rows of the Table 4.9. The description of the parameters $(S, X_s, \sigma_s, Z, \frac{Z_{0.05}}{2}, H_0, H_1)$ is given in Table 2.1. A negative sign means a poor result when comparing the two columns being compared, a positive sign means a good result. The H1 hypothesis was accepted because the values of Z calculated outside the range in all tests ($|\mathrm{Z\ Values}| > |\pm 1.96|$).

Table 4.9 Statistical comparison of the methods using sign test.

Size	PSO-DAP (Proposed Method)	ACO (Tosun, 2014b)	Sign	RTS (Tosun, 2014b)	Sign	GA1 (Tosun, 2014b)	Sign	GA2 (Tosun, 2014b)	Sign	GA3 (Tosun, 2014b)	Sign	HG-MTS (Tosun, 2014b)	Sign	SA (Tosun et al., 2013b)	Sign
5	***0.02***	*0.04*	-	*0.04*	-	*0.04*	-	*0.04*	-	*0.04*	-	*0.04*	-	*0.04*	-
10	***0.05***	*0.31*	-	*0.31*	-	*0.32*	-	*0.31*	-	*0.31*	-	*0.31*	-	*0.31*	-
15	***0.41***	*0.98*	-	*0.98*	-	*0.99*	-	*0.98*	-	*0.98*	-	*0.98*	-	*0.98*	-
20	***0.77***	*2.61*	-	*2.61*	-	*2.63*	-	*2.64*	-	*2.64*	-	*2.61*	-	*2.61*	-
25	***3.74***	*5.19*	-	*5.19*	-	*5.25*	-	*5.26*	-	*5.24*	-	*5.19*	-	*5.15*	-
30	***3.19***	*10.27*	-	*10.27*	-	*10.39*	-	*10.42*	-	*10.41*	-	*10.27*	-	*10.27*	-
35	***9.04***	*16.39*	-	*16.39*	-	*16.64*	-	*16.61*	-	*16.66*	-	*16.39*	-	*16.41*	-
40	***19.24***	*25.91*	-	*25.9*	-	*26.28*	-	*26.33*	-	*26.21*	-	*25.92*	-	*26.02*	-
45	***27.04***	*37.28*	-	*37.26*	-	*37.73*	-	*37.8*	-	*37.82*	-	*37.27*	-	*37.40*	-
50	***34.43***	*53.93*	-	*53.89*	-	*54.76*	-	*54.63*	-	*54.69*	-	*53.88*	-	*54.08*	-
55	***51.38***	*71.30*	-	*71.19*	-	*72.72*	-	*72.40*	-	*72.13*	-	*71.21*	-	*71.40*	-
60	*97.78*	*90.35*	+	*90.16*	+	*91.76*	+	*91.49*	+	*91.56*	+	***90.20***	+	*90.50*	+
65	*125.01*	*112.31*	+	*112.13*	+	*113.59*	+	*113.75*	+	*113.84*	+	***112.08***	+	*112.49*	+
70	***138.69***	*146.41*	-	*146.19*	-	*148.48*	-	*148.8*	-	*148.18*	-	*146.15*	-	*146.73*	-
75	***171.47***	*177.90*	-	*177.7*	-	*180.04*	-	*180.75*	-	*180.63*	-	*177.65*	-	*178.16*	-
80	*260.86*	*219.40*	-	*219.26*	+	*223.10*	+	*222.80*	+	*222.96*	+	***219.18***	+	*219.81*	+
85	***260.63***	*262.24*	-	*261.88*	-	*267.04*	-	*266.15*	-	*266.19*	-	*261.99*	-	*262.89*	-
90	***287.09***	*316.11*	-	*315.86*	-	*320.88*	-	*320.93*	-	*320.58*	-	*315.86*	-	*316.81*	-
95	***365.06***	*370.14*	-	*369.92*	-	*375.49*	-	*375.85*	-	*375.29*	-	*369.91*	-	*371.14*	-
100	*481.58*	*428.40*	+	*428.28*	+	*436.19*	+	*436.15*	+	*434.45*	+	***427.98***	+	*429.10*	+

Statistical Notations	PSO-DAP vs. ACO	PSO-DAP vs. RTS	PSO-DAP vs. GA1	PSO-DAP vs. GA2	PSO-DAP vs. GA3	PSO-DAP vs. HG-MTS	PSO-DAP vs. SA
S	*3*	*4*	*4*	*4*	*4*	*4*	*4*
X_s	*10*	*10*	*10*	*10*	*10*	*10*	*10*
σ_s	*2.236*	*2.236*	*2.236*	*2.236*	*2.236*	*2.236*	*2.236*
Z	*-3.130*	*-2.683*	*-2.683*	*-2.683*	*-2.683*	*-2.683*	*-2.683*
$Z_{\frac{0.05}{2}}$	*±1.96*	*±1.96*	*±1.96*	*±1.96*	*±1.96*	*±1.96*	*±1.96*
$H0$	*Reject*	*Reject*	*Reject*	*Reject*	*Reject*	*Reject*	*Reject*
$H1$	***Accept***	***Accept***	***Accept***	***Accept***	***Accept***	***Accept***	***Accept***

The obtained results demonstrate that PSO-DAP is less costly and time-consuming than other methods (ACO, RTS, GA, HG-MTS, etc.). In this thesis, DAP instances are generated randomly and the PSO-DAP is used to solve these problems. To compare the PSO-DAP with new studies, researchers can use these instances in future studies.

Due to the lack of a dataset for the comparator algorithm, we randomly generate the data set according to the formulas described in Section 2.2. In terms of cost, among the twenty exiting results, sixteen are the best and four of them (60, 65, 80 and 100 sample sizes) are close to the results of our comparison algorithm in Table 4.7. As shown in Table 4.9, the proposed method is statistically significantly different from other methods in the literature. Consequently, in terms of time-consuming concept, PSO-DAP uses a short time in compare with other comparison algorithms in Table 4.8.

4.3 Experimental Results of Proposed Method Based On Greedy Algorithm For Solving DAP

The original data set has been produced based on cost formulation in section 2.6 in order to compare DAP with other algorithms. Three cases of comparisons in various fragment and site size are determined in the proposed algorithm. In the first case, the number of fragments are equal but the number of sites are variable. In the second case, the number of sites are equal but the number of fragments are different. The third case contains the same size of fragments and sites. These three cases are tested in the proposed algorithm. It is worth to note that the proposed algorithm is deterministic. This algorithm iterations results are the same. However, due to the random dataset, other algorithms cost and process time results are different. In order to get the minimum number of results, the program is tested in 20 iteration of executions in PSO-DAP whereas Greedy DAP due to giving the same results in all performances, only one iteration of execution is done. PSO-DAP iteration is done in 500 repeats. Computer parameters for comparing PSO-DAP and Greedy DAP based on an algorithm that is presented for DAP can be listed as follow: CPU 1.6 GHZ, memory 4 GB, Windows 7 and C# programming language.

4.4 Various States of Fragments and sites

4.4.1   State 1: The site size increment from 3 to 48 and fixing the number of fragments in 48.

The first state refers to the site size increment from 3 to 48 and fixing the number of fragments in 48. The result values of the algorithms in thesis (Adl and Rankoohi, 2009) has been shown just in the figure. The results of cost and process time for PSO-DAP and Greedy DAP has been shown in Table 4.11 and 4.12 respectively. In order to compare the obtained results of the proposed algorithm with figure in thesis (Adl and Rankoohi, 2009), we have mapped proposed algorithm results to those charts. Considering that other algorithms used the same method forty-six different sites numbers that are taken range from 3 to 48 and the number of the fragment is fixed by 48. Input parameters in Table

4.10 which are shown in bold refer to PSO-DAP and others are common in both PSO-DAP and Greedy DAP.

Table 4.10 Input parameters for PSO-DAP proposed method (Adl and Rankoohi, 2009)**.**

Parameter description	Parameter Name	Value
Approximation of the average fragment size	C	*10*
Unit transmission cost between two neighbor sites	UCN	*[0-1]*
Number of transactions	L	*20*
Probability of a transaction being requested at a site	RPT	*0.7*
Probability of a fragment being accessed by a transaction	APF	*0.4*
Probability of a transaction necessitates data transaction between two sites (other than the originating site)	$APFS$	*0.025*
Number of particle	$\boldsymbol{P}$	*30*
Learning factors	$\boldsymbol{c_1, c_2}$	*2*
Number of iteration	$\boldsymbol{K}$	*500*
Inertia weight	$\boldsymbol{W}$	*0,5*
Maximum velocity	$\boldsymbol{V_{max}}$	*N*
Generate random number	$\boldsymbol{rand_1}$, $\boldsymbol{rand_2}$	*[0-1]*

DAP samples sizes increment cost values and process time generation has been shown in Table 4.11. The obtained results of the Greedy DAP is compared with the other methods in the literature, Ant γ Algorithm (Adl and Rankoohi, 2009), Ant β Algorithm (Adl and Rankoohi, 2009), Ant α Algorithm (Adl and Rankoohi, 2009), Evolutionary (Adl and Rankoohi, 2009) and PSO-DAP minimum proposed method (Mahi et al., 2018).

