MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

METAHEURISTIC METHODS FOR THE OBSTACLE NEUTRALIZATION PROBLEM

RAMAZAN ALGIN

MASTER THESIS

Computer Engineering Department

ADVISOR

Assoc. Prof. Ali Fuat ALKAYA

CO-ADVISOR

Assoc. Prof. Vural AKSAKALLI

ISTANBUL, 2016

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

METAHEURISTIC METHODS FOR THE OBSTACLE NEUTRALIZATION PROBLEM

RAMAZAN ALGIN

524113007

MASTER THESIS

Computer Engineering Department

ADVISOR

Assoc. Prof. Ali Fuat ALKAYA

CO-ADVISOR

Assoc. Prof. Vural AKSAKALLI

ISTANBUL, 2016

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

Ramazan ALGIN, a Master of Science student of Marmara University Institute for Graduate Studies in Pure and Applied Sciences, defended his thesis entitled **"Metaheuristic Methods for the Obstacle Neutralization Problem"**, on June 1, 2016 and has been found to be satisfactory by the jury members.

Jury Members

Assoc. Prof. Dr. Ali Fuat ALKAYA (Advisor)
Marmara University .....(SIGN) .....

Assoc.Prof. Murat ŞENSOY (Jury Member)
Ozyegin University .....(SIGN) .....

Assist.Prof. Ali Haydar ÖZER (Jury Member)
Marmara University .....(SIGN) .....

APPROVAL

Marmara University Institute for Graduate Studies in Pure and Applied Sciences Executive Committee approves that Ramazan ALGIN be granted the degree of Master of Science in Department of Computer Engineering on June 13, 2016. (Resolution no:2016/13-02).

Director of the Institute
Prof. Dr. Uğur YAHŞİ

ACKNOWLEDGMENT

Firstly, I want to thank my advisors Assoc. Prof. Ali Fuat ALKAYA and Assoc. Prof. Vural AKSAKALLI for their guidance, patience and support in the development of thesis application and in the completion of this thesis.

Besides my advisors, I would like to thank my father, my mother and my brother for their continuous support and understanding.

I would additionally like to thank all my friends for their understanding.

This research was supported by The Scientific and Technological Research Council of Turkey (TUBITAK), Grant 114M069.

June, 2016 **Ramazan ALGIN**

TABLE OF CONTENTS

ÖZET

ETKİSİZLEŞTİRME PROBLEMİ İÇİN METASEZGİSEL ÇÖZÜM METOTLARININ GELİŞTİRİLMESİ VE UYGULANMASI

Bu tez çalışmasında bir askeri deniz senaryosu olarak gerçek hayatta karşılığı olan ve gerçek bir problemden ilham alınarak tanımlanmış Etkisizleştirme Problemi (EP) üzerinde teorik ve deneysel çalışmalar yapılarak yeni çözüm yöntemleri geliştirilmiştir. EP'de bir ajan bulunmakta ve problemin amacı bu ajanın belirli bir başlangıç noktasından hedef noktasına, disk şeklinde engellerin (mayın) bulunduğu bir ortamda, en hızlı ve en güvenli şekilde ulaşmasını sağlamaktır. Ajan belli bir masraf karşılığında verilen etkisizleştirme sayısını aşmamak koşuluyla engelleri etkisizleştirme kabiliyetine sahiptir.

Bu tez çalışmasında EP'yi çözmek için çeşitli metasezgilseller kullanılmıştır. Karınca sistemi algoritması, karınca kolonisi algoritması, göç eden kuşlar algoritması, genetik algoritma ve benzetimli tavlama metasezgiselleri EP için geliştirilip uygulanmıştır. Metasezgisellerin yanında EP, ZIMPL dili kullanılarak modellenmiştir ve bu model SCIP çözücüsü kullanılarak kesin çözüm elde edilmiştir.

Yapılan deneyler hem gerçek hem de rassal örnekler üzerinde gerçekleştirilmiştir. Geliştirilen metasezgisellerin bulduğu çözümler SCIP çözücüsünün sonuçlarıyla küçük ve orta boyutlu problem örnekleri üzerinde kıyaslamalı bir şekilde verilmiştir. Yapılan çalışmalar sonucunda, geliştirdiğimiz metasezgisellerin uygun zaman içerisinde en iyiye yakın sonuçlar elde edildiği görülmüştür. Kesin çözüm algoritması büyük boyutlu örnekler için uygun olmadığı için bu tip örneklerde metasezgisellerin kullanılması uygundur.

ABSTRACT

METAHEURISTIC METHODS FOR THE OBSTACLE NEUTRALIZATION PROBLEM

Within the scope of this thesis both theoretical and computational research have been conducted for developing new solution methods for Obstacle Neutralization Problem (ONP) which is actually a military navy scenario and it is inspired from a real problem. In this thesis, we develop metaheuristics for the ONP which is a path planning problem where the aim is to safely and swiftly traverse an agent from a given start point to a target point through a plan of potential mine or threat discs in the plane. A neutralization capability is given to the agent. He can neutralize the threats without exceeding the given neutralization limit. When the agent neutralizes a threat, neutralization cost of this threat is added to the total cost.

To solve the ONP, ant system, ant colony system, migrating birds optimization, genetic algorithm and simulated annealing algorithms are developed and customized. In addition to these metaheuristics SCIP solver is used to find exact solution. ONP is modeled with ZIMPL language to become ready for SCIP solver.

We provide computational experiments both on real-world and synthetic data to empirically assess their performance. The results of the metaheuristics are compared with exact solutions on small and moderate instances. The comparison results present that our algorithms find near-optimal solutions in reasonable execution times. For larger instances SCIP is not applicable because of high run time complexity, therefore, metaheuristics developed for ONP are suitable for larger instances.

LIST OF SYMBOLS

s: start point

t: terminal point

$\mathcal{A}$*:* finite set of open discs in $\mathbb{R}^2$

c: cost function

K: given neutralization number

α^*: largest penalty cost

C: current neutralization number

G: graph

V: set of vertices

E: set of edges

τ: pheromone level

τ_0: initial pheromone level

$\delta(x, y)$: cost of edge that is between vertex x and vertex y

β: parameter which determines the relative importance of pheromone versus distance

n: iteration number

J: set that keeps possible vertices

α: pheromone decay parameter between 0 and 1

ρ: is pheromone decay parameter in ACS

m: number of ants

L_k: path length performed by k-th ant

r: initial solution

T: temperature parameter

f(r): cost of solution r

ω: weight function

$\lambda(x)$*:* the shortest path from vertex *x* to *t*

$\eta(\lambda(x))$*:* weight of shortest path from vertex *x* to *t*

$\mu(a_k)$: number of remaining neutralizations for k-th ant

$\vartheta(x,y)$: weight of edge which is between vertex *x* and vertex *y*

nv: next vertex

S: solution for ONP

p_S: shortest path

bs: best solution

cs: current solution

R: number of iterations at each temperature setting

a: temperature decrease ratio

b: increase ratio in iteration number at each setting

nog: predefined number of iterations

nos: number of random initial solutions

ps: population set

xp: crossover probability

mp: mutation probability

noit: predefined number of iterations

nob: solutions (birds) number

not: tour number

non: neighbor solutions number to be generated from a solution

olf: solution number to be shared with the following solution (bird)

δ(p): cost of path

q: random numbers between 0 and 1

q_0 : random numbers between 0 and 1

LIST OF ABBREVIATIONS

ONP: Obstacle Neutralization Problem

AS: Ant System

ACS: Ant Colony System

GA: Genetic Algorithm

SA: Simulated Annealing

MBO: Migrating Birds Optimization

PSA: Penalty Search Algorithm

DCLC: Delay Constrained Least Cost

DCUR: Delay-Constrained Unicast Routing

DCCR: Delay Cost Constrained Routing

TSP: Travelling Salesman Problem

STR: State Transition Rule

PPP: Post Processing Procedure

RDP: Random Disambiguation Paths

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

One of the important topics in operations research and mathematics is finding the shortest path under certain constraints. This topic is mostly referred to as path planning or constrained shortest path problem. The time for data to reach target considering a given total delay limit minimization problem in the telecommunications industry (Kuipers et al., 2004), for a military aircraft finding a path with lower likelihood of being noticed by enemy radar considering fuel constraints (Zabarankin et al., 2002), and the curve approximation problem with maximum breakpoints number considering the constraint related with storage in computer graphics (Dahl & Realfsen, 1997) are some of the constrained shortest path problems observed in real life. These problems are mostly NP-Hard problems for which applying exact solution methods is not reasonable on moderate or large instances. In this case, heuristic algorithms are usually preferred. In the literature, there is also a class of heuristic algorithms which can be used to solve a large class of problems either directly or with minor modifications hence getting the name metaheuristics.

The metaheuristics often generate good solutions in reasonable times. So far many metaheuristics are proposed by researchers. Genetic algorithms, simulated annealing, tabu search, ant system are some of the widely used metaheuristics in the literature (Holland, 1975; Kirkpatrick et al., 1983; Glover, 1986; Dorigo, 1992). On the other hand, artificial bee colony, migrating birds optimization, differential evolution are examples of other competitive metaheuristics proposed recently (Karaboga & Basturk, 2007; Duman et al., 2012; Storn & Price, 1997). As their names imply, the metaheuristics are mostly nature inspired. In the literature, there are some studies that apply metaheuristics to solve the path planning algorithms (Latourell et al., 1998; Lee, 1995; Royset et al., 2009).

In this thesis, we tackle a path planning problem and design tailor-made metaheuristics for solving the problem. Specifically, the undertaken problem is called obstacle neutralization problem (ONP) wherein an agent needs to safely and swiftly navigate from a given source location to a destination through an arrangement of disc-shaped obstacles in the plane. In the ONP, the agent has a neutralization capability which is limited by given a constant number. After neutralizing an obstacle, agent can enter this area and the cost of neutralization is added to its traversal length. But the neutralization capability of the agent is limited, say by K, due to payload capacity of the agent. ONP is closely related to the problems undertaken in real world applications in diverse fields such as telecommunications routing (Kuipers et al., 2004), curve approximations (Dahl & Realfsen, 1997), scheduling and minimum-risk routing of military

vehicles and aircraft (Zabarankin et al., 2002). Therefore, the techniques developed in this study may also be applied to other application domains.

Mathematically, ONP instance is a tuple *(s, t, $\mathcal{A}$, c, K)*, where *s* is start point and *t* is terminal point in $\mathbb{R}^2$, $\mathcal{A}$ is a finite set of open discs in $\mathbb{R}^2$, *c* is a cost function $\mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$, and *K* is a given constant in $\mathbb{R}_{\geq 0}$. In this problem we have an agent that wants to go from *s* to *t*. This agent cannot enter the discs unless he has an option to neutralize discs that can be considered as obstacle or threat like mine. The agent has neutralization capability that is limited with *K* number of discs where $K \leq |\mathcal{A}|$. When a disc is neutralized, its neutralization cost is added to the traversal length of the path. In this problem our aim is taking the agent from *s* to *t* safely using the shortest path.

There is a simple example for ONP in Figure 1. In this figure, each disc has radius of 5 and neutralization cost of 1. As seen in the figure, when the agent has *K*=0 neutralization, he chooses red (solid) path. Similarly, when *K* = 1, 2, 3 green (dotted), blue (dashed), and black (bold solid) paths are our optimum paths.

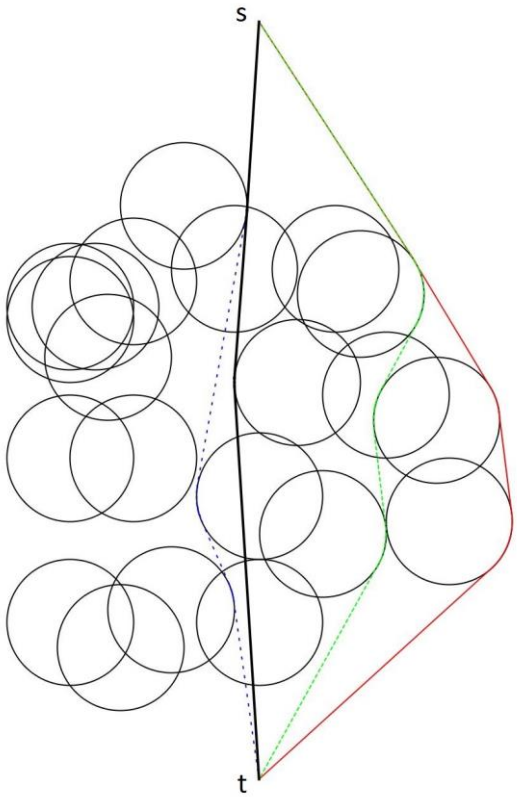

Figure 1 – An example to the obstacle neutralization problem and optimal paths for k=0, 1, 2 and 3.

