JOURNAL OF INDUSTRIAL AND
MANAGEMENT OPTIMIZATION
Volume **13**, Number **2**, April **2017**

doi:10.3934/jimo.2016049

pp. **835–856**

AN OPTIMAL ALGORITHM FOR THE OBSTACLE NEUTRALIZATION PROBLEM

Ali Fuat Alkaya

Computer Engineering Department, Marmara University
Istanbul, 34722, Turkey

Dindar Oz

Software Engineering Department, Yasar University
Izmir, 35100, Turkey

(Communicated by Chao Xu)

Abstract. In this study, an optimal algorithm is presented for the obstacle neutralization problem (ONP). ONP is a recently introduced path planning problem wherein an agent needs to swiftly navigate from a source to a destination through an arrangement of obstacles in the plane. The agent has a limited neutralization capability in the sense that the agent can safely pass through an obstacle upon neutralization at a cost added to the traversal length. The goal of an agent is to find the sequence of obstacles to be neutralized en route minimizing the overall traversal length subject to the neutralization limit. Our optimal algorithm consists of two phases. In the first phase an upper bound of the problem is obtained using a suboptimal algorithm. In the second phase, 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 algorithm is presented with computational experiments conducted both on real and synthetic naval minefield data. Results are promising in the sense that the proposed method can be applied in online applications.

1. **Introduction.** In this study, a path planning problem is considered where the goal is to safely and swiftly navigate an agent from a given source location to a destination through an arrangement of potential mine or threat discs in the plane. The agent is given a neutralization capability with which the agent can safely traverse through the disc after neutralizing it for a cost added to its traversal length. However, number of allowed neutralizations is limited, say by K, due to a particular reason such as the payload capacity of the agent or vehicle and the central issue is how to direct the agent to optimally utilize this neutralization capability. This problem, called obstacle neutralization problem (ONP), is a recently introduced NP-Hard problem [7]. Note that, if the number of allowed neutralizations is infinite, then the problem turns out to be the classical (unconstrained) shortest path problem and can be easily solved.

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

2010 *Mathematics Subject Classification.* Primary: 68U01, 49M25; Secondary: 97K30.
Key words and phrases. Obstacle neutralization problem, combinatorial optimization, optimal algorithm, path planning, graph theory.

as fast as possible where the mine data is given to the combat unit in advance by an airborne mine detection tactical system [34, 10]. This unit has a neutralization capability limited by K. 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. Therefore, the main research question for ONP is how to use this neutralization capability so the optimum path is followed. Furthermore, ONP is also closely related to the problems undertaken in several other real world applications in diverse fields such as telecommunications routing, curve approximations, scheduling and minimum-risk routing of military vehicles and aircraft.

In order to formally define the problem, the following definitions are needed. Let s and t be two points in $\mathbb{R}^2$, D be a finite set of open discs in $\mathbb{R}^2$, C be a constant in $\mathbb{R}_{\geq 0}$, and K be a given constant in $\mathbb{Z}_{\geq 0}$. For simplicity and convenience, an instance of ONP will be denoted by a tuple $(s,\ t,\ D,\ C,\ K)$. An agent wants to traverse from s to t through $\mathbb{R}^2$, along a continuous curve that is as short as possible in the sense of arclength. The traversing agent cannot enter discs which are obstacles or considered to be obstacles but, if and when the agent is located at the boundary ∂d for any $d \in D$, the agent has the option to neutralize the disc d at a cost C added to the traversal arclength. Thus, only after neutralizing a disc the agent can enter the disc and the agent has K allowed number of neutralizations. The objective is to direct the agents traversal to optimally use this neutralization capability; that is, to find a policy for the agent which minimizes the total cost of the agents s, t traversal. Throughout this study, it is assumed all disks have the same neutralization cost and the same radius.

It can be easily observed that the neutralization problem is a combinatorial optimization problem where among D discs ($|D| \geq K$), the agent must choose at most K of them to neutralize.