Table 4.11 Generate first cost and execution time for increasing site numbers (n) and fragment number is fixed by 48.

n	Cost×10^9	Time (s)	n	Cost×10^9	Time (s)
3	0.002	0.902	26	0.192	8.093
4	0.004	1.164	27	0.223	8.470
5	0.005	1.488	28	0.232	9.236
6	0.009	1.909	29	0.247	9.330
7	0.015	2.275	30	0.236	10.254
8	0.025	2.450	31	0.262	10.297
9	0.019	2.750	32	0.300	10.175
10	0.030	3.166	33	0.324	10.692
11	0.043	3.498	34	0.301	11.845
12	0.036	4.069	35	0.330	12.273
13	0.047	4.158	36	0.384	11.525
14	0.056	4.264	37	0.347	11.640
15	0.042	4.662	38	0.427	13.358
16	0.075	5.216	39	0.414	13.810
17	0.082	5.464	40	0.468	14.476
18	0.099	5.567	41	0.527	13.810
19	0.110	6.143	42	0.544	13.892
20	0.085	6.073	43	0.522	14.702
21	0.104	7.674	44	0.565	16.328
22	0.142	6.941	45	0.639	15.306
23	0.151	7.816	46	0.597	16.405
24	0.167	8.070	47	0.619	17.472
25	0.128	8.092	48	0.671	15.557

Table 4.12 and Table 4.13 contains cost and process time values respectively and the best results are shown in bold.

Table 4.12 Cost comparison of methods for increasing site numbers (n) and fragment number is fixed by 48.

n	PSO-DAP (Proposed Method)			Greedy DAP (Proposed Method) x10^9	n	PSO-DAP (Proposed Method)			Greedy DAP (Proposed Method) x10^9
	Average x10^9	Minimum x10^9	Standard Deviation			Average x10^9	Minimum x10^9	Standard Deviation	
3	0.0004	**0.0004**	0.000	**0.0004**	26	0.183	0.178	0.008	**0.173**
4	0.002	**0.002**	0.000	**0.002**	27	0.216	0.211	0.020	**0.201**
5	0.003	**0.003**	0.000	**0.003**	28	0.225	0.221	0.025	**0.209**
6	0.004	**0.004**	0.000	**0.004**	29	0.244	0.235	0.016	**0.206**
7	0.009	0.009	0.000	**0.008**	30	0.226	0.216	0.022	**0.201**
8	0.022	**0.022**	0.000	0.023	31	0.255	0.253	0.019	**0.235**
9	0.017	0.016	0.004	**0.013**	32	0.301	0.294	0.012	**0.260**
10	0.025	0.023	0.005	**0.020**	33	0.308	0.306	0.030	**0.272**
11	0.037	0.035	0.001	**0.032**	34	0.290	0.277	0.044	**0.271**
12	0.031	**0.030**	0.001	0.033	35	0.319	0.314	0.028	**0.281**
13	0.035	**0.033**	0.000	0.036	36	0.372	0.370	0.048	**0.330**
14	0.042	**0.042**	0.001	0.044	37	0.336	0.333	0.051	**0.321**
15	0.035	0.033	0.001	**0.026**	38	0.419	0.415	0.078	**0.397**
16	0.071	0.061	0.002	**0.059**	39	0.388	0.381	0.078	**0.341**
17	0.074	0.070	0.002	**0.066**	40	0.436	0.434	0.113	**0.402**
18	0.086	0.084	0.003	**0.082**	41	0.520	0.512	0.069	**0.477**
19	0.101	0.100	0.004	**0.088**	42	0.535	0.533	0.077	**0.480**
20	0.071	0.064	0.002	**0.064**	43	0.511	0.509	0.116	**0.488**
21	0.094	0.090	0.005	**0.088**	44	0.541	0.526	0.111	**0.472**
22	0.114	0.113	0.002	**0.109**	45	0.616	0.614	0.109	**0.585**
23	0.137	0.134	0.006	**0.121**	46	0.579	0.561	0.163	**0.502**
24	0.158	0.156	0.017	**0.138**	47	0.621	0.616	0.166	**0.565**
25	0.125	0.123	0.005	**0.119**	48	0.641	0.634	0.109	**0.590**

Comparison of obtained results demonstrates that Greedy DAP has less cost and process time-consuming in compare with PSO-DAP. DAP samples are created randomly and Greedy DAP and PSO-DAP used them to solve problems in this thesis. Datasets are created randomly according to the formula described in section 2.2 due to the absence of compared algorithm’s datasets. Results demonstrate that, among 46 obtained results, 39 of them are the best in terms of cost and are almost similar to our compared algorithm's results in Table 4.13 Table 4.14 has shown that, the proposed algorithm is different from other methods statistically. Consequently, PSO-DAP consumes less time than other algorithms as shown in Table 4.14.

Table 4.13 Execution time (s) comparison of methods for increasing site numbers (n) and fragment number is fixed by 48.

n	PSO-DAP Minimum (Proposed Method)	Greedy DAP (Proposed Method)	n	PSO-DAP Minimum (Proposed Method)	Greedy DAP (Proposed Method)
3	15.466	0.366	26	18.158	7.956
4	16.021	0.069	27	16.910	8.767
5	15.538	0.133	28	17.082	9.438
6	15.007	0.210	29	16.942	10.296
7	15.694	0.299	30	16.630	10.967
8	15.772	0.455	31	16.973	11.872
9	15.787	0.608	32	16.770	12.683
10	15.818	0.718	33	16.942	13.603
11	16.006	0.905	34	17.176	15.803
12	15.928	1.061	35	17.035	16.068
13	15.694	1.232	36	17.082	16.520
14	15.694	1.498	37	17.410	17.456
15	15.803	1.685	38	17.098	19.578
16	15.678	1.950	39	19.016	20.608
17	17.690	2.200	40	16.957	23.026
18	17.893	2.761	41	16.957	24.508
19	18.174	3.635	42	17.534	25.802
20	17.612	3.635	43	17.300	27.035
21	18.580	4.664	44	18.143	28.548
22	19.781	5.320	45	18.923	30.420
23	19.687	5.897	46	17.160	32.105
24	16.739	6.646	47	16.879	33.680
25	16.926	7.348	48	16.754	33.758

Performance of proposed algorithm is different from alternatives and this is calculated using sign test (Lurie et al., 2011; Mann, 2013). H0 hypothesis has no outstanding difference between two algorithms but H1 hypothesis show difference between them. All calculations were implemented at a level of five percent significance. Table 4.13 contains sign test results. The H1 hypotheses were accepted because the computed Z values are outside the range in all tests ($|\mathrm{Z\ Values}| > |\pm 1.96|$). Proposed algorithm and other Alternative ones comparison results are shown in the last two rows of Table 4.14. The description of the parameters ($S, X_S, \sigma_S, Z, \frac{Z_{0.05}}{2}, H_0, H_1$) is given in Table 2.1. A negative sign means a poor result when comparing the two columns being compared, a positive sign means a good result.

Table 4.14 Statistical comparison of the methods using sign test for increasing site numbers (n) and fragment number is fixed by 48.

n	Greedy DAP (Proposed Method)	PSO-DAP Minimum (Proposed Method)	Sign	n	Greedy DAP (Proposed Method)	PSO-DAP Minimum (Proposed Method)	Sign
3	441872	410603	+	26	172615295	177702535	-
4	1664678	1244572	+	27	200975127	211071504	-
5	2580880	2591430	-	28	208627304	221408655	-
6	3918779	3898362	+	29	205512776	234776499	-
7	8475686	8892369	-	30	201362336	216136938	-
8	22671850	21702160	+	31	234527270	253376973	-
9	12915310	16140468	-	32	260187978	293588991	-
10	20473330	23256179	-	33	272106382	306261617	-
11	32423834	35198738	-	34	271027460	277055891	-
12	33156448	30360039	+	35	280603467	314138626	-
13	35662487	33175773	+	36	330381584	370207013	-
14	44037891	41718648	+	37	320758035	333278681	-
15	26337352	32883834	-	38	397051238	414989215	-
16	59321311	60711667	-	39	341185743	381319377	-
17	66032543	70417580	-	40	401932655	433510525	-
18	82477752	84454729	-	41	477285889	512475049	-
19	88167561	99960603	-	42	479891345	532748715	-
20	64205904	64473334	-	43	487529241	508885582	-
21	87930403	89750233	-	44	471533110	526005587	-
22	109131359	113228406	-	45	585214613	614206988	-
23	121130352	134426817	-	46	501554620	560510603	-
24	138049507	156493230	-	47	565195696	615874059	-
25	118674870	122929074	-	48	590016599	633864873	-

Statistical Notations	Greedy DAP vs PSO-DAP
S	*7*
X_s	*22.5*
σ_s	*3.354*
Z	*-4.621*
$Z_{0.05}$	± 1.96
H_0	*Reject*
$\boldsymbol{H_1}$	***Accept***

Greedy DAP cost and time results comparisons with other methods such as Ant γ Algorithm (Adl and Rankoohi, 2009), Ant β Algorithm (Adl and Rankoohi, 2009), Ant α Algorithm (Adl and Rankoohi, 2009), Evolutionary (Adl and Rankoohi, 2009) and PSO-DAP minimum (Mahi et al., 2018) have been shown in Figure 4.2 and 4.3. Not paying attention to the results of the table in the (Adl and Rankoohi, 2009), we could not draw the comparison graph. So we had to add the results of the greedy algorithm and the PSO to the graph.