The integer programming formulation of the ONP is given in Equations 1-5. In this formulation, x_{ij} is binary variable and if the edge (*i*, *j*) (red line in Figure 2) belongs to the optimal path then x_{ij} =1, otherwise 0.

$$min \sum_{i=1}^{|V|} \sum_{\substack{j=1 \\ j \neq i}}^{|V|} c_{ij} x_{ij} \quad (1)$$

s.t.

$$\sum_{\substack{j=1 \\ j \neq s}}^{|V|} x_{sj} - \sum_{\substack{i=1 \\ i \neq s}}^{|V|} x_{is} = 1 \quad (2)$$

$$\sum_{\substack{i=1 \\ i \neq t}}^{|V|} x_{it} - \sum_{\substack{j=1 \\ j \neq t}}^{|V|} x_{tj} = 1 \quad (3)$$

$$\sum_{\substack{j=1 \\ j \neq k}}^{|V|} x_{kj} - \sum_{\substack{i=1 \\ i \neq k}}^{|V|} x_{ik} = 0 \ \forall k \in V - \{s, t\} \quad (4)$$

$$\sum_{i=1}^{|V|} \sum_{\substack{j=1 \\ j \neq i}}^{|V|} w_{ij} x_{ij} \leq K \quad (5)$$

$$x_{ij} \in \{0,1\} \ \forall i, j \in \{1, \dots, |V|\}$$

Eq. (2) states that the difference in the number of edges leaving *s* and entering into *s* is 1. Eq. (3) shows that the difference in the number of edges entering into *t* and leaving *t* is 1. Eq. (4) states that for all other nodes except *s* and *t*, the number of edges entering them is equal to the number of edges leaving them. Eq. (5) enforces the total weight limit.

To understand integer formulation better, see Figure 2. In this figure, each green point is a vertex and the line between two green points is called an edge. The edges are bidirectional; therefore, every vertex has eight outgoing edges and eight incoming edges. Assume that the neutralization cost of the disc is 1 and the length of edge is 1 (for diagonal edges it is $\sqrt{2}$). Half of the neutralization cost (0.5) is added to the edges intersecting with this disc. Then the cost of the edge becomes 1.5 ($\sqrt{2} + 0.5$ for diagonal edge) if it intersects with one disc, otherwise its cost is equal to its length. For this example, the cost of the path is equal to 14.

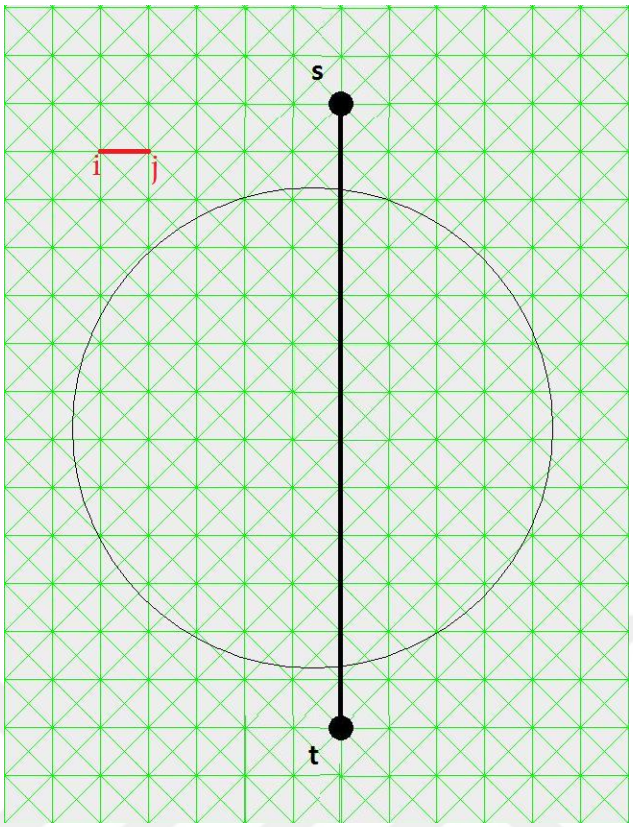

Figure 2 – ONP example on the grid graph

To our knowledge, the ONP is studied in (Alkaya et al., 2014; Alkaya & Oz, 2016; Algin et al., 2013). In (Alkaya et al., 2014) the authors develop a heuristic for solving the ONP. On the other hand, in (Alkaya & Oz, 2016) the authors develop an efficient exact method for solving small and moderate sized graphs. However, both of the proposed algorithms are based on the assumption that every disc has the same radius and same neutralization cost. In this thesis, that constraint is released and we provide solution methods that can be applicable for more realistic scenarios. In (Algin et al., 2013) the authors develop an ant system algorithm for the ONP. However, in their study they just present the algorithm and the results of the computational experiments without any other metaheuristic comparison.

In this thesis, our contribution is three-fold: (1) we present metaheuristic algorithms that can solve ONP instances having various radii and neutralization cost, (2) our GA, SA and MBO algorithms designed for ONP outperform AS and ACS, AS was developed for ONP in a former study, (3) we show that the proposed metaheuristics present very close results on small and moderate sized graphs and therefore can be used on large graphs.

The contributions of this thesis to the literature are as follows:

- Alkaya, A. F., & Algin, R. (2015). Metaheuristic based solution approaches for the obstacle neutralization problem. Expert Systems with Applications, 42(3), 1094-1105. (SCI Expanded)
- Algin, R. & Alkaya, A. F. (2015). Solving the Obstacle Neutralization Problem Using Swarm Intelligence Algorithms. The Seventh International Conference on Soft Computing and Pattern Recognition (SoCPaR 2015). Fukuoka, Japan.
- Algin, R., Alkaya, A.F., & Aksakalli, V. (2015). Performance Comparison of Metaheuristics for the Obstacle Neutralization Problem. 13th Cologne-Twente Workshop on Graphs & Combinatorial Optimization (CTW 2015). Istanbul, Turkey.

This thesis is organized as follows. In Section 2, literature survey on ONP and related problems are given and the metaheuristics used in this study are explained. Section 3 explains how these metaheuristics are customized for demonstrating their best performance on the ONP. Section 4 reports the results of extensive computational experiments which are conducted with real and synthetic data. Section 5 concludes the thesis with concluding remarks and some future work.

2. LITERATURE SURVEY

In this section, we firstly provide some background about the studies carried out solely on the ONP and then studies carried out on related problems. Thereafter, brief definitions of the ant system, ant colony system, genetic algorithm, simulated annealing and migrating birds optimization are given.

2.1. ONP and Related Problems

In this subsection, we firstly present the studies carried out on the ONP and give the contributions of this study in comparative manner. Then we make a survey on the studies related with the ONP because the techniques developed in this study may also be applied to them.

2.1.1. Previous Work on ONP

In the literature, ONP is defined in (Alkaya et al., 2014) where the authors propose a heuristic for solving the ONP. The proposed algorithm is works as follows: firstly, find the highest penalty term $\alpha^* \geq 1$ such that the shortest path without constraints (i.e., the path found by discarding neutralization bounds) with Euclidean length of disc-intersecting edges increased by $(\alpha^* C)/2$ entails the maximum neutralization numbers without exceeding K, therefore the name penalty search algorithm (PSA). PSA returns this path and neutralization limit constraint is satisfied in this path. Straightforward bisection method is used to find the penalty term. They present special cases where their algorithm is provably optimal. However, the PSA works correctly under the assumptions of (1) equal radii of the discs, and (2) equal neutralization cost of the discs which may not be realistic in many cases.

In another study on the ONP, an exact algorithm is proposed (Alkaya & Oz, 2016). The exact algorithm consists of two phases. In the first phase an upper bound to the problem is obtained by using the PSA algorithm. In the second phase, if there is a gap from optimal solution, starting from the bound obtained from phase I, a k-th shortest path algorithm is exploited to find the optimal solution. The performance of the exact algorithm is tested on both grid and continuous graphs where it works very fast on small and moderate sized graphs. However, since it is based on the PSA it requires the same assumptions that PSA has.

In another study, an ant system algorithm for the ONP is developed (Algin et al., 2013). In their proposed algorithm, the state transition rule makes use of certain problem-

specific information to guide the ants. They show how the parameters of the algorithm can be fine-tuned for enhanced performance and they present limited computational experiments including a real-world naval minefield dataset. However, in their study they just present the algorithm and the results of the computational experiments without any other metaheuristic comparison.

Summarizing the shortages of the state-of-the-art studies on ONP, we can easily say that: (1) PSA and exact method works correctly under the assumptions of equal radii of the discs, and equal neutralization cost of the discs which may not be realistic in many cases (2) the performance metaheuristics are not fully exploited. Therefore, in this study, our contributions are as follows:

- we develop and compare metaheuristic algorithms, namely ant system algorithms, ant colony system, genetic algorithms, simulated annealing and migrating birds optimization algorithms, that can solve the ONP instances having various radii and neutralization cost,
- our GA, SA and MBO algorithms designed for the ONP outperform AS which was developed for ONP in a former study,
- we show that the proposed metaheuristics present very close results on small and moderate sized graphs and therefore can be used on large graphs.

Next, we present a survey of the studies realized on the related problems.

2.1.2. Previous Work on Related Problems

In a problem very similar to the ONP, there is a mine hunting team which operates before the agent and neutralizes the necessary discs (not exceeding the given limit) to create a zero-risk path for the agent. However, in that problem the neutralization cost is considered as zero since the objective is to minimize the length of zero-risk path for the agent, not the mine hunting team. Therefore, that problem is totally different from ONP and hence their techniques cannot be used for ONP. Bekker & Schmid (2006) formulate a fitness function which favors shorter paths with fewer numbers of neutralized discs and use genetic algorithms to tackle the problem. Li (2009) develops a mission planning tool to find a minimum-risk route for a surface ship through a mapped minefield. He also proposes a greedy heuristic to form a prioritized list of mines to be cleared (neutralized) so that a zero-risk path is obtained.

However, in his problem definition he assumes that a minesweeper is sent before the ship and it can reach and clear any mine in any sequence, safely.

On the other hand, there are several problems which are closely related with the ONP such as signal routing in telecommunication networks with QoS guarantees. In this problem, the objective is to send data to the destination within a delay constraint by minimizing the path cost, either in terms of money or cost of utilizing network resources. In the literature, that problem is studied under the name Delay Constrained Least Cost (DCLC) path problem. Reeves & Salama (2000) study the above mentioned problem and develop a simple distributed heuristic method called the Delay Constrained Unicast Routing (DCUR) algorithm. Lagrangian relaxation techniques are used by Jüttner et al. (2001) to solve it. Guo & Matta (2003) develop Delay-Cost Constrained Routing (DCCR) algorithm for this problem problem. The authors use a variant of the Lagrangian relaxation method developed by Handler & Zang, (1980) to reduce the search space exploited by DCCR. The aforementioned heuristics and other six heuristics for DCLC are compared in (Kuipers et al., 2004). In their study these proposed algorithms for DCLC are compared according to their solution quality and run-time complexity. According to the simulation results on square lattices DCUR algorithm gives better results in general. DCUR is also used in (Alkaya et al., 2014) for a comparison with PSA where PSA outperforms DCUR.

Minimizing the risk for the military vehicles and aircraft in a threat environment is another closely related topic to the ONP. In that problem, the aim is minimizing the total threat on the route taken from start point to target point that satisfies a given constraint due to supply of fuel or flight time. Lee (1995) works on a problem where the search space which is discretized into a three dimensional grid and the constraint of the problem is a given consumption limit. Similar to (Handler & Zang, 1980), the lagrangian dual of the problem is used where in order to find the best lagrange multiplier value, bisection method is used. Genetic algorithms are used in (Latourell et al., 1998) to minimize the threat on the military vehicles' path. In their model, they consider the limitation on the sharp angels of the turns that the aircraft can make. Zabarankin et al. (2002) study both discrete and analytical optimization approaches. They show that optimum can be reached when there is only one radar in a continuous setting. For discrete setting they use the algorithm proposed in (Dumitrescu & Boland, 2001). In a further study, they extend their work and develop a model where the trajectory of an aircraft in a three-dimensional setting is determined (Zabarankin et

al., 2006). Royset et al. (2009) apply the algorithm in (Carlyle et al., 2008) to route planning problem for various types of military aircraft.