An example to the neutralization problem is shown in Figure 1. Suppose the positions of s and t are (50,100) and (50,1), respectively. Each disk has a radius of 10 and 0.5 for the neutralization cost. In the figure, the optimal paths for $K = 0$, 1, 2 and 3 are depicted with bold, dashed, dotted and solid lines, respectively. Note that it is not always the case that all K allowable neutralizations are used.

There are two studies directly related with the ONP where heuristic methods are proposed and used for solving it. In [7], the authors propose a penalty based algorithm to find suboptimal solutions for the ONP. On the other hand, in [4], an ant system algorithm is adapted to solve the ONP. Being heuristic based methods, both of those studies propose suboptimal algorithms that are not guaranteed to be optimal. Hence, the literature is lack of an optimal algorithm that can solve the ONP instances. Therefore, a fast and robust optimal algorithm that can be used online is of vital importance in the scenarios listed above. Our contribution in this study is two-fold: (1) an effective optimal algorithm for the ONP is presented, (2) unlike the state-of-the-art studies where the underlying graph is discretized in order to simplify the problem, the proposed algorithm works both in continuous and discretized graphs.

Proposed algorithm consists of two phases. In phase I, a recently proposed algorithm is used to obtain a suboptimal solution. In the second phase, a k-th shortest path algorithm is used to close any gaps. The correctness of the algorithm is proven theoretically and computational tests are conducted both on discrete lattice and

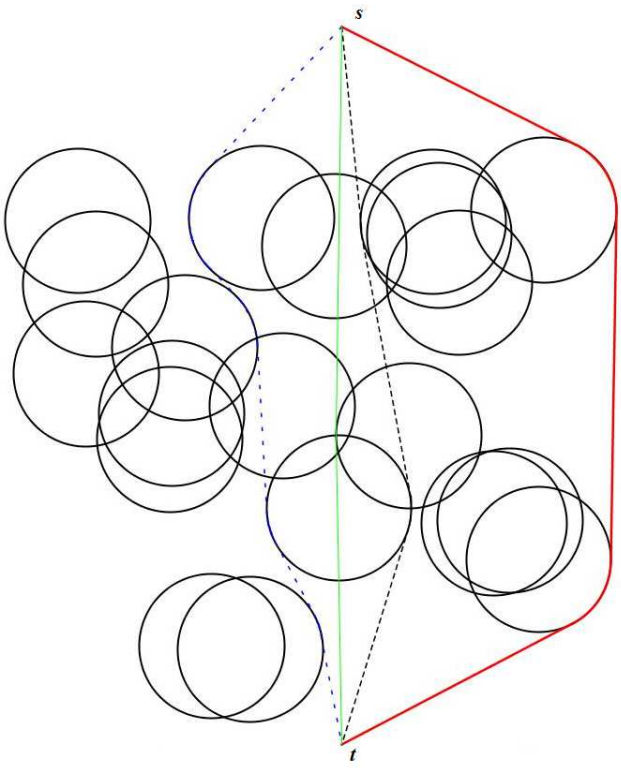

FIGURE 1. An example to the obstacle neutralization problem and optimal paths for $K = 0$, 1, 2 and 3

continuous graphs to reveal its performance. Furthermore, its results and performance are compared with an IP solver. Results show that the proposed algorithm performs well in terms of running time on continuous graphs and it can be used in online applications.

The rest of this manuscript is organized as follows: Section 2 gives a comprehensive overview of current state-of-the-art in ONP related research. Section 3 provides details of proposed algorithm, discusses its properties, and proves its optimality. Section 4 describes the graph modelling including the tangent arc graphs and the proof that the optimum path resides in the tangent arc graph. Section 5 presents real and random minefield data, computational experiments, and Section 6 summarizes and concludes our work.

2. **ONP and related work.** For a given graph $G = (V, E)$, let c_{ij} and w_{ij} denote the cost and weight of the edge $(i, j) \in E$ respectively. The objective in ONP is to find a minimum cost path from a start node s to a destination node t in V such that the sum of weights of the path edges is at most a given limit K.

Let the binary variable $x_{ij} = 1$ if the edge (i, j) is part of the optimal path and 0 otherwise. The integer programming formulation of the ONP is as follows:

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

s.t.