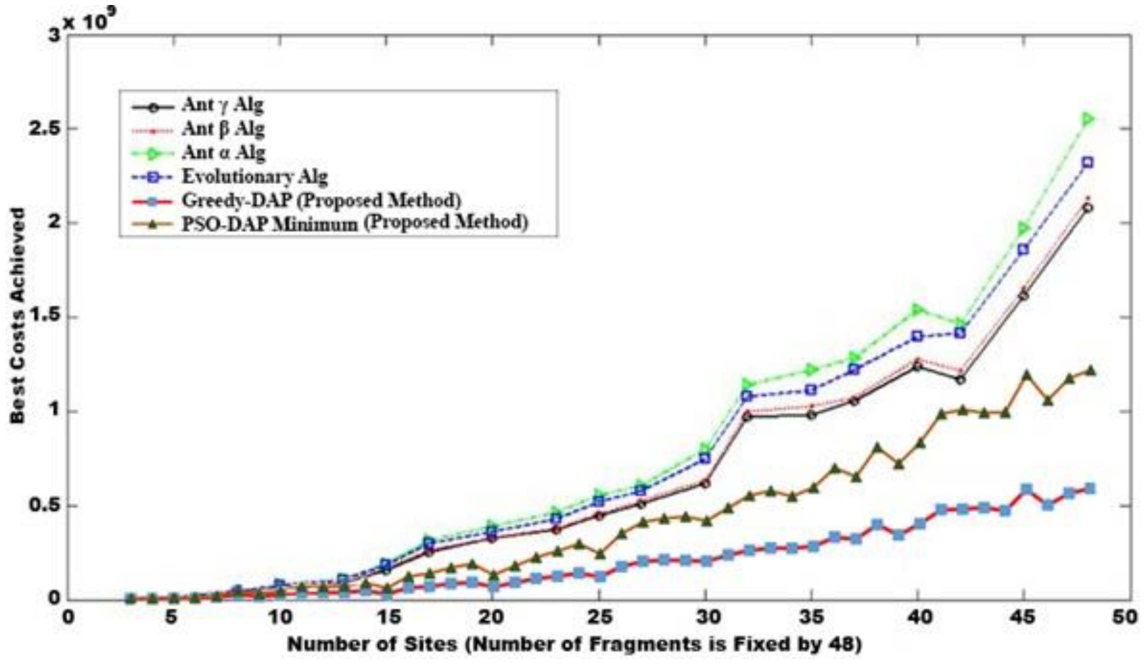


Figure 4.2 Evaluating the Results achieved by the algorithms in a state 1 comparison for cost (Adl and Rankoohi, 2009).

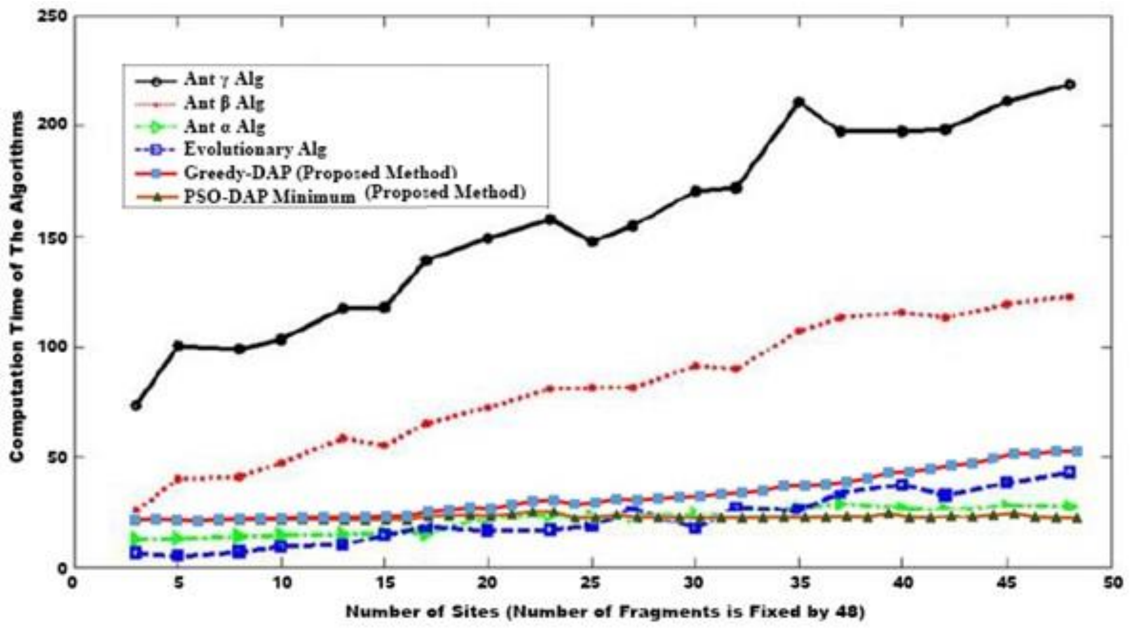


Figure 4.3 Evaluating the Computation time of the algorithms in a state 1 comparison for time (Adl and Rankoohi, 2009).

4.4.2 State 2: Fragment size increment from 20 up to 50 by step 1 and fix the number of sites in 20.

The second state of the algorithm refers to the fragment size increment from 20 up to 50 by step 1 and fix the number of sites in 20. Obtained results of the algorithm in thesis (Adl and Rankoohi, 2009) are shown in the Table 4.15. Results of the cost and process time of PSO-DAP and Greedy DAP are shown in Table 4.15 and 4.16. To compare the results of the thesis (Adl and Rankoohi, 2009) that are shown in figures, we have mapped our obtained results to those charts. So it can be inferred that other algorithms which are used the same methods with thirty fragment size, are taken their fragment range from 20 up to 50 and fix site size in 20. DAP samples size increment in terms of cost values and process time has been shown in Table 4.17. Greedy DAP and achieved results have been compared with other methods such as Ant γ Algorithm (Adl and Rankoohi, 2009), Ant β Algorithm (Adl and Rankoohi, 2009), Ant α Algorithm (Adl and Rankoohi, 2009), Evolutionary (Adl and Rankoohi, 2009) and PSO-DAP (Mahi et al., 2018).

Table 4.15 Generate first cost and execution time for increasing fragment numbers (m) and site number is fixed by 20.

m	Cost×10^8	Time (s)	m	Cost×10^8	Time (s)
20	0.268	0.961	36	0.690	5.343
21	0.227	1.215	37	0.686	6.132
22	0.278	1.189	38	0.674	7.018
23	0.254	1.384	39	0.655	7.775
24	0.348	1.566	40	0.758	8.390
25	0.364	1.590	41	0.951	9.807
26	0.342	1.894	42	0.803	10.488
27	0.351	1.864	43	0.823	12.391
28	0.419	2.362	44	0.955	14.276
29	0.523	2.450	45	1.083	14.880
30	0.586	2.721	46	0.978	18.289
31	0.485	3.290	47	1.025	19.571
32	0.515	3.820	48	1.135	20.931
33	0.507	4.167	49	1.114	24.177
34	0.649	4.160	50	1.178	25.570
35	0.621	5.017			

The comparison which cover s cost and process time values are given respectively in Table 4.16 and 4.17. The best results are highlighted in bold.