Another related application is curve approximation. In the domains like computer aided design, computer graphics, image processing, and mathematical programming, piecewise linear functions are often used to approximate complex curves (Dahl & Realfsen, 1997). Such an approximation, on the other hand, often requires optimization in the presence of bandwidth or storage limits. Dahl & Realfsen (2000) aim to minimize the approximation error and study four different algorithms: a dynamic programming algorithm, Lagrangian relaxation based algorithm, a combinatorial algorithm, and a linear programming algorithm. Nygaard et al. (1998) study the same problem and they use dynamic programming approach for solving the problem.

State-of-the-art algorithms proposed for the aforementioned problems either demonstrate poor performance or require significant computational resources on the ONP especially when the problem dimension gets bigger. As introduced in the next subsections, we use ant system, ant colony system, genetic algorithms, simulated annealing and migrating birds optimization metaheuristics and modify them to be applicable in solving the ONP in an efficient way which can also be used for the related applications.

2.2. Ant System

Ant system (AS) algorithm was first introduced by Marco Dorigo in 1992 and applied for solving the Travelling Salesman Problem (TSP). In real life ants have the ability to find shortest path between their nests and the food. They do this with the help of some chemical called pheromone. The pheromone helps ants to communicate with each other. An ant lays some pheromone on the path that it uses. When another ant comes and senses this pheromone it is also inclined to choose this path. Hence, the pheromone level on this path increases. On the other hand, there is pheromone evaporation on all paths over time. With the pheromone evaporation, unselected paths becomes less desirable and after some time almost all ants start to choose the path which has higher pheromone level. To understand better, consider the scenario given in Figure 3. In this scenario, firstly, ants are walking between food and nest without any obstacles on their path Figure 3(a). Suppose an obstacle is placed on the path Figure 3(b). Immediately after this case, about half of the ants choose the upper path, the other half chooses the lower path Figure 3(c). As the time passes, since ants walking on the shorter

lower path reach the other side more quickly, more pheromone accumulates on the shorter path Figure 3(d). Consequently, more and more ants start to choose this lower path over time.

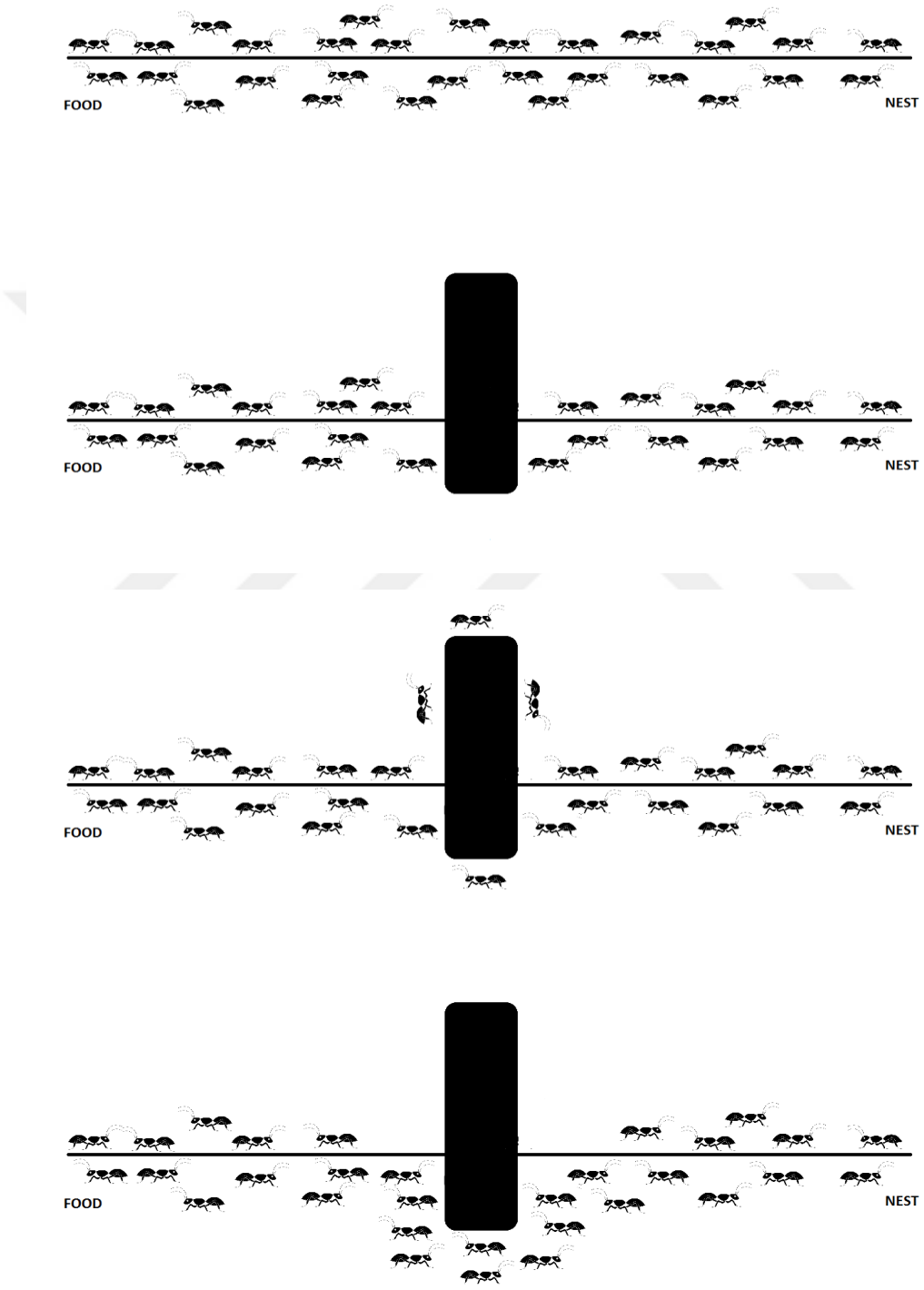

Figure 3 – (a) Ants are walking between food and nest without any obstacles on their path. (b) An obstacle occurs. (c) About half of the ants choose the upper path, the other half chooses the lower path. (d) Since ants walking on the shorter lower path reach the other side more quickly, more pheromone accumulates on the shorter path. Consequently, more and more ants start to choose this lower path over time.

To run AS on a problem, that problem must be modeled as a graph *G = (V,E)* where *V* is the set of vertices and *E* is the set of edges. In the AS algorithm, first all ants are initialized and then every ant is placed on a vertex. According to the predetermined initial pheromone level (τ_0) all edges are initialized. After initialization part is finished, ants start building their paths by choosing next vertex according to the state transition rule (STR), if that vertex is not visited before. The state transition rule assigns probability to the edges and edges are selected according to this probability (Eq.6)). When all ants complete their tours, global update rule is applied. In global update rule, pheromone evaporation occurs on all edges. Despite the pheromone evaporation, if ants lay some pheromone on an edge, the pheromone level on that edge increases and it becomes more desirable by other ants. On the other hand, the edges that are not selected by ants lose pheromone and become less desirable.

The state transition rule for ant system is applied according to the formula given by Eq. (6). This formula gives the probability of ant $k(a_k)$ that wants to go from vertex *x* to vertex *y*.

$$p_k(x,y) = \begin{cases} \frac{[\tau(x,y)] \,.\, [1/\,\delta(x,y)]^{\beta}}{\sum_{u \in \mathrm{J_k}(x)} [\tau(x,u)] \,.\, [1/\delta(x,u)]^{\beta}}, & \text{if y} \in \mathrm{J}_{a_k}(x) \\ 0, & otherwise \end{cases} \quad (6)$$

where τ is the pheromone, $\delta(x,y)$ is the cost of edge that is between vertex *x* and vertex *y*. β is a parameter which determines the relative importance of pheromone versus distance ($\beta > 0$). $\mathrm{J}_{a_k}(x)$ is a set that keeps the possible vertices that a_k can travel. In Eq. (6), by multiplying the $\tau(x,y)$ and the heuristic value $1/\,\delta(x,y)$, the edges which have short length get higher pheromone.

When all ants reach the destination, global update rule is applied to all edges according to the formula given in Eq. (7).

$$\tau(x,y) = (1-\alpha).\tau(x,y) + \sum_{k=1}^{m} \Delta\tau_k(x,y) \quad (7)$$

where $\Delta\tau_k(x,y) = \begin{cases} \frac{1}{L_k}, & \text{if } (x,y) \in \text{ path constructed by ant } k \\ 0, & otherwise \end{cases}$

α is pheromone decay parameter between 0 and 1. *m* is the number of ants and L_k is path length performed by a_k.

2.3. Ant Colony System

Ant colony system (ACS) algorithm is similar to the ant system algorithm but there are three main differences in applying;

- the state transition rule (STR),
- the global update rule (it is applied to the edges of best ant's path), and
- the local update rule (which does not take place in AS).

ACS algorithm works like the AS algorithm. Ants are initialized and then positioned on vertices randomly and they start to construct their paths by applying the STR. While constructing the paths each ant applies local update rule to the visited edges. With the local update rule pheromone level on edges is modified. When all ants finish constructing their paths, pheromone level on edges is modified again by applying the global update rule. With this rule, the pheromone level on edges which belong to the current shortest path is increased and so other ants will choose the edges which are near to the current shortest path with the hope of finding the optimum path. In this algorithm, local update rule is used to shuffle the paths so that ants will make better use of pheromone information instead of searching in a narrow neighborhood of the best previous path (Dorigo & Gambardella, 1997).

The STR for ant colony system is applied according to the formula given by Eq. (8). This formula gives the probability of a_k that wants to go from vertex *x* to vertex *y*.

$$y = \begin{cases} \arg\max_{u \in J_k(x)} \{[\tau(x,u)] \,.\, [1/\delta(x,u)]^{\beta}\}, & \text{if } q < q_0 \text{ (exploitation)} \\ S, \quad otherwise & (biased\ exploration) \end{cases} \tag{8}$$

where q and q_0 are random numbers between 0 and 1, *S* is a random variable selected according to Eq. (6). q_0 is a parameter which determines the relative importance of exploitation versus biased exploration.

Global updating rule of ACS is defined as follows:

$$\tau(x,y) = (1-\alpha).\tau(x,y) + \alpha.\Delta\tau(x,y) \tag{9}$$

where $\Delta\tau(x,y) = \begin{cases} \frac{1}{L_{gb}}, & \text{if } (x,y) \in \text{global best path} \\ 0, & otherwise \end{cases}$

where α is pheromone decay parameter between 0 and 1, L_{gb} is the length of global best path.

ACS local updating rule is defined as follows:

$$\tau(x,y) = (1-\rho).\tau(x,y) + \rho.\Delta\tau(x,y) \quad (10)$$

where ρ is pheromone decay parameter between 0 and 1. Here $\Delta\tau$ is set as initial pheromone level ($\Delta\tau = \tau_0$) as preferred in (Dorigo & Gambardella, 1997). The pheromone level of edge decreases after the local update rule is applied. Instead of searching in neighborhood of previous best path, ants make search in a wider area to find better solutions.

2.4. Genetic Algorithm

Genetic algorithms are population based methods inspired by the principles of natural evolution (Holland, 1986). The algorithm starts the search with a population of individual chromosomes (solutions) generated randomly or heuristically. At each iteration, the population is evolved using genetic operators such as mutation and crossover to produce offspring (new individuals of the next generation). Mutation is unary operator that introduces random modifications of the chromosome in order to add diversity to the population. The crossover operator combines two parents (individuals from the current generation) to generate new offspring. The crossover operation aims to propagate good solution components from parents to offspring. The selection mechanism chooses the parents based on survival of the fittest. That is, the better fitness values are more likely to be chosen to undergo reproduction in order to produce offspring (Beasley et al., 1993).