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

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

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

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

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in \{1,\ldots,|V|\} \tag{6}$$

The first constraint imposes that the difference in the number of edges leaving s and entering into s is 1. The second constraint states that the difference in the number of edges entering t and leaving t is 1. The third constraint states that for all other nodes except s and t, the number of edges leaving them is equal to the number of edges entering them. The fourth constraint enforces the total weight limit.

ONP is a recently introduced problem. However, in the literature, there are several problems which are closely related with ONP such as routing of signals in telecommunication networks with QoS guarantees. Delay sensitive applications such as real-time video and voice, require traffic to be received at the destination within a delay constraint. On the other hand, it is highly desirable to reduce the path cost, either in terms of money or cost of using network resources. In the literature, that problem is studied under the name Delay Constrained Least Cost Path (DCLC) problem. Juittner et al. [17] used Lagrangian relaxation techniques to solve it. The algorithm, called Lagrange Relaxation Based Aggregated Cost (LARAC) Algorithm, uses the concept of aggregated costs and provides an efficient method to find the optimal multiplier based on Lagrangian relaxation. Reeves and Salama [29] study the DCLC problem and propose a simple distributed heuristic method called the Delay-Constrained Unicast Routing (DCUR) Algorithm. The DCUR algorithm starts by building the least delay (LD) and least cost (LC) paths from each node to destination t. The path is constructed one node at a time from the source to destination. While building the path, DCUR chooses edges on LC path if traveling on that edge to the destination guarantees a permissible amount of delay by taking into account the delay accumulated thus far. Otherwise the algorithm chooses edges on the LD path. Guo and Matta [15] proposed another method for the DCLC problem, called the Delay Cost Constrained Routing (DCCR) Algorithm. In order to reduce the search space exploited by DCCR, the authors used a variant of the Lagrangian relaxation method proposed in [16]. The aforementioned heuristics along with six other ones for DCLC are compared in [19]. This study compares these algorithms on their path cost and computational complexity. Results of simulations on square lattices suggest the DCUR algorithm provides superior results in general.

Another closely related topic is minimum risk routing of military vehicles and aircraft in a threat environment where for a given risk function the edge lengths are assigned as probability of being detected by radars or surface air missiles. In this problem, the aim is to minimize the total risk on the path taken from source to destination that satisfies a given constraint due to supply of fuel or flight time. The following are some of the studies on minimum risk routing of military vehicles.

Lee [21] worked on a search space which is discretized into a three dimensional grid. The constraint of the problem is a given maximum fuel consumption. Similar to [16], the lagrangian dual of the problem is used where in order to find the best lagrange multiplier value, bisection method is used. Latourell et al. [20] used genetic algorithms to optimize minimum risk routing of military vehicles. In their model, the authors considered the limitation on the sharpness of the turns that the aircraft can make. In [24] an ant system based path planning algorithm is proposed for the unmanned air combat vehicles where the objective is to reach the destination with minimum probability of being found and minimum consumed energy. In another study, biologically inspired two level method is proposed for real-time path planning in a complex and dynamic environment, employable in ground vehicles [23].

Zabarankin et al. [37] studied both analytical and discrete optimization approaches. They have shown the optimum can be reached when there is only one radar in a continuous setting. For discrete setting they use the algorithm proposed in [13]. In a further study, they extended their work and develop a model where the trajectory of an aircraft in a three-dimensional setting is determined [36]. Royset et al. [30] applied the algorithm in [9] to route planning problem for various type of military aircraft. Muhandiramge et al. [25] contributed the literature by giving a general scheme for convergence of network discretizations. Robot navigation is also a very similar topic to routing of vehicles. In [38], a novel and effective algorithm is proposed for dynamic unknown environments for robot navigation. Yang et al. [32] designed a distributed control algorithm for target search problem of multiple robots. In their work, they adapted the behaviour of bacteria to trap their target and use voronoi diagram to divide a large search space into small voronoi cells. The proposed BC algorithm is shown to be comparable with the state-of-the-art methods.