Table 4.16 Cost comparison of methods for increasing fragment numbers (m) and site number is fixed by 20.

m	PSO-Avg Proposed Method	PSO-min Proposed Method	Cost Proposed Method	Standard Deviation	m	PSO-Avg Proposed Method	PSO-min Proposed Method	Cost Proposed Method	Standard Deviation
20	0.252	0.249	0.244	0.035	36	0.600	0.589	0.657	0.059
21	0.220	0.208	0.219	0.030	37	0.606	0.600	0.619	0.010
22	0.277	0.267	0.256	0.047	38	0.664	0.638	0.564	0.028
23	0.250	0.247	0.228	0.033	39	0.616	0.584	0.532	0.033
24	0.334	0.330	0.331	0.013	40	0.694	0.682	0.675	0.023
25	0.307	0.304	0.312	0.041	41	0.923	0.906	0.858	0.037
26	0.316	0.314	0.309	0.027	42	0.763	0.761	0.656	0.066
27	0.354	0.339	0.312	0.029	43	0.720	0.711	0.668	0.053
28	0.373	0.367	0.369	0.034	44	0.839	0.812	0.847	0.004
29	0.482	0.472	0.404	0.042	45	0.985	0.943	0.867	0.053
30	0.566	0.563	0.578	0.041	46	0.848	0.837	0.797	0.033
31	0.463	0.429	0.429	0.039	47	0.866	0.847	0.928	0.066
32	0.483	0.471	0.436	0.015	48	1.061	1.030	0.910	0.105
33	0.480	0.468	0.444	0.048	49	1.121	1.016	1.014	0.070
34	0.583	0.572	0.512	0.030	50	1.029	1.004	1.033	0.200
35	0.575	0.562	0.527	0.041					

Table 4.17 Execution time (s) comparison of methods for increasing fragment numbers (m) and site number is fixed by 20.

m	PSO-DAP Minimum (Proposed Method)	Greedy DAP (Proposed Method)	m	PSO-DAP Minimum (Proposed Method)	Greedy DAP (Proposed Method)
20	3.111	0.218	36	3.472	2.028
21	3.422	0.203	37	3.686	1.888
22	3.113	0.265	38	4.146	2.168
23	2.814	0.281	39	3.564	2.558
24	3.091	0.374	40	3.741	2.902
25	3.507	0.421	41	3.300	3.370
26	3.197	0.437	42	3.399	3.416
27	3.856	0.499	43	4.352	3.292
28	3.984	0.608	44	3.774	4.243
29	4.593	0.796	45	3.259	5.538
30	4.290	0.764	46	4.037	6.193
31	3.988	0.998	47	3.540	7.004
32	4.678	1.076	48	3.862	7.738
33	3.444	1.030	49	3.372	8.658
34	2.995	1.466	50	3.187	9.142
35	3.532	1.685			

Performance of the proposed method is different from alternatives and this is calculated using sign the test (Lurie et al., 2011; Mann, 2013). H0 hypothesis has no outstanding difference between the two algorithms but H1 hypothesis show difference between them. All calculations were implemented at a level of five percent significance. Table 4.18 contains sign test results and each algorithm comparisons with proposed method. The statistical comparison of the proposed method and the other method is given in the last six rows of Table 4.18. The H1 hypotheses were accepted because the computed Z values are outside the range in all tests ($|\text{Z Values}| > |\pm 1.96|$). Table 4.18 for increasing fragment numbers (m) and site number is fixed.

Table 4.18 For increasing fragment numbers (m) and site number is fixed by 20. Obtained results of methods comparisons with sign test.

m	Greedy DAP (Proposed Method)	PSO-DAP Minimum (Proposed Method)	Sign	m	Greedy DAP (Proposed Method)	PSO-DAP Minimum (Proposed Method)	Sign
20	24433882	24872829	+	36	65701506	58860638	+
21	21919467	20829479	-	37	61854394	60038953	-
22	25623071	26714534	+	38	56381250	63792446	-
23	22791175	24670355	-	39	53194842	58383603	-
24	33091691	33007551	+	40	67450888	68192047	-
25	31215154	30397092	-	41	85806612	90561561	+
26	30898685	31405235	-	42	65616347	76080351	-
27	31228117	33875190	-	43	66842906	71074916	-
28	36943665	36689174	-	44	84666662	81243023	-
29	40435858	47213491	-	45	86716618	94333685	+
30	57775641	56279930	+	46	79691135	83669594	-
31	42873844	42876940	-	47	92754465	84678495	-
32	43620635	47103581	-	48	91001545	103032353	-
33	44429550	46819696	-	49	101370565	101586213	+
34	51215186	57234406	-	50	103267955	100401042	+
35	52734960	56175376	-				

Statistical Notations	Greedy DAP vs PSO Proposed Method
S	*9*
X_s	*15*
σ_s	*2.738*
Z	*-2.191*
$Z_{\frac{0.05}{2}}$	± 1.96
H_0	*Reject*
H_1	***Accept***

Achieved results demonstrate that PSO-DAP is less costly and time-consuming than other methods (ACO, RTS, GA, HG-MTS and etc.). In this thesis, DAP sample database is produced randomly and PSO-DAP is used them to solve problems. Datasets are created randomly according to a formula described in section 2.2 due to the absence of compared algorithm’s datasets. Obtained results demonstrate that among 20 achievements, 16 of them are close to our compared proposed method in Table 4.18. A proposed method in this study is significantly different from other methods. So PSO-DAP, as shown in Table 4.18 consumes less time in comparison with others. The Figure 4.4 and 4.5 show cost and process time results of the Greedy DAP in compare with other methods in the literature, Ant γ Algorithm, Ant β Algorithm, Ant α Algorithm, Evolutionary (Adl and Rankoohi, 2009) and PSO-DAP average and PSO-DAP minimum (Mahi et al., 2018).

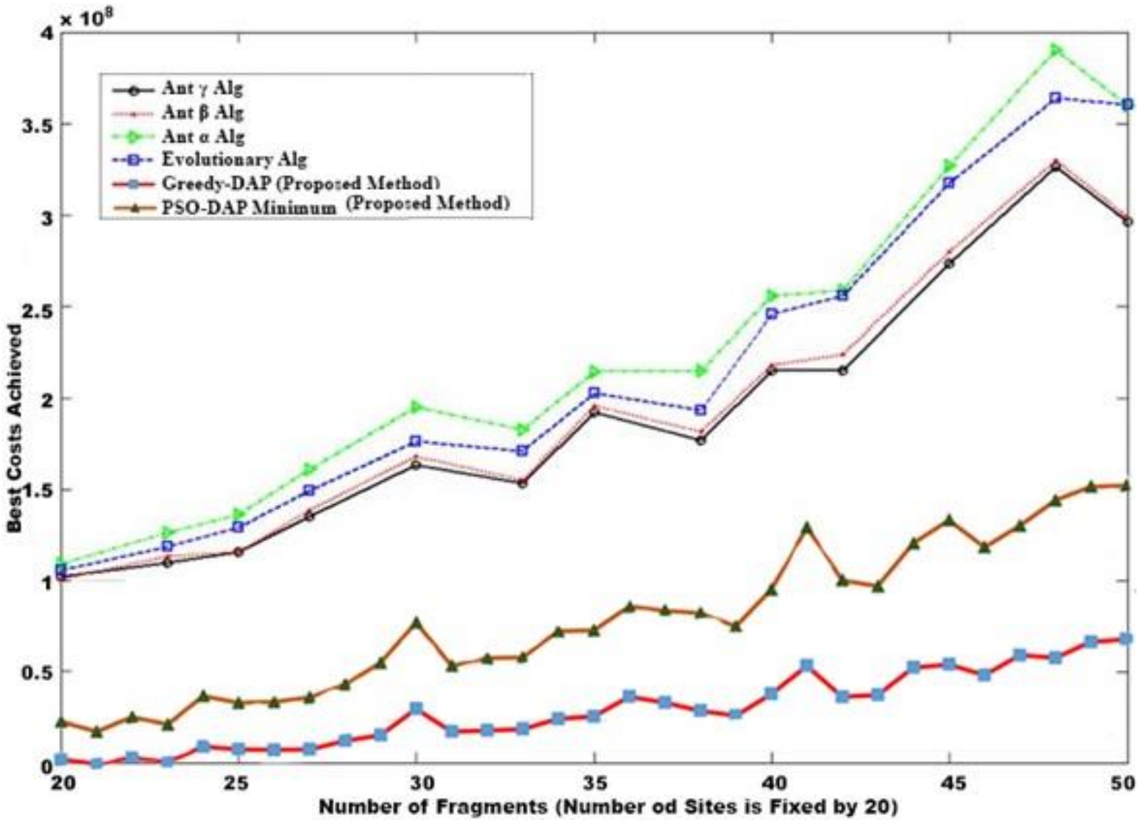


Figure 4.4 Evaluating the Results achieved by the methods in a state 2 comparison for cost (Adl and Rankoohi, 2009).