2.5. Simulated Annealing

Simulated Annealing (SA) is commonly said to be the oldest among the meta-heuristics and surely one of the first algorithms that had an explicit strategy to escape from local minima. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution (uphill moves) in order to escape from local minima. The probability of doing such a move is decreased during the search. The algorithm starts by generating an initial solution, *r*, (either randomly or heuristically constructed) and by initializing the so-called temperature parameter *T*. Then, at each iteration a solution *r'* in *N(r)* is randomly sampled and it is accepted as new current solution depending on *f(r), f(r')* and *T* where *f(r)* denotes the cost of solution *r*. *r'* replaces *r* if $f(r') < f(r)$ or, in case $f(r') \geq f(r)$, with

a probability which is a function of T and $f(r')-f(r)$. The probability is generally computed following the Boltzmann distribution (Blum & Roli, 2003).

2.6. Migrating Birds Optimization

The Migrating Birds Optimization (MBO) algorithm is one of the swarm intelligence techniques proposed recently inspired from the migrating birds and their V formation. There is a leader bird in the algorithm and it is chosen randomly, whereas the other birds are divided into two groups behind the leader bird as in V formation. Each bird generates a number of feasible solutions that determines the speed of the flock (rather than feasible solution, solution will be used in the remainder of the manuscript). Flock's search area depends on its speed. If the speed is higher this means flock can do search in a wider area.

The algorithm works as follows. Initial solutions are generated by placing the birds to the search space randomly in a hypothetical V formation. The leader bird generates its neighbors and selects one of them that is better than the current solution. Then the leader bird gives a parametric number of best unused solutions to the bird behind. After all birds share their neighbors with the following birds, one so-called flapping is completed. After m flappings the leader is replaced to the last position of one of the tails and the process starts over. The stopping criterion in this algorithm is the number of iterations or number of neighbors generated.

In the next section, we provide detailed information about how the aforementioned metaheuristics are modified to give high performance for the obstacle neutralization problem.

3. PROPOSED AS, ACS, GA, SA and MBO for the ONP

In this section, we describe how the metaheuristics are adapted for an efficient exploitation on ONP. In the following subsection we firstly describe the underlying graph structure and explain how the feasibility of a path is determined. Then, we describe the details of our AS and ACS algorithms adapted for the ONP especially with all the details about the STR. After that, the details for GA, SA and MBO are given which are again customized for ONP exploitation.

3.1. Graph Structure

An instance of the ONP can be represented by graph (*V*,*E*). In this graph there is a set of discs and the edges intersecting these discs have additional travelling cost which is calculated proportionally by the number of discs intersected. Actually, these additional costs represent the neutralization cost of the discs. Another property of the edges is their weight values. Simply, weight of an edge represents the intersection status with a disc. So, the number of intersecting discs determines the weight value for an edge. Therefore, the weight property of a path is used for checking its feasibility, i.e. satisfying the maximum number of neutralization (*K*) constraint. The details for these calculations are given in the Section 4.

3.2. AS and ACS

In the ONP, there are agents that try to navigate from *s* to *t* swiftly. In this subsection we use ants as agents and the ants have same goal. In STR (Eq. 6 and Eq. 8) of AS and ACS there are cost (δ) and pheromone level (τ) parameters to guide the ants to the shortest path. However, these parameters are not enough to guide the ants in the ONP. A path found by ants can be the shortest path but it can be also an infeasible path for the ONP due to its weight. Therefore, in this study the STR is modified by adding the weight parameter, so that the ants will be aware of the weight information while constructing their paths. At each step, with the cost, pheromone level, and weight information the ants will choose the next vertex to go next.

We modified the original STR of AS and ACS to apply the ONP. In our modified STR the probability of ant *k* (a_k) that wants to go from vertex *x* to vertex *y*, is found according to formula given in Eq. (11).

$$\textbf{STR 1: } p_k(x,y) = \begin{cases} \dfrac{[\tau(x,y)]\,.\,[1/\omega(x,y,t)\;\delta(x,y)]^{\beta}}{\sum_{u \in \mathrm{J_k}(x)}[\tau(x,u)]\,.\,[1/\omega(x,u,t)\delta(x,u)]^{\beta}}, & \text{if y} \in \mathrm{J}_{a_k}(x) \\ 0, & otherwise \end{cases} \tag{11}$$

where $\tau(x,y)$ is the pheromone level of edge which is between vertex x and vertex y, t is our terminal or destination point, $\delta(y,t)$ is total cost of path that is between vertex y and vertex t (terminal). β is a parameter which determines the relative importance of pheromone versus distance (β>0). $J_{a_k}(x)$ is a set that keeps the possible vertices that a_k can travel. ω is the weight function which is found by

$$\omega(x,y,t) = \eta(\lambda(x)) - \mu(k) + \vartheta(x,y) + 1 \tag{12}$$

where $\lambda(x)$ is the shortest path from vertex x to t, $\eta(\lambda(x))$ is weight of shortest path from vertex x to t, $\mu(a_k)$ is the number of remaining neutralizations for a_k, $\vartheta(x,y)$ is weight of edge which is between vertex x and vertex y. The reason why we add a constant term to this formula is to avoid a zero value as a denominator in Eq. 12.

$$y = \begin{cases} \arg\max_{u \in J_{a_k}(x)} \{[\tau(x,u)] \,.\, [1/(\omega(x,y,t).\delta(y,t))]^{\beta}\}, & \text{if } q < q_0 \text{ (exploitation)} \\ S, \quad otherwise & (biased\ exploration) \end{cases} \tag{13}$$

where q and q_0 are random numbers between 0 and 1, S is a random variable selected according to (6). q_0 is a parameter which determines the relative importance of exploitation versus biased exploration.

In this formulation, we favor the edges which have greater amount of pheromone. and which have smaller cost and smaller weight on the shortest path to t. Global update rule that we use in our problem is exactly same as in (Eq. 7 and 9) for AS and ACS respectively. For ACS local update rule that used in our problem is also same with (10).

We also generate four more STR for AS and ACS then compare them with each other. Other STRs are listed below.

$$\textbf{STR 2}: p_k(x,y) = \begin{cases} \frac{[\tau(x,y).\omega(x,y,t)] \,.\, [1/\,\delta(y,t)]^{\beta}}{\sum_{u \in J_{a_k}(x)} [\tau(x,u).\omega(x,u,t)] \,.\, [1/\,\delta(u,t)]^{\beta}}, & \text{if } y \in J_{a_k}(x) \\ 0, & otherwise \end{cases} \tag{14}$$

$$\textbf{STR 3}: p_k(x,y) = \begin{cases} \frac{[\tau(x,y)] \,.\, [1/\,\delta(y,t)]^{\beta}}{\sum_{u \in J_{a_k}(x)} [\tau(x,u)] \,.\, [1/\delta(u,t)]^{\beta}}, & \text{if } y \in J_{a_k}(x) \\ 0, & otherwise \end{cases} \tag{15}$$

$$\textbf{STR 4}: p_k(x,y) = \begin{cases} \frac{[\tau(x,y)].[mw-\omega(x,y,t)]\,.\,[1/\,\delta(y,t)]^{\beta}}{\sum_{u\in J_{a_k}(x)}[\tau(x,u)].[mw-\omega(x,u,t)].\,[1/\delta(u,t)]^{\beta}}\,, & \text{if } y \in J_{a_k}(x) \\ 0, & otherwise \end{cases} \quad (16)$$

$mw = \max\{\omega(x,u,t)\}$

$$\textbf{STR 5}: p_k(x,y) = \begin{cases} \frac{[\tau(x,y)]\,.\,[1/.cost(v,t)]^{\beta}}{\sum_{u\in J_{a_k}(x)}[\tau(x,u)]\,.\,[1/cost(u,t)]^{\beta}}\,, & \text{if } y \in J_{a_k}(x) \\ 0, & otherwise \end{cases} \quad (17)$$

$$\text{cost}(v,t) = \Omega.\delta(y,t) + (1-\Omega).\omega(y,t) \quad (18)$$

Descriptions of five STRs are given briefly.

STR1: In this function we favor the edges which have greater amount of pheromone, lower cost and smaller weight on the shortest path to t. So ant makes neutralization as late as possible. With this function and can finds the path that has less neutralization number.

STR2: In this function we favor the edges which have greater amount of pheromone, higher weight and lower cost on the shortest path to t. So ant makes max number of neutralization at the beginning then follows the path that found by Dijkstra's algorithm.

STR3: This is same as in original AS algorithm. Ant generates path by ignoring the weight parameter and of course without exceeding the max limit number. Ant follows the path which has lower cost and higher pheromone level.

STR4: In this function ant follows the path that has higher pheromone level and lower cost. While ant choosing the next vertex to go, it chooses the vertex that has higher weight number between this vertex and destination point.

STR5: In this function ant follows the path that has higher pheromone level and lower cost. New cost function is generated for this STR. According to this cost function cost is calculated not only path length but also weight parameter by some constant ratio. Ω, is a given parameter between 0 and 1.

When we compared these STRs, experiments show that all functions give similar results but STR1 has best results among them. The results can be seen in Table 1. In other experiments we choose STR1 and conducted according to this.

Table 1 – Comparison of STRs

STR	Cost (average)
STR1	**121.29**
STR2	123.30
STR3	122.77
STR4	122.68
STR5	122.51

Based on the above definitions and equations, our AS and ACS algorithms developed for the ONP is given in Figure 4. During the initialization of the underlying graph, one important step is the invocation of the Dijkstra's algorithm. With that, each vertex keeps the cost and weight values of the path from that vertex to *t*. The stopping criteria for both AS and ACS is the iteration number which is a parameter of these algorithm as given in Table 2.

```
1. Initialize the underlying graph
2. repeat
3.       Initialize ants and place on start point
4.       if for any ant a_i , μ(a_i ) ≤ η(λ(s))
5.                Ant follows λ(s)
6.                Return  λ(s)
7.       repeat
8.                for each ant, a_i
9.                         if μ(a_i )=0
10.                                a_i follows zero neutralization path found by Dijkstra's algorithm
11.                        else a_i chooses next vertex (nv) according to the STR
12.                        if nv = null /*This means ant jammed, cannot move*/
13.                                Restart the ant
14.                        else ant moves to nv
15.                        Apply local update rule (only for ACS)
16.                        if η(λ(nv)) ≤ μ(a_i )
17.                                obtain the shortest path from nv to t using Dijkstra's algorithm
18.                                ant follows this path
19.                                set status of a_i as finished
20.      until all ants finish their paths
21.      Apply global update rule to all edges
22. until stopping criteria is met
23. Return the shortest path found up to now.
```

Figure 4 – Pseudocode of our AS and ACS algorithm for the ONP

3.3. GA, SA and MBO

Recall that the goal in the ONP is to find the shortest path from s to t with the constraint of neutralizing at most *K* discs. Therefore, we can consider the problem as selecting at most K discs whose neutralization produces the shortest path. Hence, for SA, GA

and MBO, a solution to the ONP will be represented with a set, say S, which has at most K discs. In order to calculate the cost of a solution, we set the neutralization cost of the discs in $\mathcal{A}\backslash S$ to a maximum value, and find the shortest path, p_S, using Dijkstra's algorithm. The cost of the path p_S is returned as the cost of the solution. Then, the neutralization costs of all discs are returned to their original values since we need the original cost values before starting to query another solution. An example solution is given in Figure 5.