Curve approximation is another related application to ONP. Piecewise linear functions are often used to approximate complex curves in a number of domains including computer aided design, computer graphics, image processing, and mathematical programming [11]. Such an approximation, on the other hand, often needs to be made in the presence of transmission rate or storage constraints. There are two variants of this problem: (1) minimizing the approximation error subject to breakpoint constraints, and (2) minimizing the number of breakpoints given an approximation error bound. Dahl and Realfsen [12] aimed to minimize the approximation error and studied four different algorithms: a combinatorial algorithm, Lagrangian relaxation based algorithm, a dynamic programming algorithm and a linear programming algorithm. Nygaard et al. [26] studied the same problem and they use dynamic programming approach for solving the problem.

In another problem very similar to 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. Bekker and Schmid [8]

formulated a fitness function which favors shorter paths with a fewer number of neutralized discs and use genetic algorithms to tackle the problem. Li [22] developed a mission-planning tool to find a minimum-risk route for a surface ship through a mapped minefield. The author also proposed a greedy heuristic to form a prioritized list of mines to be cleared (neutralized) so that a zero-risk path is obtained. However, in the problem definition the author assumes that a minesweeper is sent before the ship and it can reach and clear any mine in any sequence, safely.

Several metaheuristic algorithms are tailored in [6, 5] in order to solve Obstacle Neutralization Problem. They applied, namely, ant colony optimization, genetic algorithm and migrating bird optimization methods. Although the solutions were suboptimal, they achieved to get very close results to optimal in reasonable time.

State-of-the-art algorithms proposed for the aforementioned closely related problems either require significant computational resources or demonstrate poor performance on the neutralization problem. Furthermore, most of them propose heuristic solutions with no guarantee of optimality and only a few of the work on continuous graphs. As introduced in the next section, an optimal algorithm for the obstacle neutralization problem is developed that can also be used for related applications.

3. **Proposed optimal algorithm.** This section introduces the proposed optimal algorithm for ONP. The algorithm consists of two phases. In the first phase, a suboptimal algorithm, named Penalty Search Algorithm (PSA) is called. PSA is explained in more detail in [7] but here a basic explanation of it is provided for the sake of completeness. If the path returned by PSA has exactly K neutralizations, then the algorithm returns this path since it is proven that when PSA dictates exactly K neutralizations, then the path returned by PSA is indeed optimal [7]. However, PSA cannot always find a path with exactly K neutralizations and returns a path with less number of neutralizations than available (thus an upper bound solution). Should this be the case, proposed optimal algorithm initiates a second phase in order to close this gap. In the second phase, starting from the upper bound given by PSA and using the edge costs that PSA adjusted, a k-th shortest path algorithm (kSPA) is used to find the optimal path. A kSPA not only determines the shortest path, but also the second shortest, the third shortest, and so up to the k-th shortest path. Therefore, using kSPA, one can eventually find the optimal solution to the ONP. In the literature there are essentially two main contributions on kSPAs and in this study it is preferred to use the one proposed in [35] where paths are required to be loop free.

Recall that the optimum path may have K or fewer than the maximum allowable number of neutralizations. If second phase of the optimal algorithm returns a path with less than K neutralizations, its optimality must be guaranteed. In the following subsections, the details of the phases of the optimal algorithm is presented and its optimality is shown. But first, the notation that characterizes the model is presented.

- $G = (V, E)$ denotes the graph representing the obstacle field and the tuple (s, t, D, C, K) represents an ONP instance
- p^* denotes the optimal path for the ONP instance (s, t, D, C, K)
- $\hat{p}^*$ denotes the best path found by PSA
- For $e \in E$, $\ell(e)$ denotes its Euclidean length
- For $e \in E$ and $d \in D$, $I(e, d) = 1$ if e enters and exits d at once, 0.5 if e enters d but does not exit d, 0 if e does not enter d