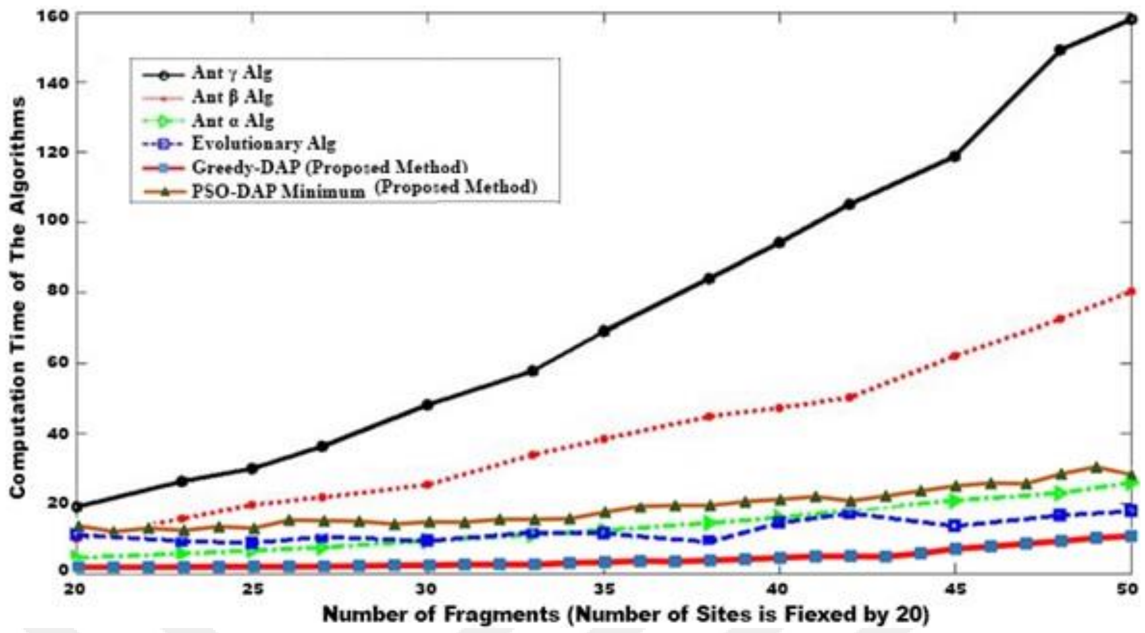


Figure 4.5 Evaluating the Computation time of the methods in a state 2 comparison for time (Adl and Rankoohi, 2009).

4.4.3 State 3: The number of fragments and sites are equal

The number of fragments and sites are equal so our proposed method is implemented on an equal number of fragments and sites. The cost and initial production time are shown in Table 4.19.

Table 4.19 ***Generate first cost and execution time for increasing DAP instance sizes.***

Size	Cost×10^6	Time (s)	Size	Cost×10^6	Time (s)
5	*0.07*	*0.14*	*55*	*145.23*	*25.18*
10	*0.21*	*0.20*	*60*	*212.56*	*55.97*
15	*0.55*	*0.75*	*65*	*290.99*	*147.06*
20	*2.49*	*0.89*	*70*	*319.86*	*166.73*
25	*8.91*	*1.56*	*75*	*430.69*	*197.95*
30	*9.65*	*2.28*	*80*	*594.30*	*253.48*
35	*25.67*	*3.17*	*85*	*771.51*	*243.56*
40	*43.91*	*6.13*	*90*	*1047.78*	*323.75*
45	*73.52*	*10.83*	*95*	*1274.53*	*331.08*
50	*112.09*	*21.04*	*100*	*1487.04*	*337.76*

Table 4.20 and 4.21 are shown the comparison which cover s cost and process time values respectively. The best results are shown in bold.

Table 4.20 Cost comparison of methods for increasing DAP instance sizes (cost value is column x 10^6) (Mahi et al., 2018).

Size	ACO (Tosun, 2014b)	RTS (Tosun, 2014b)	GA1 (Tosun, 2014b)	GA2 (Tosun, 2014b)	GA3 (Tosun, 2014b)	HG-MTS (Tosun, 2014b)	SA (Tosun et al., 2013b)	PSO-DAP		Greedy DAP (Proposed Method)
								Average	Standard Deviation	
5	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.02	0.0007	**0.003**
10	0.31	0.31	0.32	0.31	0.31	0.31	0.31	0.05	0.0130	**0.04**
15	0.98	0.98	0.99	0.98	0.98	0.98	0.98	0.41	0.0532	**0.14**
20	2.61	2.61	2.63	2.64	2.64	2.61	2.61	0.77	0,0816	**0.44**
25	5.19	5.19	5.25	5.26	5.24	5.19	5.15	3.74	1.2826	**0.60**
30	10.27	10.27	10.39	10.42	10.41	10.27	10.27	3.19	1.0470	**1.36**
35	16.39	16.39	16.64	16.61	16.66	16.39	16.41	9.04	4.5046	**1.96**
40	25.91	25.9	26.28	26.33	26.21	25.92	26.02	19.24	8.9307	**3.51**
45	37.28	37.26	37.73	37.8	37.82	37.27	37.40	27.04	16.7220	**5.54**
50	53.93	53.89	54.76	54.63	54.69	53.88	54.08	34.43	25.6230	**7.60**
55	71.30	71.19	72.72	72.40	72.13	71.21	71.40	51.38	37.4836	**34.91**
60	90.35	90.16	91.76	91.49	91.56	90.20	90.50	97.78	59.9261	**44.75**
65	112.31	112.13	113.59	113.75	113.84	112.08	112.49	125.01	91.0824	**68.88**
70	146.41	146.19	148.48	148.8	148.18	146.15	146.73	138.69	109.8225	**106.34**
75	177.90	177.7	180.04	180.75	180.63	177.65	178.16	171.47	139.8139	**163.89**
80	219.40	219.26	223.10	222.80	222.96	219.18	219.81	260.86	160.2445	**159.95**
85	262.24	261.88	267.04	266.15	266.19	261.99	262.89	260.63	240.2334	**196.66**
90	316.11	315.86	320.88	320.93	320.58	315.86	316.81	287.09	302.8202	**265.25**
95	370.14	369.92	375.49	375.85	375.29	369.91	371.14	365.06	392.3170	**353.27**
100	428.40	428.28	436.19	436.15	434.45	**427.98**	429.10	481.58	459.9464	**408.62**

Table 4.21 Execution time (s) comparison of methods for increasing DAP instance sizes (Mahi et al., 2018).

DAP Size	ACO (Tosun, 2014b)	RTS (Tosun, 2014b)	GA1 (Tosun, 2014b)	GA2 (Tosun, 2014b)	GA3 (Tosun, 2014b)	HG-MTS (Tosun, 2014b)	SA (Tosun et al., 2013b)	PSO-DAP	Greedy DAP (Proposed Method)
5	*9.26*	*0.83*	*76.27*	*56.11*	*88.11*	*1.44*	*130.29*	0.74	**0.004**
10	*14.52*	*2.73*	*87.80*	*60.37*	*94.91*	*2.45*	*143.84*	1.35	**0.01**
15	*13.74*	*5.66*	*90.76*	*66.22*	*104.13*	*2.65*	*214.30*	2.17	**0.02**
20	*17.91*	*8.89*	*123.79*	*84.13*	*167.22*	*4.17*	*243.30*	3.65	**0.08**
25	*25.86*	*14.52*	*131.98*	*81.96*	*125.30*	*5.21*	*351.23*	4.25	**0.34**
30	*31.17*	*20.89*	*132.46*	*104.64*	*137.02*	*7.38*	*461.89*	6.45	**5.05**
35	*43.31*	*29.06*	*150.06*	*111.87*	*151.02*	*10.73*	*393.73*	**6.81**	12.89
40	*56.59*	*37.05*	*166.80*	*128.75*	*173.21*	*15.60*	*420.65*	**8.85**	32.06
45	*80.92*	*48.67*	*191.93*	*159.10*	*202.10*	*20.80*	*437.74*	**8.42**	67.78
50	*105.33*	*62.74*	*471.98*	*207.56*	*359.57*	*26.80*	*511.40*	**9.60**	134.00
55	*126.00*	*76.07*	*268.31*	*201.43*	*261.71*	*27.22*	*516.86*	**13.74**	335.31
60	*166.55*	*91.79*	*315.31*	*208.37*	*290.46*	*39.56*	*828.14*	**16.09**	659.49
65	*204.35*	*109.20*	*421.93*	*284.08*	*336.01*	*48.92*	*1090.77*	**16.09**	1015.80
70	*320.62*	*131.54*	*536.15*	*344.20*	*358.03*	*63.13*	*1303.21*	**17.34**	1868.15
75	*309.51*	*155.31*	*609.77*	*379.07*	*380.81*	*73.41*	*976.97*	**17.01**	2712.19
80	*396.18*	*193.63*	*464.17*	*331.17*	*416.18*	*87.84*	*1234.48*	**16.29**	3755.91
85	*807.43*	*195.80*	*532.05*	*364.71*	*586.21*	*102.79*	*898.11*	**18.31**	5516.20
90	*621.55*	*215.58*	*563.15*	*400.37*	*531.13*	*123.19*	*1336.74*	**20.98**	7898.36
95	*725.93*	*250.72*	*629.55*	*974.24*	*569.92*	*143.16*	*1128.08*	**18.15**	11376.30
100	*1203.99*	*278.63*	*1236.30*	*568.73*	*808.82*	*179.07*	*1389.19*	**20.97**	15350.07

Proposed method difference from its alternatives has been evaluated through a sign test (Lurie et al., 2011; Mann, 2013). There is no outstanding difference between two methods as H0 hypothesis but H1 hypothesis demonstrate a significant difference between the two algorithms. All calculations have been done at the level of five percent significance. Table 4.22 contains sign test results and comparison results of each algorithm with the proposed algorithm. The statistical comparison of the proposed method and the other method is given in the last six rows of the Table 4.22. The description of the parameters ($S, X_S, \sigma_S, Z, \frac{Z_{0.05}}{2}, H_0, H_1$) is given in Table 2.1. A negative sign means a poor result when the comparisons of two columns have been compared, a positive sign means a good result. The H1 hypotheses were accepted because the computed Z values are outside the range in all tests ($|\text{Z Values}| > |\pm 1.96|$).