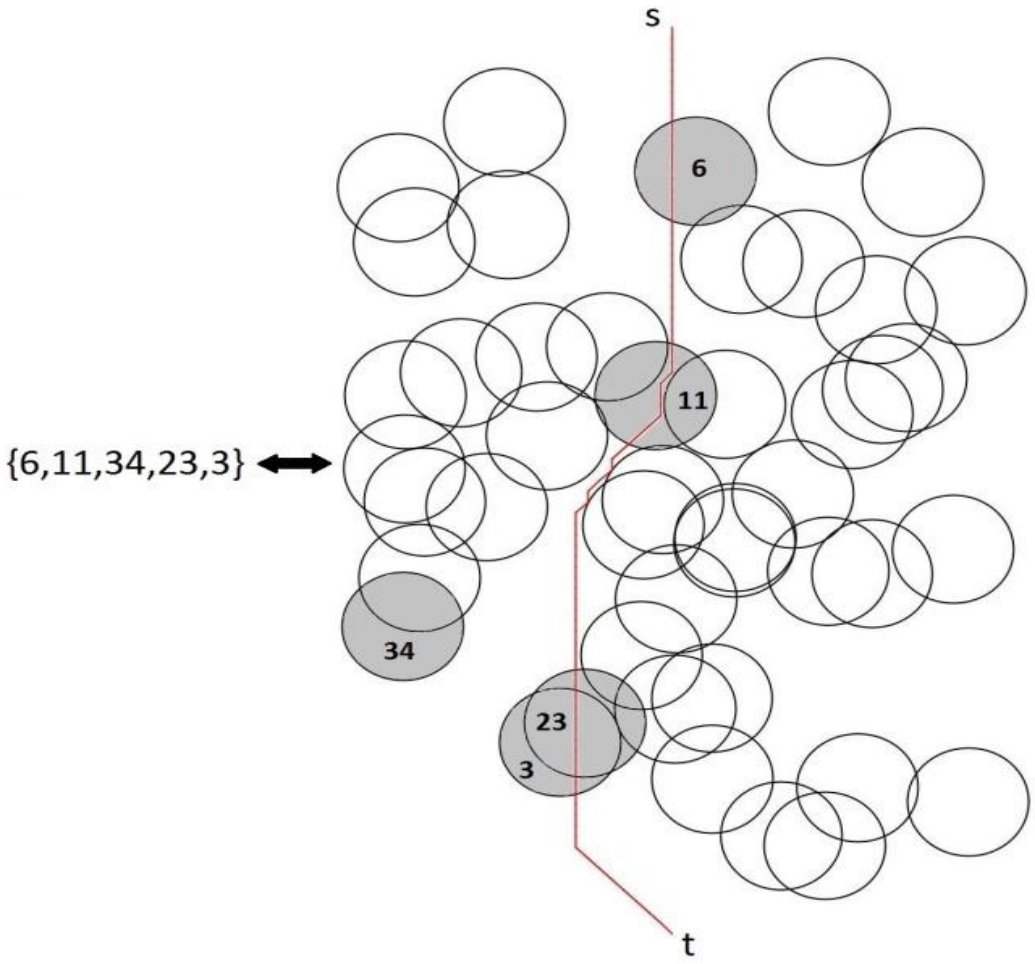

Figure 5 – A solution with five discs: {6,11,34,23,3}. In order to calculate the cost of this solution neutralization cost of all discs except {6,11,34,23,3} are maximized. Then, under this setting, the shortest path is found using Dijkstra's algorithm. In this example only four of the discs are neutralized. After finding the shortest path, the neutralization cost of all discs is returned back to their original values.

In the following subsections, we introduce the details of the implemented GA, SA and MBO algorithms.

3.3.1. SA

Our SA implementation is given in Figure 6. In this implementation, the algorithm stops either when the T parameter reaches 0 or the best solution is not improved for a predefined number of times.

```
1. generate a random solution and mark it as current solution, cs
2. best solution (bs) = cs
3. while termination condition is not satisfied
4.       for R times
5.                obtain a neighbor solution, ns, of cs
6.                delta= cost of ns – cost of cs
7.                if delta <0
8.                         cs=ns
9.                else if random() < e^(|delta|/T)
                           cs=ns
11.               if cost of ns < cost of bs
12.                        bs = ns
13.      end for
14.      T= T / a
15.      R = R * b
16. end while
```

Figure 6 – Pseudocode of our SA

The parameters for our proposed SA algorithm are as follows:

- Initial temperature (*T*): The larger this value, the more inferior exchanges encouraged. In numerical computations, *T* is set to either 100 or 1000.
- Temperature decrease ratio (*a*): After a predetermined number of iterations, *T* is set to *T=a (i.e., T := T=a).* When *a* is large, the temperature decrease is faster and the acceptance of inferior exchanges become less likely at a greater rate. In numerical computations, *a* is set to either 1.1 or 1.5.
- Number of iterations at each temperature setting (*R*): Greater values of *R* correspond to slower cooling, that is, more exchanges occurring when there is a greater likelihood of inferior exchanges being accepted. In numerical computations, *R* is set to either 5 or 20.
- Increase ratio in iteration number at each setting (*b*): After a predetermined number of iterations, *R* is set to *R x b (i.e., R := R x b).* In numerical computations, *b* is set to either 1.1 or 1.5.

The most important operation of a simulated annealing implementation is the neighbor finding method. As explained above, a solution for an ONP is a set of discs, say *S*, of size *K*.

In our implementation, a neighbor of a solution is obtained by replacing one of the elements of *S* by one of its closely located discs by avoiding any replicates in *S*. Specifically, a disc is closely located to another if their centers are at most 3 * radius away from each other. If there are no available closely located discs around a disc, then any disc from $\mathcal{A}$ is selected for replacement. Figure 7 depicts our neighbor finding method.

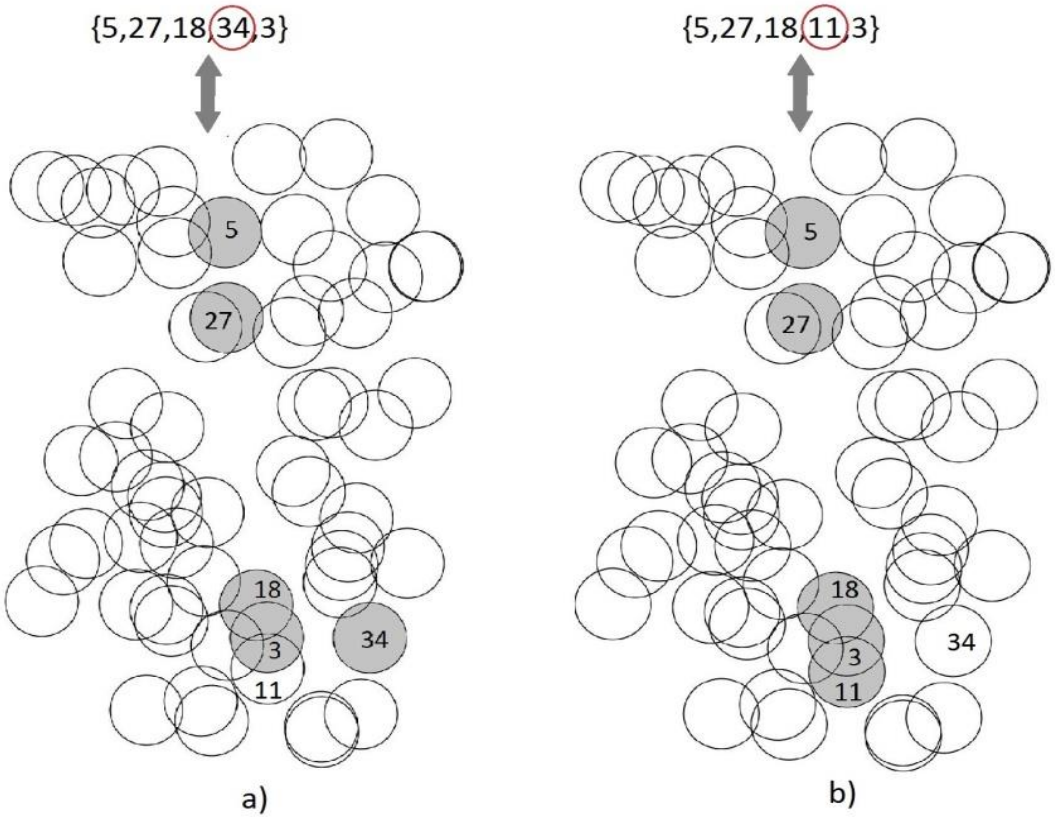

Figure 7 – In (a) a solution with five elements is represented. One of the elements of the solution is selected randomly (d_{34}). In (b) In the solution set, d_{34} is replaced with one of its neighbors (d_{11}).

3.3.2. GA

Pseudocode of our GA is given in Figure 8. The termination condition for the algorithm is either a convergence condition which is achieved when all chromosomes in the population have the same cost or a predefined number of iterations, *nog*, which is a parameter of the algorithm.

```
1.  generate nos solutions and put them in a population set, ps
2.  while termination condition is not satisfied
3.      for nos/2 times
4.          Select two solutions from ps, s1 and s2
5.          Generate a random number, rn, between 0 and 1
6.          if rn < xp
7.              Apply crossover to s1 and s2 and obtain two offspring, s3 and s4
8.          else
9.              Create s3 and s4 as clones of s1 and s2, respectively
10.         Mutate s3 and s4 by probability mp
11.         Add s3 and s4 into ps
12.     end for
13.     Leave best nos solutions in ps and remove the rest from ps
14. end while
```

Figure 8 – Pseudocode of our GA

In our genetic algorithm implementation, firstly, *nos* number of random initial solutions are produced. While generating a new generation, *nos* offspring are generated using the crossover operator and added to the population set (*ps*) after being mutated. Then best *nos* solutions are chosen for constituting the next generation. Parameters of the algorithm are *nos*, *nog*, *mp* and *xp*. *nog* represents how many generations are being built, *nos* is the number of solutions surviving in each generation, *xp* is the crossover probability and *mp* is the mutation probability.

In order to apply genetic algorithms to the ONP, we need to define chromosome structure as well as the mutation and crossover operators. Our chromosome setting for the obstacle neutralization problem includes *K* number genes which are the IDs of the discs that are allowed to be neutralized. For example, if there are 100 discs on minefield (with IDs from 1 to 100) and *K* is given as five, then a chromosome can be like one given in Figure 9.

Figure 9 – Chromosome structure used in the proposed GA

In order to determine the best crossover technique for the ONP, we designed several crossover methods and found the best one after extensive computational experiments. The crossover method that is decided on is depicted in Figure 10 and called interference crossover.

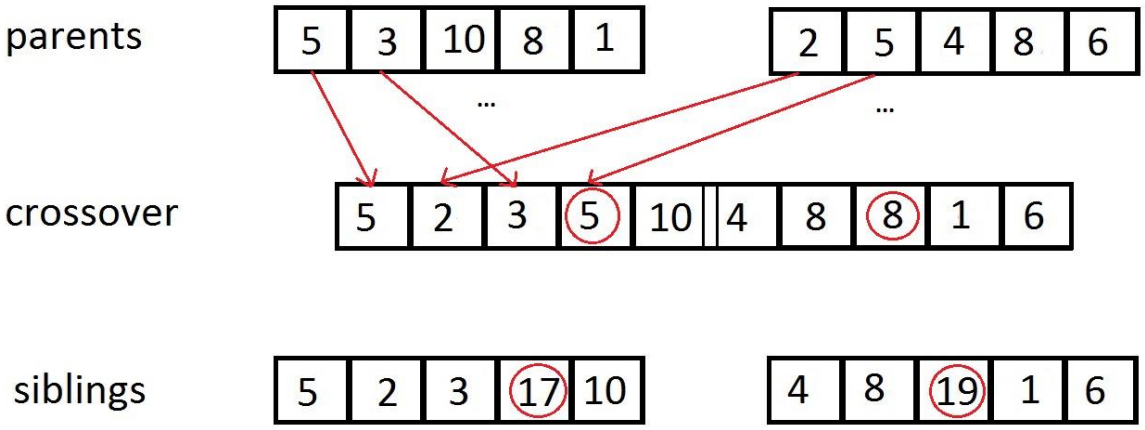

Figure 10 – Interference crossover: Two chromosomes are crossed over in an interfering manner and a longer chromosome is obtained. Then this interfered chromosome is split into two. If there are repetitions in a sibling chromosome, then the repeated genes (discs) are replaced with one of their neighbors using the neighbor function given in the previous subsection. In this specific example first sibling in order to avoid the repetition of d_5, it is replaced with d_{17}, whereas, in the second sibling d_8 is replaced with d_{19}.

The mutation operator of GA is just the same that we use for obtaining a neighbor for SA.

3.3.3. MBO

Pseudocode of our proposed MBO algorithm is given in Figure 11. The termination condition for the algorithm is either a convergence condition which is achieved when all birds (solutions) in the flock have the same cost or a predefined number of iterations, *noit*, which is a parameter of the algorithm.