- For $e \in E$, $\theta(e)$ is the number of disks that edge e intersects. If e enters a disc but does not exit than it is counted as a half. That is, $\theta(e) := \sum_{d \in D} I(e, d)$. $\theta(null)$ is considered to be zero
- For $e \in E$, its associated neutralization cost is defined as $c(e) := C \times \theta(e) = C \times \sum_{d \in D} I(e, d)$
- For $e \in E$, its total cost associated with α is denoted by $\tau(e, \alpha) := \ell(e) + \alpha \times c(e)$

For a path p in G :

- $\ell(p) := \sum_{e \in p} \ell(e)$, $\theta(p) := \sum_{e \in p} \theta(e)$, and $c(p) := \sum_{e \in p} c(e)$. Observe that $\theta(p)$ is the number of disks p intersects. Thus, an agent following path p would neutralize exactly $\theta(p)$ obstacle disks
- $\tau(p, \alpha) := \sum_{e \in p} \tau(e, \alpha) = \sum_{e \in p} \ell(e) + \alpha \times c(e) = \ell(p) + \alpha \times c(p)$. Note that $c(p) = C \times \theta(p)$ and $\tau(p, \alpha) = \ell(p) + \alpha \times C \times \theta(p)$. $\tau(null, \alpha)$ is considered to be infinite
- $\alpha_1 = 1 < \alpha_2 < \ldots < \alpha_n$ denotes a sequence of real-valued α values

From these definitions, one can easily obtain the following equation-which will be used frequently in the following subsections.

$$\tau(p, 1) = \tau(p, \alpha) - \theta(p) \times C \times (\alpha - 1) \tag{7}$$

3.1. **Phase I: PSA.** PSA is based on the following simple idea: find the largest penalty term $\alpha^* \geq 1$ such that, the unconstrained shortest path (i.e., the path without any neutralization limits) with Euclidean length of disk-intersecting edges augmented by $(\alpha^* \times C \times \theta(e))$ requires the highest number of neutralizations without exceeding K, hence the name "penalty search". This is the path returned by PSA and it clearly satisfies the neutralization limit constraint. The search for the penalty term is found by a straightforward bisection method [7].

The correctness of the PSA algorithm resides on the fact that as α is increased, the number of neutralizations dictated by PSA decreases and the total cost increases monotonically. Using this fact, the authors developed an efficient mechanism for finding α^* using a bisection approach. PSA is presented in Figure 2 where comments are given in C language style. In the figure, Dijkstra(α) denotes execution of the Dijkstra's Algorithm for finding the unconstrained optimal $s - t$ path using $\tau(e, \alpha)$ as the edge costs. PSA starts by setting $\alpha = 1$. If Dijkstra($\alpha = 1$) returns a path with neutralizations less than or equal to K, then PSA terminates with the current path. Otherwise, the first while loop finds a coarse interval $[\alpha_{min}, \alpha_{max}]$ such that $\alpha_{min} \leq \alpha^* \leq \alpha_{max}$. The second while loop takes a bisection approach to fine-tune α_{min} and α_{max} until $\alpha_{max} - \alpha_{min} \leq \varepsilon$ where ε is a pre-specified tolerance parameter. It is important to note that in the underlying graph, it is assumed that there is always a walk around path with 0 neutralizations that PSA returns in case it can not find a better one. In this way, the PSA is always guaranteed to return a valid path.

Most important property of PSA is that if $\theta(\hat{p}^*) = K$, that is, if the best path found by PSA requires exactly K neutralizations, then $\hat{p}^*$ is indeed the optimum path for the ONP instance (s, t, D, C, K). However, as mentioned earlier, PSA is not guaranteed to result in a path with exactly K neutralizations [7]. Should this be the case, a second phase is initiated to close the gap for finding the optimum path as discussed in the next section.