Table 4.22 *Statistical comparison of the methods using sign test.*

Size	Greedy DAP (Proposed Method)	PSO-DAP (Proposed Method)	Sign	ACO (Tosun, 2014a)	Sign	RTS (Tosun, 2014a)	Sign	GA1 (Tosun, 2014a)	Sign	GA2 (Tosun, 2014a)	Sign	GA3 (Tosun, 2014a)	Sign	HG-MTS (Tosun, 2014a)	Sign	SA (Tosun et al., 2013b)	Sign
5	**0.003**	0.02	-	0.04	-	0.04	-	0.04	-	0.04	-	0.04	-	0.04	-	0.04	-
10	**0.04**	0.05	-	0.31	-	0.31	-	0.32	-	0.31	-	0.31	-	0.31	-	0.31	-
15	**0.14**	0.41	-	0.98	-	0.98	-	0.99	-	0.98	-	0.98	-	0.98	-	0.98	-
20	**0.44**	0.77	-	2.61	-	2.61	-	2.63	-	2.64	-	2.64	-	2.61	-	2.61	-
25	**0.60**	3.74	-	5.19	-	5.19	-	5.25	-	5.26	-	5.24	-	5.19	-	5.15	-
30	**1.36**	**3.19**	-	10.27	-	10.27	-	10.39	-	10.42	-	10.41	-	10.27	-	10.27	-
35	**1.96**	9.04	-	16.39	-	16.39	-	16.64	-	16.61	-	16.66	-	16.39	-	16.41	-
40	**3.51**	19.24	-	25.91	-	25.9	-	26.28	-	26.33	-	26.21	-	25.92	-	26.02	-
45	**5.54**	27.04	-	37.28	-	37.26	-	37.73	-	37.8	-	37.82	-	37.27	-	37.40	-
50	**7.60**	34.43	-	53.93	-	53.89	-	54.76	-	54.63	-	54.69	-	53.88	-	54.08	-
55	**34.91**	51.38	-	71.30	-	71.19	-	72.72	-	72.40	-	72.13	-	71.21	-	71.40	-
60	**44.75**	97.78	-	90.35	-	90.16	-	91.76	-	91.49	-	91.56	-	90.20	-	90.50	-
65	**68.88**	125.01	-	112.31	-	112.13	-	113.59	-	113.75	-	113.84	-	112.08	-	112.49	-
70	**106.34**	138.69	-	146.41	-	146.19	-	148.48	-	148.8	-	148.18	-	146.15	-	146.73	-
75	**163.89**	171.47	-	177.90	-	177.7	-	180.04	-	180.75	-	180.63	-	177.65	-	178.16	-
80	**159.95**	260.86	-	219.40	-	219.26	-	223.10	-	222.80	-	222.96	-	219.18	-	219.81	-
85	**196.66**	260.63	-	262.24	-	261.88	-	267.04	-	266.15	-	266.19	-	261.99	-	262.89	-
90	**265.25**	287.09	-	316.11	-	315.86	-	320.88	-	320.93	-	320.58	-	315.86	-	316.81	-
95	**353.27**	365.06	-	370.14	-	369.92	-	375.49	-	375.85	-	375.29	-	369.91	-	371.14	-
100	**408.62**	481.58	-	428.40	-	428.28	-	436.19	-	436.15	-	434.45	-	**427.98**	-	429.10	-

Statistical Notations	Greedy DAP vs. ACO	Greedy DAP vs. ACO	Greedy DAP vs. RTS	Greedy DAP vs. GA1	Greedy DAP vs. GA2	Greedy DAP vs. GA3	Greedy DAP vs. HG-MTS	Greedy DAP vs. SA
S	0	0	0	0	0	0	0	0
X_s	10	10	10	10	10	10	10	10
σ_s	2.236	2.236	2.236	2.236	2.236	2.236	2.236	2.236
Z	-4.472	-4.472	-4.472	-4.472	-4.472	-4.472	-4.472	-4.472
$Z_{\frac{0.05}{2}}$	±1.96	±1.96	±1.96	±1.96	±1.96	±1.96	±1.96	±1.96
$H0$	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject
$\boldsymbol{H1}$	**Accept**	**Accept**	**Accept**	**Accept**	**Accept**	**Accept**	**Accept**	**Accept**

Results have been shown that PSO-DAP is less costly and time-consuming than other methods such as ACO, RTS, GA HG MTS and etc. DAP samples are produced randomly and the PSO-DAP is used to solve them in this thesis. Researchers can be used these samples for future works. Datasets in the proposed method are generated randomly based on the formula described in section 2.2 due to the absence of our compared algorithms datasets. Among the 20 existing results, 16 of them are the best and close to our compared algorithms results in Table 4.20. As shown in Table 4.22, the proposed method is statistically different from other methods. So in terms of time-consuming, PSO-DAP consumes less time in comparison with other algorithms in Table 4.21. Greedy DAP method is given the best result values in all three states in terms of cost but time parameter of it is higher than other methods.

5. CONCLUSION

In this thesis, two important optimization problems are discussed. The first problem was TSP solved by a hybrid method. In the proposed method, the parameters of the ACO algorithm was optimized with PSO. The 3-Opt algorithm was then used to avoid local minimums. The second problem was to solve the DAP by the PSO algorithm, which was not previously applied in the literature. In addition, a greedy method for DAP was proposed. Further explanation, a hybrid method of PSO, ACO and 3-Opt algorithms have been proposed to solve the TSP's. In the proposed method, we use the PSO algorithm to determine the parameters of the α and β algorithms in order to obtain a better result of the ACO algorithm and also to solve the intersections of the edges created in the tour using the 3-Opt algorithm. To show the performance of the proposed method for ten standard data sets on the TSPLIB, we perform operations average route length, standard deviation and percentage relative error values. By analysing the number of ant 10, 20 and 30, the number of ant effects on the results of the proposed method has been investigated. Obtained results are shown for the number of ants for the proposed method, with fewer ants, better performance is seen. Considering the comparison of the proposed method with the previous and similar methods, we have obtained better results, which indicates better performance of the proposed method. The basis of this thesis is based on the allocation of non-repeated data in distributed database systems. Reduced query execution time and transaction costs are targeted at DAP. Population-based exploratory algorithms are often used to accomplish this goal. In this thesis, we proposed a PSO-based PSO-DAP method to minimize query runtime and transaction costs. The PSO-DAP performance is reviewed on 20 different DAPs and the results are compared with the results of available methods, with regard to the time of execution of the query and the transaction cost. Experimental results show that PSO-DAP is better than other comparable methods in intervals of solution quality and runtime in almost all instances. When the dimensions of the problem increase, the function of the method decreases because space symbolically grows. But when the results are explored, the results of the proposed method prove their superiority to the results of previous methods. Because the proposed method has less computation.

Non-replicated distributed database systems are targeted in this thesis. DAP is evaluated to decrease the query runtime and transaction cost so population heuristic algorithms are used to achieve this purpose. In this thesis, a new method based on the Greedy algorithm, Greedy DAP in order to alleviate the query runtime and transaction cost has been presented. To evaluate the proposed algorithm, three states have been considered. The first state refers to the fragment fix size and site size increment. In the second state, the size of sites increase and the size of fragments be fixed. The third state contains the equal size of site and fragments. Obtained results of the proposed algorithm based on these three states demonstrate that the presented method has given a good performance. Greedy DAP is founded on various DAP samples in several states and obtained results in comparison with other algorithms has been evaluated in terms of query execution time and transaction cost. Obtained results have been shown Greedy DAP in the duration of solution quality and runtime has better performance. Due to solution space exponentially growing along with increasing the dimensionality of the problem, the performance of the method is decreased. However, but during results analyse, the presented method generates comparable results with the state of art algorithms especially in terms of execution time due to its lower number of computation. The firefly algorithm can be used to solve DAP in the future because it works better for larger problems.