1. Generate *nob* initial solutions in a random manner and place them on a hypothetical V formation arbitrarily
2. **while** termination condition is not satisfied
3. **for** *nof* times
4. Try to improve the leading solution by generating and evaluating *non* neighbors of it
5. **for** each solution s_i in the flock (except leader)
6. Try to improve s_i by evaluating (*non–olf*) neighbors of it and *olf* unused best neighbors from the solution in the front
7. **end for**
8. **end for**
9. Move the leader solution to the end and forward one of the solutions following it to the leader position
10. **end while**
11. **return** the best solution in the flock

Figure 11 – Pseudocode of our MBO

MBO algorithm has the following parameters: number of birds (*nob*), number of tours (*not*), neighbor solutions number to be generated from a solution (*non*) and solutions number to be shared with the following solution (*olf*). However, due to the inherent design of the algorithm *non* value has to be equal to or greater than 2**olf* + 1. In order to apply MBO to the ONP, we need to define the neighbor finding function. The neighbor finding function of MBO is just the same that we use for obtaining a neighbor for SA (see Figure 7).

4. EXPERIMENTAL WORK AND DISCUSSION

In this study, we modeled the underlying graphs using grids. That is, the obstacle field is discretized using 8-regular lattices. An example to our discretization is provided in Figure 12 – **Lattice graph for ONP**.

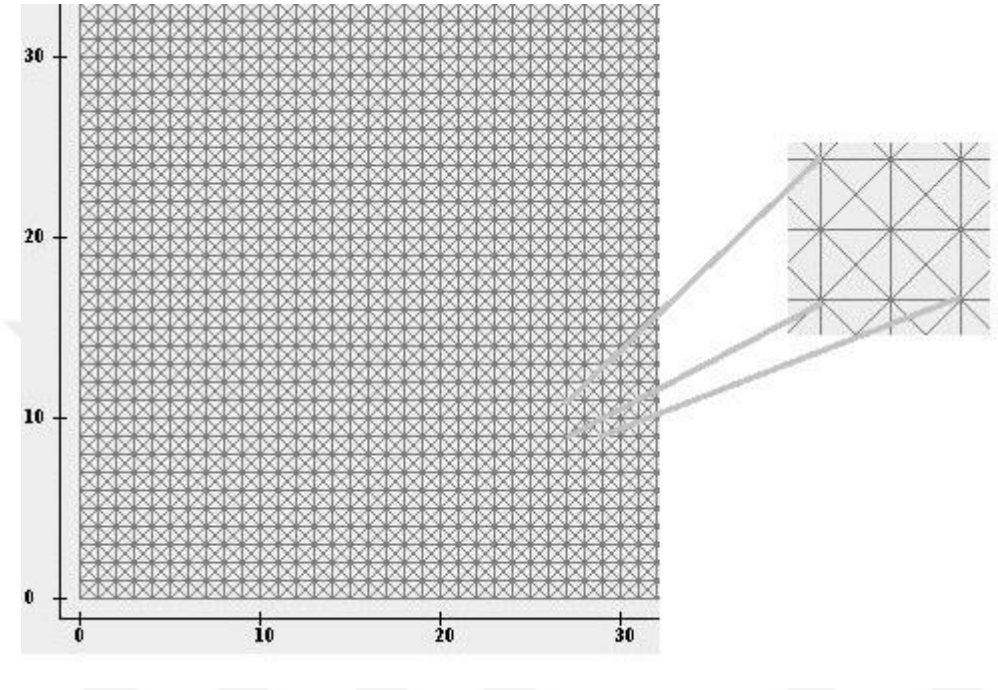

Figure 12 – Lattice graph for ONP

4.1. Experimental Setup

After the proposed metaheuristics for the ONP are implemented, computational tests are performed with real and synthetic data. The real data that we used in our experiments is COBRA data which is the U.S. Navy minefield data set. This data set contains 39 disc-shaped obstacles and first appeared in (Witherspoon et al., 1995). Figure 13 shows the COBRA minefield. Experimental work on COBRA data presents that all metaheuristics reach the optimum results.

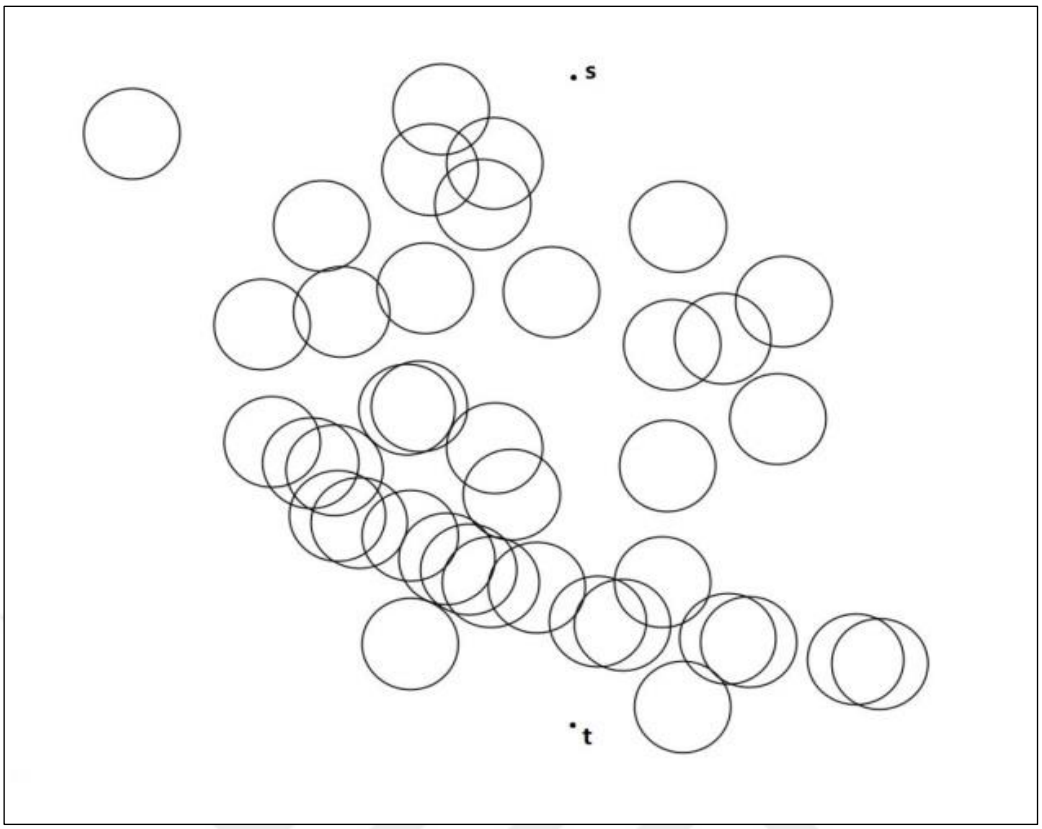

Figure 13 – COBRA minefield

In addition to the real data explained above, we used several COBRA-like instances in order to fully reveal the performance of the proposed algorithm. Specifically, we created 10 instances for the obstacle field with 100 discs with radius=5 on a [0,100]x[0,100] rectangle.

A desirable feature of the lattice discretization is that its resolution can be increased or decreased as needed to achieve a desired balance between accuracy and computational burden. As an example on one of random obstacle fields we present the solution of a problem instance (s, t, $\mathcal{A}$, 1, 5) discretized space with three different resolution settings (Figure 14) p_1, p_2 and p_3 are the optimum solutions when discretization is performed with 10x10, 20x20 and 50x50 vertices, respectively ($\delta(p_1) = 121.57$, $\delta(p_2) = 113.28$, and $\delta(p_3) = 110.63$ where $\delta(p)$ is the cost of a path).

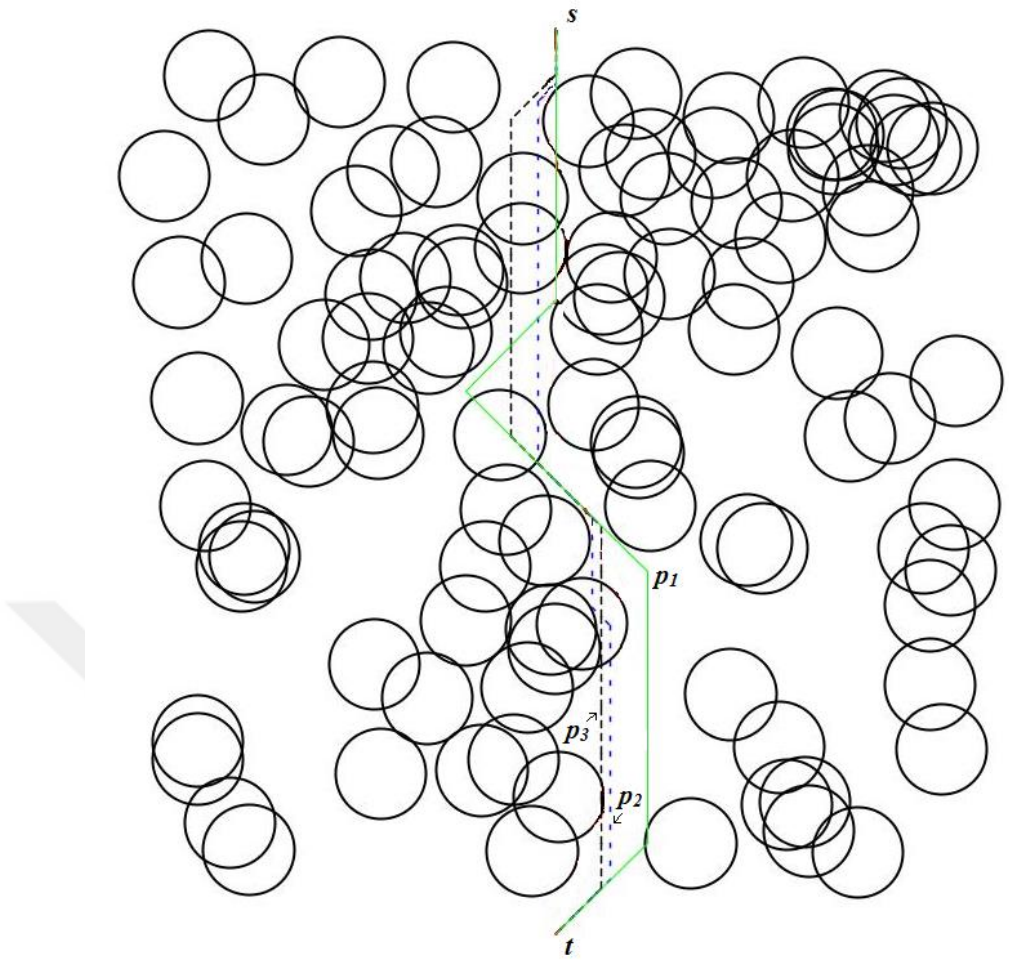

Figure 14 – Solution of a problem instance (s,t, $\mathcal{A}$,1,5) on discretized space with three different resolution settings

The algorithms are compared using two criteria. The first one is the cost of the paths returned by the algorithms. To have a concrete idea on the quality of the paths returned by the algorithms, we compared them with the exact solutions which are obtained by using an exact algorithm proposed in a recent study (Alkaya & Oz, 2016). The second comparison criterion is the run time of the algorithms in seconds.

4.2. Scip and Zimpl

As a problem solver we used SCIP tool and ZIMPL language to model the ONP (www.http://scip.zib.de/ and www.http://zimpl.zib.de/). SCIP is a free solver. According to the ZIMPL language we modeled the ONP as below and then solved it with SCIP solver.

```
set V :={read "gams_edgedata.txt" as "<1s>"};
set A :={read "gams_edgedata.txt" as "<1s,2s>"};
```

```
param c[A] := read "gams_edgedata.txt" as "<1s,2s> 3n";
param w[A] := read "gams_edgedata.txt" as "<1s,2s> 4n";

defset dminus(v) := {<i,v> in A};
defset dplus(v) := {<v,j> in A};

set s_out := {<"s",v> in A};
set s_in := {<v,"s"> in A};
set t_out := {<"t",v> in A};
set t_in := {<v,"t"> in A};

var x[A] binary;
#var x[A]  real >=0 <=1;
minimize cost: sum<i,j> in A: c[i,j] * x[i,j];

subto fc1: sum<i,v> in s_out: x["s",v] -  sum<v,i> in s_in: x[v,"s"] == 1;
subto fc2: sum<v,i> in t_in: x[v,"t"] -  sum<i,v> in t_out: x["t",v] == 1;
subto fc:
forall <v> in V - {"s","t"}:
sum<i,v> in dminus(v): x[i,v] == sum<v,i> in dplus(v): x[v,i];

subto weightLimit: sum<i,j> in A: w[i,j] * x[i,j]<=5;
```

4.3. Parameter Fine Tuning for the Algorithms

We found the best performing parameters of the metaheuristics after extensive computational experiments. The results are given in Table 2 where the bold ones are the best performing values. For example, for AS we found that ant number (*m)* = 10, β = 100, α = 0.1, and iteration number (*n)* = 50 are the best performing parameter values.