Input: ONP instance (s, t, D, C, K)
Output: The shortest unconstrained path p with the largest $\theta(p)$ such that $\theta(p) \leq K$
1. $\alpha = 1$
2. p = Dijkstra(α) *// run Dijkstra on G*
3. **if** ($\theta(p) \leq K$) return (p, α) *// If p is feasible return p and α*
4. **while** ($\theta(p) > K$)
5. $\alpha_{min} = \alpha$
6. $\alpha = \alpha \times 10$
7. update G as edge cost = $\tau(e, \alpha)$
8. p = Dijkstra(α)
9. **if** ($\theta(p) == K$) return (p, α)
10. **end while**
11. $\alpha_{max} = \alpha$
12. **while** (($\alpha_{max} - \alpha_{min}$) $> \varepsilon$ or $\theta(p) \geq K$))
13. $\alpha = (\alpha_{max} - \alpha_{min})/2$
14. update G as edge cost = $\tau(e, \alpha)$
15. p = Dijkstra(α)
16. **if** ($\theta(p) \leq K$) *// feasible path*
17. $\alpha_{max} = \alpha$ *// update upper bound for α*
18. $\hat{p}^* = p$ *// store last found feasible path in* $\hat{p}^*$
19. $\alpha^* = \alpha_{max}$
20. **else**
21. $\alpha_{min} = \alpha$ *// update lower bound for α*
22. **end if**
23. **if** ($\theta(p) == K$) return (p, α)
24. **end while**
25. return $(\hat{p}^*, \alpha)$ *// No solution with K neutro is found. return last feasible path*

FIGURE 2. PSA

3.2. **Phase II: kSPA.** There is a drawback of PSA that requires attention. Specifically, in some cases, PSA cannot find a path with exactly K neutralizations regardless of how fine α is tuned. The reason why PSA can not find the optimum is explained with examples in [7] and we direct the reader to there.

At this point PSA is just ended and kSPA is starting. Recall that PSA returns two pieces of information: its best path, $\hat{p}^*$, and α^* which means that when the cost of edges intersecting discs is set to $\alpha^* \times C \times \theta(e)$, then $\hat{p}^*$ is the shortest path. Using this adjusted graph, kSPA will start to find the optimal path with K or less number of neutralizations.

Before proceeding with the proposed optimal algorithm, it should be noted that the path returned by kSPA satisfying the maximum allowed number of neutralizations constraint might not necessarily be the optimum path. The example depicted in Figure 3 explains the situation. In the figure, the dashed path (p_1) is the optimum path of PSA's modified graph wherein the neutralization cost is modified with $\alpha = 1.307$. The dotted path (p_2) is one of the feasible paths longer than p_1 on the modified graph. The solid path (p_3) is longer than the other two paths on the modified graph and thus will be reported latterly by kSPA. However, p_3 is the optimum path for this (s, t, D, 1, 4) instance as it is proven below.

As it is demonstrated with the above example, a well designed stopping criterion is needed to terminate the kSPA as well as the proof that shows the path obtained with that stopping criterion is the optimum path. To show this, some preliminary arguments are needed.

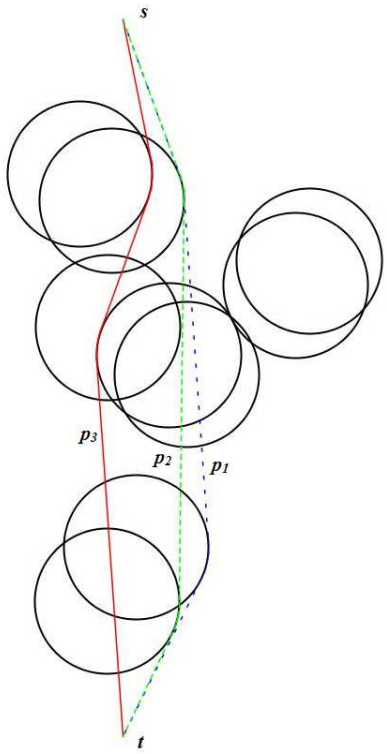

FIGURE 3. An example that depicts the case where any path returned by kSPA satisfying the maximum allowed number of neutralizations constraint may not necessarily be the optimum path.

The kSPA is designed to give the next shortest path at each iteration. That is $\tau(p_i, \alpha^*) \leq \tau(p_j, \alpha^*)$ for $i < j$, where p_i is the i-th shortest path given by kSPA (Note that $p_1 = \hat{p}^*$).