REFERENCES

Abdalla, H. I., 2012, A New Data Re-Allocation Model for Distributed Database Systems, *International Journal of Database Theory and Application*, 5 (2), 45-60.

Adl, R. K. and Rankoohi, S. M. T. R., 2009, A new ant colony optimization based algorithm for data allocation problem in distributed databases, *Knowledge and Information Systems*, 20 (3), 349-373.

Al-Sanhani, A. H., Hamdan, A., Al-Thaher, A. B. and Al-Dahoud, A., 2017, A comparative analysis of data fragmentation in distributed database, *Information Technology (ICIT), 2017 8th International Conference on*, 724-729.

Amer, A. A. and Abdalla, H. I., 2012, A heuristic approach to re-allocate data fragments in DDBSs, *Information Technology and e-Services (ICITeS), 2012 International Conference on*, 1-6.

Astrachan, O. L., Duvall, R. C., Forbes, J. and Rodger, S. H., 2002, Active learning in small to large courses, *Frontiers in Education, 2002. FIE 2002. 32nd Annual*, T2A-T2A.

Bai, Q., 2010, Analysis of particle swarm optimization algorithm, *Computer and information science*, 3 (1), 180-184.

Barbalios, N. and Tzionas, P., 2014, A robust approach for multi-agent natural resource allocation based on stochastic optimization algorithms, *Applied Soft Computing*, 18, 12-24.

Bontoux, B. and Feillet, D., 2008, Ant colony optimization for the traveling purchaser problem, *Computers & Operations Research*, 35 (2), 628-637.

Bouchaud, J.-P., 2018, Greedy algorithms and Zipf laws.

Chen, S.-M. and Chien, C.-Y., 2011a, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, *Expert Systems with Applications*, 38 (12), 14439-14450.

Chen, S. M. and Chien, C. Y., 2011b, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, *Expert Systems with Applications*, 38 (12), 14439-14450.

Colorni, A., Dorigo, M. and Maniezzo, V., 1991, Distributed optimization by ant colonies, *Proceedings of the first European conference on artificial life*, 134-142.

de Freitas, J. C. and Penna, P. H. V., 2018, A Randomized Variable Neighborhood Descent Heuristic to Solve the Flying Sidekick Traveling Salesman Problem, *Electronic Notes in Discrete Mathematics*, 66, 95-102.

Deng, W., Chen, R., Gao, J., Song, Y. J. and Xu, J. J., 2012, A novel parallel hybrid intelligence optimization algorithm for a function approximation problem, *Computers & Mathematics with Applications*, 63 (1), 325-336.

Dong, G., Guo, W. W. and Tickle, K., 2012a, Solving the traveling salesman problem using cooperative genetic ant systems, *Expert Systems with Applications*, 39 (5), 5006-5011.

Dong, G. F., Guo, W. W. and Tickle, K., 2012b, Solving the traveling salesman problem using cooperative genetic ant systems, *Expert Systems with Applications*, 39 (5), 5006-5011.

Dorigo, M. and Gambardella, L. M., 1997, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Transactions on evolutionary computation*, 1 (1), 53-66.

Dorigo, M. and Stutzle, T., 2004, Ant Colony Optimization, *Massachusetts Institute of Technology*.

Dorigo, M. and Stützle, T., 2009, Ant colony optimization: overview and recent advances, *Techreport, IRIDIA, Universite Libre de Bruxelles*, 8.

Du, J., Li, R., Xiao, Z., Tong, Z. and Zhang, L., 2017, Optimization of data allocation on CMP embedded system with data migration, *International Journal of Parallel Programming*, 45 (4), 965-981.

Eberhart, R. and Kennedy, J., 1995, Particle swarm optimization, proceeding of IEEE International Conference on Neural Network, *Perth, Australia*, 1942-1948.

Ferreira, C., 2006, Gene expression programming: mathematical modeling by an artificial intelligence, Springer.

Gao, W., Friedrich, T., Neumann, F. and Hercher, C., 2018, Randomized Greedy Algorithms for Covering Problems.

Geng, X., Chen, Z., Yang, W., Shi, D. and Zhao, K., 2011, Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search, *Applied Soft Computing*, 11 (4), 3680-3689.

Goldberg, D. E., 1989, Genetic Alogorithms in Search, *Optimization & Machine Learning*.

Grefenstette, J., Gopal, R., Rosmaita, B. and Van Gucht, D., 1985, Genetic algorithms for the traveling salesman problem, *Proceedings of the first International Conference on Genetic Algorithms and their Applications*, 160-168.

Gu, X., Lin, W. J. and Veeravalli, B., 2006, Practically realizable efficient data allocation and replication strategies for distributed databases with buffer constraints, *Ieee Transactions on Parallel and Distributed Systems*, 17 (9), 1001-1013.

Gülcü, Ş., Mahi, M., Baykan, Ö. K. and Kodaz, H., 2018, A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem, *Soft Computing*, 22 (5), 1669-1685.

Gündüz, M., Kıran, M. S. and Özceylan, E., 2014, A hierarchic approach based on swarm intelligence to solve traveling salesman problem. Turkish Journal of Electrical Engineering & Computer Sciences.

Gündüz, M., Kiran, M. S. and Özceylan, E., 2015, A hierarchic approach based on swarm intelligence to solve the traveling salesman problem, *Turkish Journal of Electrical Engineering & Computer Sciences*, 23 (1), 103-117.

Guo, Z., Chen, R. L.-Y., Fan, N. and Watson, J.-P., 2017, Contingency-constrained unit commitment with intervening time for system adjustments, *IEEE Transactions on Power Systems*, 32 (4), 3049-3059.

Jolai, F. and Ghanbari, A., 2010, Integrating data transformation techniques with Hopfield neural networks for solving travelling salesman problem, *Expert Systems with Applications*, 37 (7), 5331-5335.

Jun-man, K. and Yi, Z., 2012, Application of an Improved Ant Colony Optimization on Generalized Traveling Salesman Problem, *Energy Procedia*, 17, 319-325.

Junqiang, W. and Aijia, O., 2012, A hybrid algorithm of ACO and delete-cross method for TSP, *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on*, 1694-1696.

Karaboga, D. and Basturk, B., 2007, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of global optimization*, 39 (3), 459-471.

Karaboga, D. and Gorkemli, B., 2011, A combinatorial artificial bee colony algorithm for traveling salesman problem, *Innovations in intelligent systems and applications (inista), 2011 international symposium on*, 50-53.

Kennedy, J. and Eberhart, R., 1995, Particle swarm optimization, *1995 Ieee International Conference on Neural Networks Proceedings, Vols 1-6*, 1942-1948.

Khan, I. and Maiti, M. K., 2018, A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem, *Swarm and Evolutionary Computation*.

Khan, I., Pal, S. and Maiti, M. K., 2018, A modified particle swarm optimization algorithm for solving traveling salesman problem with imprecise cost matrix, *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*, 1-8.

Kiran, M. S. and Gunduz, M., 2013, A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems, *Applied Soft Computing*, 13 (4), 2188-2203.

Krohling, R. A. and dos Santos Coelho, L., 2006, Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36 (6), 1407-1416.

Li, S. P. and Wong, M. H., 2013, Data Allocation in Scalable Distributed Database Systems Based on Time Series Forecasting, *2013 Ieee International Congress on Big Data*, 17-24.

Lin, C.-J., Chen, C.-H. and Lin, C.-T., 2009, A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39 (1), 55-68.

López-Ibáñez, M. and Blum, C., 2010, Beam-ACO for the travelling salesman problem with time windows, *Computers & Operations Research*, 37 (9), 1570-1583.

Lurie, D., Abramson, L. R. and Vail, J. A., 2011, Applying statistics, Citeseer.

Mahi, M., Baykan, O. K. and Kodaz, H., 2015, A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem, *Applied Soft Computing*, 30, 484-490.

Mahi, M., Baykan, O. K. and Kodaz, H., 2018, A new approach based on particle swarm optimization algorithm for solving data allocation problem, *Applied Soft Computing*, 62, 571-578.

Mamaghani, A. S., Mahi, M., Meybodi, M. R. and Moghaddam, M. H., 2010, A novel evolutionary algorithm for solving static data allocation problem in distributed database systems, *Network Applications Protocols and Services (NETAPPS), 2010 Second International Conference on*, 14-19.

Mann, P. S., 2013, Introductory Statistics, 8th Edition, Wiley.

Mashwani, W. K. and Salhi, A., 2012, A decomposition-based hybrid multiobjective evolutionary algorithm with dynamic resource allocation, *Applied Soft Computing*, 12 (9), 2765-2780.

Masutti, T. A. and de Castro, L. N., 2009a, A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, *Information Sciences*, 179 (10), 1454-1468.