Table 2 – Performing parameters tuples. (Bold ones are the best performing values)

AS	m={1,5,**10,**50,100,200,500,1000} β={1,5,10,50,**100,**200,500,1000} α={0.01,0.05,**0.1,**0.2} n={1,10,**50,**100,200,500,1000}
ACS	m={1,5,**10**,50,100,200,500,1000} β={1,5,10,50,**100**,200,500,1000} α={0.01,0.05,**0.1**,0.2} ρ={0.01,0.05,**0.1**,0.2} q_0={0,0.1,0.5,**0.9**} n={1,10,**50**,100,200,500,1000}
SA	T={**100,**1000} R={**5,**20} a={1.1,**1.5**} b={1.1,**1.5**}
GA	xp={0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,**0.9,**1} mp={0.01,0.05,0.1,0.2,0.4,0.6,0.8,**0.95**} nos={10,20,**25**,50,100} nog={10,**25**,50,100}
MBO	nob={**13**} not={**3**,5,10} non={3,**5**,7} olf={**1**,2,3} $noit$={**25**,50}

During our tests for parameter fine tuning, we observed that the AS and ACS algorithms are constructing paths with significant zigzags. The reason for that is the randomness in the STR of the algorithm. That is, although we guide the ants to select the edges that have less cost and more pheromone, because of the inherent randomness in STR, ants may select other edges. An example is given in Figure 15(a). To avoid these zigzag patterns on the path, we apply a post processing procedure (PPP) (see Figure 15). Let S be the set of discs that the path returned by AS/ACS intersects with. In the PPP, firstly, we set the neutralization cost of the discs in $\mathcal{A}\backslash S$ to a maximum value. Then, under this setting, we

find the shortest path, p_S, using the Dijkstra's algorithm and p_S is returned as the result of the PPP. Even though the PPP returns a path where zigzags are removed, it can return a completely different (but definitely shorter) path than that of AS/ACS (Figure 15(b)). Figure 15 shows the original paths found by AS/ACS algorithm and the paths where PPP takes place. In this figure, the solid path is found by an ant and the dashed path is found by PPP. After a set of computational tests are conducted to reveal the benefits of PPP, we observed that PPP improves AS/ACS up to 3.3%. When its computational cost is considered, we can easily deduce that it is worth to invoke the PPP.

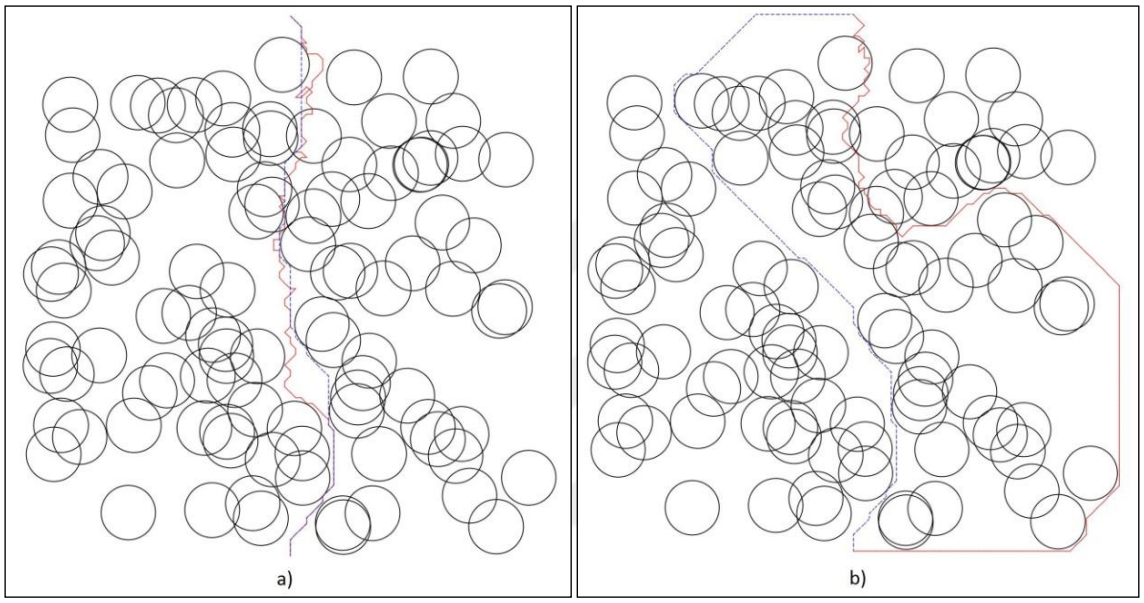

Figure 15 – An example to original paths returned by AS and the paths obtained by PPP. Solid lines represent the path found by AS/ACS and the dashed lines represent the path found by PPP.

1. let S be the set of discs that the path returned by AS/ACS intersects with.
2. set the neutralization cost of the discs in $\mathcal{A}\backslash S$ to a maximum value
3. find the shortest path. p_S, using the Dijkstra's algorithm
4. return p_S

Figure 16 – Post processing procedure (PPP) for AS/ACS

4.4. Results and Discussion

After best parameter sets are selected for AS, ACS, MBO, SA, and GA, computational experiments are conducted on different resolution settings for grids. Results are compared with the SCIP according to cost values and run times as given in Table 3 and Figure 17.

Table 3 – Performance of algorithms on various graph resolutions.

Graph Resolution	AS	ACS	MBO	SA	GA	SCIP
[10x10]	22.67%	**%12.44**	15.36%	13.86%	16.82%	126.74
[20x20]	13.20%	%8.79	2.20%	**2.01%**	5.17%	116.49
[50x50]	9.64%	%7.19	**0.08%**	0.25%	0.58%	112.72
[100x100]	116.06	112.62	**110.44**	110.61	111.18	*

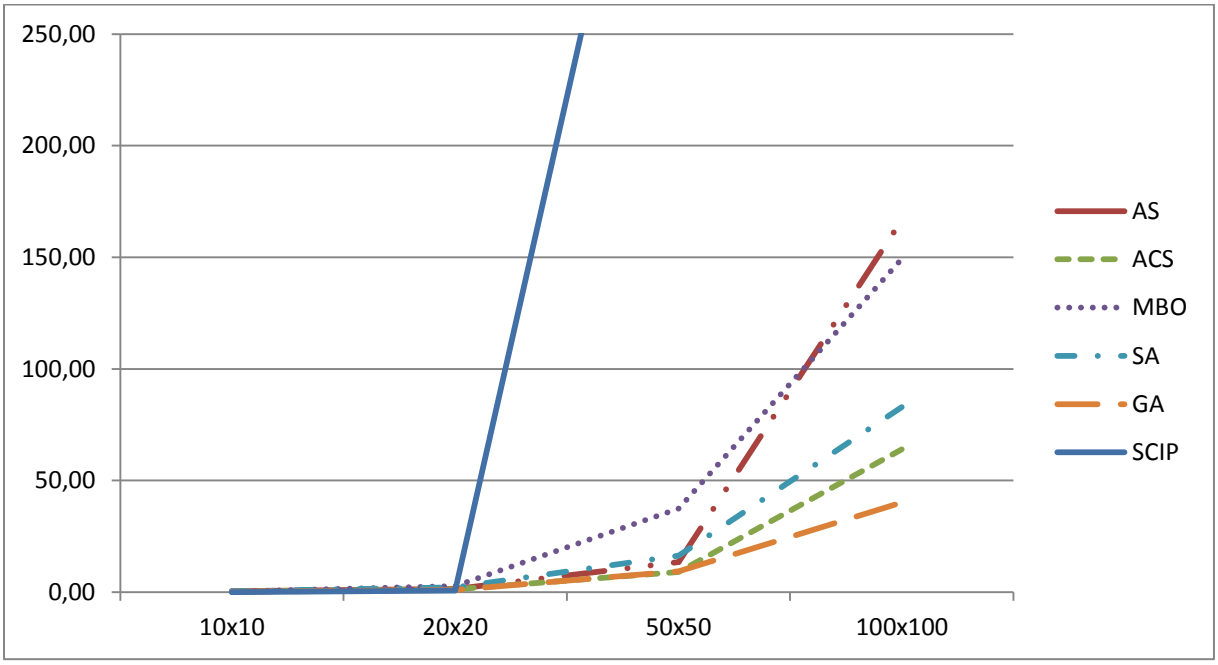

Figure 17 – Run time results of the algorithms.

Regarding Figure 17, even though the exact algorithm solves the smaller graphs in very low run times, it shows an exponential increase as the size of the graph increases. We stopped the SCIP solver whenever its run time exceeds a day (24 hours). Therefore, we cannot give the exact algorithm results for 100x100 resolution graphs.

In Table 3, we can easily observe that the performance of the metaheuristic algorithms improves when the resolution of the graph increases. This is because of the fact that when a metaheuristic chooses a wrong combination for the set of neutralized discs; the path may digress from the optimum path severely which causes larger deviation of total cost. However, if the resolution increases the digressions can be tolerated in a much cheaper way in term of cost.

Observe that the runtime complexity of MBO, SA, and GA present an O(vlogv) behavior where v denotes the number of vertices in a graph. Actually this is due to the fact that all these three algorithms use Dijkstra's algorithm to find the shortest path for a solution.

On the other hand, AS and ACS presents a different run time pattern than the former three algorithms. Since AS and ACS are stochastic algorithms, the time it takes for the ants to reach the target vertex is highly deviational.

According to the results of the computational tests, the performance order of the algorithms is SA, MBO, GA, ACS and AS (from best to worst) where SA and MBO present comparable results. However, the run times of the algorithms are not equal which we believe affects the performance. Therefore, to draw a stronger and fair conclusion we should better design experiments that allow equal run times to the algorithms.

In another set of computational test, we tried to reveal the performance of the algorithms when the number of discs is increased. For this, in addition to the minefield instances with 100 discs each having a radius of 5 on [0,100]x[0,100] rectangle having 100x100 resolution, we created 10 minefields with 200 discs each having a radius of 7 on [0,200]x[0,200] rectangles having 200x200 resolution and 10 minefields with 400 discs each having a radius of 10 on [0,400]x[0,400] rectangles having 400x400 resolution. As stated formerly, to make a fair performance comparison among the metaheuristics, we limit their run time by 65, 200 and 400 seconds for [0,100] x [0,100], [0,200] x [0,200] and [0,400] x [0,400] sized minefields, respectively. In these tests, C=1 and K=5. With this set of computational test, we try to reveal the performance of the algorithms on various sized graphs when they are given equal run times. Results are given in Table 4.

Table 4 – Performance of algorithms on various graph sizes (C=1 and K=5).

Minefield Size	AS	ACS	MBO	SA	GA
[100x100]	112.55	113.51	**110.79**	110.86	111.37
[200x200]	225.36	226.58	**223.36**	224.62	223.93
[400x400]	515.87	505.39	458.74	**450.47**	452.35

Table 4 presents the costs of paths found by the algorithms. Each figure in the table is an average of 10 different minefields. We can easily observe that the AS and ACS present the worst performance and their performance gets worse when the graph size increases. This is due to the fact that in the design of AS and ACS paths are constructed by selecting edges whereas in the other algorithms paths are built by using discs. However, number of edges increase much faster than number of discs.

On the other hand, when we try to compare the MBO, SA and GA, even though MBO looks like the winner, the presented results are not adequate to draw a strong conclusion about

the winner. Therefore, we conducted t-tests as a further analysis. A t-test returns the probability associated with a Student's t-Test. We use t-test to determine whether two samples are likely to have come from the same underlying population that have the same mean. We applied t-tests among the pairs of SA, GA and MBO algorithms, where the tests did not find a significant difference in terms of the best performances. For example, when the performance values of SA and GA on 100x100 grid are subject to t-test, we obtained 43% probability whereas t-tests among other pairs were very similar to this one. As a result, we can conclude that the metaheuristics present comparable results for the ONP.