However, as it is demonstrated above, since kSPA is working in the adjusted graph, the next path returned by kSPA at i-th iteration, say p_i, is a better (thus a candidate for optimal) solution only if it satisfies the neutralization limit constraint ($\theta(p_i) \leq K$) and it has a less total cost than the current best candidate path in the original graph ($\tau(p_i, 1) < \tau(p_{cand_{i-1}}, 1)$). Mathematically,

$$p_{cand_i} \leftarrow \begin{cases} p_i, & \text{if } \theta(p_i) \leq K \text{ and } \tau(p_i, 1) < \tau(p_{cand_{i-1}}, 1) \\ p_{cand_{i-1}}, & \text{otherwise, where } p_{cand_0}, \text{ is defined as } \textit{null}. \end{cases} \tag{8}$$

Hence, the following two propositions are obtained easily from (8), where the proofs are trivial.

Proposition 1. $\tau(p_{cand_i}, 1) \geq \tau(p_{cand_j}, 1)$ *for* $i < j$.

Proposition 2. $\tau(p_i, \alpha^*) \geq \tau(p_{cand_i}, \alpha^*)$.

Rule (8) guarantees that each candidate path has K or fewer number of neutralizations on it. A stronger claim is that the number of neutralizations on candidate paths increase monotonically. This is proven in the following corollary.

Corollary 1. $\theta(p_{cand_i}) \leq \theta(p_{cand_j})$ *for* $i < j$.

Proof. Using equation (8) it can be easily seen that

$$\tau(p_{cand_i}, \alpha^*) \leq \tau(p_{cand_j}, \alpha^*) \text{ for } i < j.$$

On the other hand

$$\tau(p_{cand_i}, 1) \geq \tau(p_{cand_j}, 1), \text{ by Proposition 1.}$$

Using equation (7), one can obtain

$$\tau(p_{cand_i}, \alpha^*) - \theta(p_{cand_i}) \times C \times (\alpha^* - 1) \geq \tau(p_{cand_j}, \alpha^*) - \theta(p_{cand_j}) \times C \times (\alpha^* - 1).$$

Rearranging the equation:

$$0 \geq \tau(p_{cand_i}, \alpha^*) - \tau(p_{cand_j}, \alpha^*) \geq (\alpha^* - 1) \times C \times (\theta(p_{cand_i}) - \theta(p_{cand_j})).$$

Therefore,

$$0 \geq (\alpha^* - 1) \times C \times (\theta(p_{cand_i}) - \theta(p_{cand_j})) \text{ and } \theta(p_{cand_i}) \leq \theta(p_{cand_j}) \text{ as claimed.}$$

□

As stated earlier, a stopping criterion is needed to stop phase II. Next theorem presents the condition and proves its optimality.

Theorem 3.1. *Suppose j is the first iteration that satisfies the following*

$$\tau(p_j, \alpha^*) - \tau(p_{cand_j}, \alpha^*) \geq (K - \theta(p_{cand_j})) \times C \times (\alpha^* - 1) \tag{9}$$

Then the following assertion holds for all $k > j$

$$\tau(p_k, 1) \geq \tau(p_{cand_j}, 1) \text{ if } \theta(p_k) \leq K \tag{10}$$

Proof. For a contradiction, assume that there exists a $k > j$ such that $\tau(p_k, 1) < \tau(p_{cand_j}, 1)$ and $\theta(p_k) \leq K$. Then, from this assumption and rule (8), one can conclude that $p_{cand_k} = p_k$. Using this conclusion and the definition of kSPA,

$$\tau(p_j, \alpha^*) \leq \tau(p_k, \alpha^*) = \tau(p_{cand_k}, \alpha^*).$$

Using the assumption

$$\tau(p_k, 1) < \tau(p_{cand_j}, 1),$$

one can easily say

$$\tau(p_{cand_k}, 1) < \tau(p_{cand_j}, 1) \text{ because } p_{cand_k} = p_k.$$

Then, by equation (7),

$$\tau(p_{cand_k}, \alpha^*) - \theta(p_{cand_k}) \times C \times (\alpha^* - 1) < \tau(p_{cand_j}, 1).$$

Then, one can reach

$$\tau(p_j, \alpha^*) - \theta(p_{cand_k}) \times C \times (\alpha^* - 1) < \tau(p_{cand_j}, 1),$$

because $\tau(p_j, \alpha^*) \leq \tau(p_{cand_k}, \alpha^*)$ as shown above.