Masutti, T. A. S. and de Castro, L. N., 2009b, A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, *Information Sciences*, 179 (10), 1454-1468.

Mavrovouniotis, M. and Yang, S., 2013, Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors, *Applied Soft Computing*, 13 (10), 4023-4037.

Mayne, S. R. and Satav, S., 2017, Survey on Cloud Infrastructure Resource Allocation for Big Data Applications, *International Journal of Engineering Science*, 4008.

Mirjalili, S., Mirjalili, S. M. and Lewis, A., 2014, Grey wolf optimizer, *Advances in engineering software*, 69, 46-61.

Neri, F., Cotta, C. and Moscato, P., 2012, Handbook of memetic algorithms, Springer.

Niamir, A., Skidmore, A. K., Muñoz, A.-r., Toxopeus, A. G. and Real, R., 2018, Incorporating knowledge uncertainty into species distribution modelling, *Biodiversity and conservation*, 1-18.

Osaba, E., Yang, X.-S., Diaz, F., Lopez-Garcia, P. and Carballedo, R., 2016, An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems, *Engineering Applications of Artificial Intelligence*, 48, 59-71.

Othman, Z. A., Srour, A. I., Hamdan, A. R. and Ling, P. Y., 2013, Performance Water Flow-Like Algorithm for TSP by improving its Local Search, *International Journal of Advancements in Computing Technology*, 5 (14).

Pasti, R. and De Castro, L. N., 2006, A neuro-immune network for solving the traveling salesman problem, *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, 3760-3766.

Pedro, O., Saldanha, R. and Camargo, R., 2013, A tabu search approach for the prize collecting traveling salesman problem, *Electronic Notes in Discrete Mathematics*, 41, 261-268.

Peker, M., Sen, B. and Kumru, P. Y., 2013a, An efficient solving of the traveling salesman problem: the ant colony system having parameters optimized by the Taguchi method, *Turkish Journal of Electrical Engineering and Computer Sciences*, 21, 2015-2036.

Peker, M., ŞEN, B. and Kumru, P. Y., 2013b, An efficient solving of the traveling salesman problem: the ant colony system having parameters optimized by the Taguchi method, *Turkish Journal of Electrical Engineering & Computer Sciences*, 21 (Sup. 1), 2015-2036.

Pham, T. S., Leyman, P. and De Causmaecker, P., 2018, The intermittent travelling salesman problem, *International Transactions in Operational Research*.

Reinelt, G., 1991, TSPLIB—A traveling salesman problem library, *ORSA journal on computing*, 3 (4), 376-384.

Rodríguez Vásquez, W. C., 2016, Sistema Experto Hibrido para Optimizar los Procesos de Programación, Zonificación y Diseño de Rutas de un Servicio de Mensajería Considerando Restricciones de Ventanas de Tiempo, Múltiples Periodos de Tiempo, Fechas de Vencimiento, Distancia y Horario en la Capacidad.

Sanchez, M., Saxena, M. and Sabhikhi, A., 2018, Hybrid data architecture for use within a healthcare industry optimized cognitive environment, Google Patents.

Sen, G., Krishnamoorthy, M., Rangaraj, N. and Narayanan, V., 2016, Mathematical Models and Empirical Analysis of a Simulated Annealing Approach for Two Variants of the Static Data Segment Allocation Problem, *Networks*, 68 (1), 4-22.

Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C. and Wang, Q., 2007, Particle swarm optimization-based algorithms for TSP and generalized TSP, *Information processing letters*, 103 (5), 169-176.

Singh, A., Kahlon, K. S. and Virk, R. S., 2014, Replicated Static Allocation of Fragments in Distributed Database Design using Biogeography-based Optimization, *Proc. of Int. Conf. on Advances in Communication, Network, and Computing, CNC*, 462-472.

Sivakumar, T. and Basheer, P., 2017, A Survey on Data Leakage Detection and De-Duplication in Data Mining System, *Journal of Network Communications and Emerging Technologies (JNCET) www. jncet. org*, 7 (10).

Taillard, É. D. and Helsgaun, K., 2018, POPMUSIC for the Travelling Salesman Problem, *European Journal of Operational Research*.

Taillard, É. D. and Helsgaun, K., 2019, POPMUSIC for the travelling salesman problem, *European Journal of Operational Research*, 272 (2), 420-429.

Tosun, U., Dokeroglu, T. and Cosar, A., 2013a, A robust Island Parallel Genetic Algorithm for the Quadratic Assignment Problem, *International Journal of Production Research*, 51 (14), 4117-4133.

Tosun, U., Dokeroglu, T. and Cosar, A., 2013b, Heuristic algorithms for fragment allocation in a distributed database system, In: Computer and Information Sciences III, Eds: Springer.

Tosun, U., 2014a, A new recombination operator for the genetic algorithm solution of the quadratic assignment problem, *Procedia Computer Science*, 32, 29-36.

Tosun, U., 2014b, Distributed Database Design using Evolutionary Algorithms, *Journal of Communications and Networks*, 16 (4), 430-435.

Tsai, C. F., Tsai, C. W. and Tseng, C. C., 2004, A new hybrid heuristic approach for solving large traveling salesman problem, *Information Sciences*, 166 (1-4), 67-81.

Ulus, T. and Uysal, M., 2003, Heuristic approach to dynamic data allocation in distributed database systems, *Pakistan Journal of Information and Technology*, 2 (3), 231-239.

Wang, M., Feng, S., Ouyang, C. and Li, Z., 2015, RFID tag oriented data allocation method using artificial immune network, *The 27th Chinese Control and Decision Conference (2015 CCDC)*, 5218-5223.

Zhong, Y., Lin, J., Wang, L. and Zhang, H., 2018, Discrete comprehensive learning particle swarm optimization algorithm with Metropolis acceptance criterion for traveling salesman problem, *Swarm and Evolutionary Computation*.

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname	:	Mostafa MAHI
Nationality	:	IRAN
Birth Place and Date	:	Ajabshir Iran, 07-01-1980
Telephone	:	+90 535 037 3691, +98 914 176 6257
Fax	:	+98 41 37275100
e-mail	:	mahi@selcuk.edu.tr, mahi@pnu.ac.ir, mostafamahi@gmail.com

EDUCATION

Degree		**Name, Province, Country**	**Year of Passing**
High School	:	RAZI college, AJABSHIR, Iran	1998
Bachelors	:	SHABESTAR Islamic Azad University, Iran	2003
Master	:	South Tehran Islamic Azad University ,Iran	2006
Doctorate	:	SELCUK University, Konya, Turkey	2018

WORK EXPERIENS

Year	**Institution**	**Post**
2007-2009	Department of Computer Engineering and Information Technology, Islamic Azad University, BONAB, Iran	Lecture
2009-Till Now	Department of Computer Engineering and Information Technology, Payame Noor University, MARAGHEH, Iran	Lecture

PROFESSION

Software Engineering

FOREIGN LANGUAGES

English, Turkish, Persian, Azari

OTHERS

Awarded Ph.D. scholarship from the university

PUBLICATIONS

Before Thesis

1. Mamaghani, A. S., Mahi, M., Meybodi, M. R. and Moghaddam, M. H., 2010, A novel evolutionary algorithm for solving static data allocation problem in distributed database systems, Network Applications Protocols and Services (NETAPPS), 2010 Second International Conference on, 14-19

2. Mamaghani, A. S., Mahi, M., 2010, A learning automaton based method for data fragments allocation in distributed database systems, 2010 10th IEEE International Conference on Computer and Information Technology, 8-12

3. HS. Mend, M. Mahi, Peer Ranking in P2P Integration System on Inconsistent Data Sources, MAGNT Research Report (ISSN. 1444-8939) Vol.2 (4). PP: 511-515

4. M.Mahi, A.Amiri, Adaptiand Video Background Estimation and Moving Object Detection Using QXVSS LMS Algorithm, 2011 3rd International Conference on Computer and Automation Engineering (ICCAE 2011)

In Thesis

1. Mostafa Mahi, Halife Kodaz, Genetic Algorithm with Descendants Idea for cheduling Tasks Graph in the Multi-Processor Architecture, Journal of Advances in Computer Networks, Vol. 2, No. 1, March 2014

2. Mostafa Mahi, Ömer Kaan Baykan, Halife Kodaz (2015) A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Travelling Salesman Problem. Appl Soft Comput 30:484–490

3. Saban Gülcü¸ Mostafa Mahi, Ömer Kaan Baykan, Halife Kodaz, A parallel cooperatiand hybrid method based on ant colony optimization and 3-Opt algorithm for solving Travelling salesman problem, Published online: 17 Noandmber 2016, Soft Comput (2018) 22:1669–1685

4. Mostafa Mahi, Ömer Kaan Baykan, Halife Kodaz, A new method based on particle swarm optimization algorithm for solving data allocation problem, Applied Soft Computing 62 (2018) 571–578