To summarize, the proposed solution techniques perform well on small and moderate sized graphs with a small deviation from optimum. The exact method used to find the exact solutions runs faster than our proposed solution techniques which can be observed as a drawback for our algorithms. However, instances in real life are of large sized for which the exact method cannot be applied in practice. Therefore, our solution techniques are important contributions. Additionally, GA, SA and MBO outperform AS and ACS.

5. CONCLUSION

In this study, we tackle a path planning problem called obstacle neutralization problem (ONP) where the aim is to safely and swiftly traverse an agent from a given start point to a target point through a plan of disc-shaped obstacles in the plane. This agent can neutralize limited number of discs but neutralization of a disc causes some cost to the traversal length of the path. The ONP is of vital importance because of its real applications in real life. In a military scenario, the objective is to navigate a combat unit safely and swiftly through a coastal environment with mine threats and to reach the target location as fast as possible where the mine data is given to the combat unit in advance by an airborne mine detection tactical system. In another scenario, a merchant ship navigating in an icy region has the capability of cracking (neutralizing) the ice blocks and tries to reach the destination in minimum time. In a different scenario, a military aircraft tries to navigate on a safe path by neutralizing danger zones. In all scenarios, the unit/ship/aircraft has a neutralization capability limited by K, which makes the problem NP-Complete.

In the literature, ONP is tried to be solved under the assumption of equal radii of the discs, and equal neutralization cost of the discs which may not be realistic in many cases. In order to solve this NP-complete problem, we customized and applied four metaheuristic algorithms which can work with obstacles having different radii and neutralization costs. Therefore, one of the contributions of this study is developing algorithms that can solve ONP instances having various radii and neutralization costs. The algorithms implemented in this study are ant system (AS), ant colony system (ACS), genetic algorithm (GA), simulated annealing (SA), and migrating birds optimization (MBO) algorithms.

We provide computational experiments to empirically asses the performance of the metaheuristics. The performance of the algorithms are tested both on real-world and synthetic data. Their results are also compared with exact solutions on small and moderate instances of ONP and it is shown that the customized metaheuristics present near-optimal results (as low as 0.08% away from optimum) in reasonable execution times. Therefore, second contribution of this study is that the proposed metaheuristics present very close results on small and moderate sized graphs and therefore can be used on large graphs. Third contribution is proposing better solution techniques (GA, SA and MBO) that outperform the AS and ACS. As a result of this, we can conclude that the techniques based on disc selection present better results than the techniques based on edge selection.

In a closely related and well-studied scenario, the status of the obstacles is not known a priori, instead of the neutralization capability the agent has a disambiguation capability which can be used only if the agent reaches the boundary of an obstacle (Priebe et al., 2005; Aksakalli et al., 2011). The problem is therefore called as random disambiguation paths (RDP) in which the agent has a limited number of disambiguation. An interesting and tough future work would be trying to solve the ONP and RDP problems concurrently where the agent has both neutralization and disambiguation capabilities. As another future work, the techniques developed in this study may also be applied to other similar application domains. For example, the problem of finding the path for a merchant ship navigating through an icy region can be solved using the techniques developed in this thesis. Another application field that the proposed algorithms are expected to perform well is the problem of determining the route for an aircraft that can neutralize danger zones. In this thesis, the proposed metaheuristics are applied in discrete space. Therefore, as another future work, the algorithms can be tested in continuous space. Furthermore, extending the comparison study by including several other metaheuristics such as particle swarm optimization and differential evolution algorithms would contribute to the literature.

6. REFERENCES

Aksakalli, V., Fishkind, D., Priebe, C., & Ye, X. (2011). The reset disambiguation policy for navigating stochastic obstacle fields. Naval Research Logistics, 58 , 389–399.

Algin, R., Alkaya, A., Aksakalli, V., & Oz, D. (2013). An ant system algorithm for the neutralization problem. In Advances in Computational Intelligence (pp. 53–61). Volume 7903.

Alkaya, A., Aksakalli, V., & Periebe, C. (2014). A penalty search algorithm for the obstacle neutralization problem. Computers and Operations Research, . doi:10.1016/j.cor.2014.08.013.

Alkaya, A., & Oz, D. (2016). An exact algorithm for the obstacle neutralization problem. (Under second revision).

Beasley, D., Bull, D., & Martin, R. (1993). An overview of genetic algorithms: Part i, fundamentals. University Computing, 15 , 58–69.

Bekker, J., & Schmid, J. (2006). Planning the safe transit of a ship through a mapped minefield. Journal of the Operations Research Society of South Africa, 22 , 1–18.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys, 35 , 268–308.

Carlyle, W., Royset, J., & Wood, R. (2008). Lagrangian relaxation and enumeration for solving constrained shortest-path problems. Networks, 52 , 256–270.

Dahl, G., & Realfsen, B. (1997). Curve approximation and constrained shortest path problems. In International Symposium on Mathematical Programming.

Dahl, G., & Realfsen, B. (2000). Curve approximation constrained shortest path problems. Networks, 36 , 1–8.

Dorigo, M. (1992). Optimization, learning and natural algorithms. Ph.D. thesis Politecnico di Milano, Italy.

Dorigo, M. & Gambardella, L. (1997). Ant colony system: A cooperative learning approach to the travelling salesman problem. IEEE transaction on evolutionary computation, vol. 1, no. 1, pp. 53–66.

Duman, E., Uysal, M., & Alkaya, A. (2012). Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem. Information Sciences, 217 , 65–77.

Dumitrescu, I., & Boland, N. (2001). Algorithms for the weight constrained shortest path problem. International Transactions in Operational Research, 8 , 15–29.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 13 , 533–549.

Guo, L., & Matta, I. (2003). Search space reduction in qos routing. Computer Networks, 41 , 73–88.

Handler, G., & Zang, I. (1980). A dual algorithm for the constrained shortest path problem. Networks, 10 , 293–309.

Holland, J. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Holland, J. (1986). Escaping brittleness: the possibilities of general purpose learning algorithms applied to parallel rule-based system. Machine learning, (pp. 593–623).

Jüttner, A., Szviatovski, B., Mcs, I., & Rajk, Z. (2001). Lagrange relaxation based method for the qos routing problem. In Proceedings of 20th Annual Joint Conference of the IEEE Computer Communications Societies (pp. 859–868). volume 2.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. J Global Optim, 39 , 459–171.

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simulated annealing. Science, 220 , 671–680.

Kuipers, F., Korkmaz, T., Krunz, M., & Mieghem, P. V. (2004). Performance evaluation of constraintbased path selection algorithms. IEEE Network, 18 , 16–23.

Latourell, J., Wallet, B., & Copeland, B. (1998). Genetic algorithm to solve constrained routing problem with applications for cruise missile routing. In Proceedings of SPIE 3390 (pp. 490–500).

Lee, S. (1995). Route optimization model for strike aircraft . Master's thesis Naval Postgraduate School Monterey, California.

Li, P.-C. (2009). Planning the optimal transit for a ship through a mapped minefield. Master's thesis Naval Postgraduate School Monterey, California.

Nygaard, R., HusZy, J., & Haugland, D. (1998). Compression of image contours using combinatorial optimization. In Proceedings of the International Conference on Image Processing-ICIP98 1 (pp. 266–270).

Priebe, C., Fishkind, D., Abrams, L., & Piatko, C. (2005). Random disambiguation paths for traversing a mapped hazard field. Naval Research Logistics, 52 , 285–292.

Reeves, D., & Salama, H. (2000). A distributed algorithm for delay-constrained unicast routing. IEEE/ACM Transactions on Networking, 8 , 239–250.

Royset, J., Carlyle, W., & Wood, R. (2009). Routing military aircraft with a constrained shortest-path algorithm. Military Operations Research, 14 , 31–52.

Storn, R., & Price, K. (1997). Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11 , 341–359.

Witherspoon, N., Holloway, J., Davis, K., Miller, R., & Dubey, A. (1995). The coastal battlefield reconnaissance and analysis (cobra) program for minefield detection. In

Proceedings of the SPIE: Detection Technologies for Mines and Minelike Targets 2496 (pp. 500–508).

Zabarankin, M., Uryasev, S., & Murphey, R. (2006). Aircraft routing under the risk of detection. Naval Research Logistics, 53 , 728–747.

Zabarankin, M., Uryasev, S., & Pardalos, P. (2002). Optimal risk path algorithms. Cooperative control and optimization (r. murphey and p. pardalos ed.). Kluwer Academic, Dordrecht , (pp. 271–303).

www.http://scip.zib.de/

www.http://zimpl.zib.de/

RESUME

RAMAZAN ALGIN
Marmara University, Goztepe Campus
Technology Faculty, Computer Engineering Dept.
Room: D405, PK:34722, Kadikoy, Istanbul, Turkiye
Phone (Cell)+90-555-719-1316
e-mail: ramazan.algin@marmara.edu.tr, algin.ramazan@gmail.com

EDUCATION

M.S., Computer Science and Engineering	Marmara University, Faculty of Engineering, Istanbul, Turkey, (2013- 2016) GPA:3.69 Thesis: Metaheuristic Methods for the Obstacle Neutralization Problem Advisor: Asst. Prof. Dr. Ali Fuat Alkaya
B.S., Computer Science and Engineering	Marmara University, Faculty of Engineering, Istanbul, Turkey, (2009-2013) GPA:3.02
B.S., Mathematics	Middle East Technical University, Ankara, Turkey, (2007-2009)

WORK EXPERIENCE

2014-…	Research Assistant, in Department of Computer Engineering, Technology Faculty, Marmara University, Istanbul, Turkey.

FOREIGN LANGUAGES and EXAM SCORES

English (good), German (Beginning)

ALES: 90

IELTS: 6.5

GRE: 166 (Quantitative)

PUBLICATIONS

- Alkaya, A.F. Algin R.: "Metaheuristic Based Solution Approaches for the Obstacle Neutralization Problem", Expert Systems with Applications, (2015), 42 (3), 1094-1105
- Algin, R. & Alkaya, A. F. "Solving the Obstacle Neutralization Problem Using Swarm Intelligence Algorithms." The Seventh International Conference on Soft Computing and Pattern Recognition (SoCPaR 2015). Fukuoka, Japan, (2015).
- A. F. Alkaya, R. Algin, D. Oz, V. Aksakalli and E. Ceyhan. "Performance Evaluation of an Exact Method for the Obstacle Neutralization Problem." Proceedings of the Sixth International Conference on Industrial Engineering and Operations Management, Kuala Lumpur, Malaysia, March 8-10, 2016.

- Algin, R., Alkaya, A. F., Aksakalli, V.: "Performance Comparison of Metaheuristics for the Obstacle Neutralization Problem", CTW2015 (The Cologne-Twente Workshop on Graphs and Combinatorial Optimization), Istanbul, Turkiye, May 26-28, (2015).
- Alkaya, A.F.; Algin. R.; Sahin, Y.; Agaoglu, M.; Aksakalli, V.: "Performance of Migrating Birds Optimization Algorithm on Continuous Functions", Proceedings of ICSI2014, Part II, LNCS 8795, (2014), 452-459.
- Algin, R., Alkaya, A.F., Aksakalli, V., Öz, D.: "An Ant System Algorithm for the Neutralization Problem", Proceedings of IWANN 2013, Part II, LNCS 7903, (2013), 53-61.

PROJECTS

TUBITAK 3001	Development and application of an exact solution method and metaheuristics to the neutralization problem. (2014-2016)
Marmara University Bapko Project	Application and comparison of metaheuristic based solution methods for a generalization of the traveling salesman problem (2014-2016)
TUBITAK 1001	Graph theory based ship navigation modeling and ship anchoring field optimization. (2013 - 2014)

CONFERENCES

Organization Committee for **CTW2015**

AWARDS

Best B.S. Thesis in Computer Science and Engineering Department in 2013.

RESEARCH INTERESTS

Optimization, heuristics and meta-heuristics, combinatorial optimization problems, obstacle neutralization problem, algorithm design and analysis.