Using equation (7),

$$\tau(p_j, \alpha^*) - \theta(p_{cand_k}) \times C \times (\alpha^* - 1) < \tau(p_{cand_j}, \alpha^*) - \theta(p_{cand_j}) \times C \times (\alpha^* - 1).$$

Rearranging the equation,

$$\tau(p_j, \alpha^*) - \tau(p_{cand_j}, \alpha^*) < (\theta(p_{cand_k}) - \theta(p_{cand_j})) \times C \times (\alpha^* - 1).$$

Using the assumption

$$\theta(p_k = p_{cand_k}) \leq K,$$

one can conclude that

$$\tau(p_j, \alpha^*) - \tau(p_{cand_j}, \alpha^*) < (K - \theta(p_{cand_j})) \times C \times (\alpha^* - 1).$$

However, this contradicts to (9). Hence p_{cand_j} is proven to be the optimum path if it satisfies (9) which is the stopping criterion for the proposed optimal algorithm. □

Having defined the PSA and shown the correctness of stopping criterion for kSPA, the proposed optimal algorithm is presented next. Pseudocode of the optimal algorithm is presented in Figure 4 where comments are given in C language style.

Input: ONP instance (s, t, D, C, K)
Output: The optimum path to the ONP instance
1. $(p, \alpha^*) \leftarrow PSA$ *// run PSA for the given ONP instance*
2. **if** $\theta(p) == K$ return p *// PSA found the optimum*
3. **if** $(\theta(p) < K)$ and $\alpha^* == 1$ return p *// PSA found the optimum*
4. update G as edge cost $= \tau(e, \alpha^*)$
5. $p_{cand} \leftarrow null$ *// candidate solution*
6. init kSPA on G *// start K-Shortest Path Algorithm on G*
7. **do**
8. $p \leftarrow$ next shortest path from kSPA
9. **if** $(\theta(p) \leq K$ and $\tau(p, 1) < \tau(p_{cand}, 1))$ *// if p is feasible and better than candidate solution*
10. $p_{cand} \leftarrow p$
11. **while** $(\tau(p, \alpha^*) - \tau(p_{cand}, \alpha^*) < (K - \theta(p_{cand})) \times C \times (\alpha^* - 1))$ *// stopping criterion*
12. return p_{cand}

FIGURE 4. Optimal Algorithm

4. **Graph modeling.** In this study, the graphs are modeled in both discrete and continuous settings. Regarding the discrete setting, an 8-regular lattice discretization of the obstacle field is used. In order to model the ONP in a continuous setting, a special kind of graph, namely tangent arc graphs (TAG), is used. It is named as such since the graph contains edges of only tangents to the obstacles and arcs placed on the boundary of the obstacles. Below, firstly the explanation of TAG construction is provided and then it is shown that the optimum solution of the ONP lies in the TAG.

4.1. **Creating Tangent-Arc Graphs (TAG).** The TAG associated with an ONP instance (s, t, D, C, K) can be represented by (V, E) where V will be the set of vertices, and E will be the set of edges in the TAG. The creation of the associated TAG will be completed in two steps where in the first one the linear edges and their corresponding start and end vertices are created. The TAG is started to be created by including the s and t into V. Then, tangent edges from s and t to each disk are included into E and their associated vertices are included into V (see Figure 5(a)). For a pair of nonintersecting discs one can define eight tangent edges and eight vertices as demonstrated in Figure 5(b) and add them to the corresponding sets of TAG. Tangent edges and vertices for a pair of intersecting discs can be defined in a similar way as shown in Figure 5(c). As the last linear edge to be included in the set E, the line segment from s to t is created.

In the second step, arc shaped edges are created. After creating the tangent edges and corresponding vertices, the arcs between the vertices on the boundary of

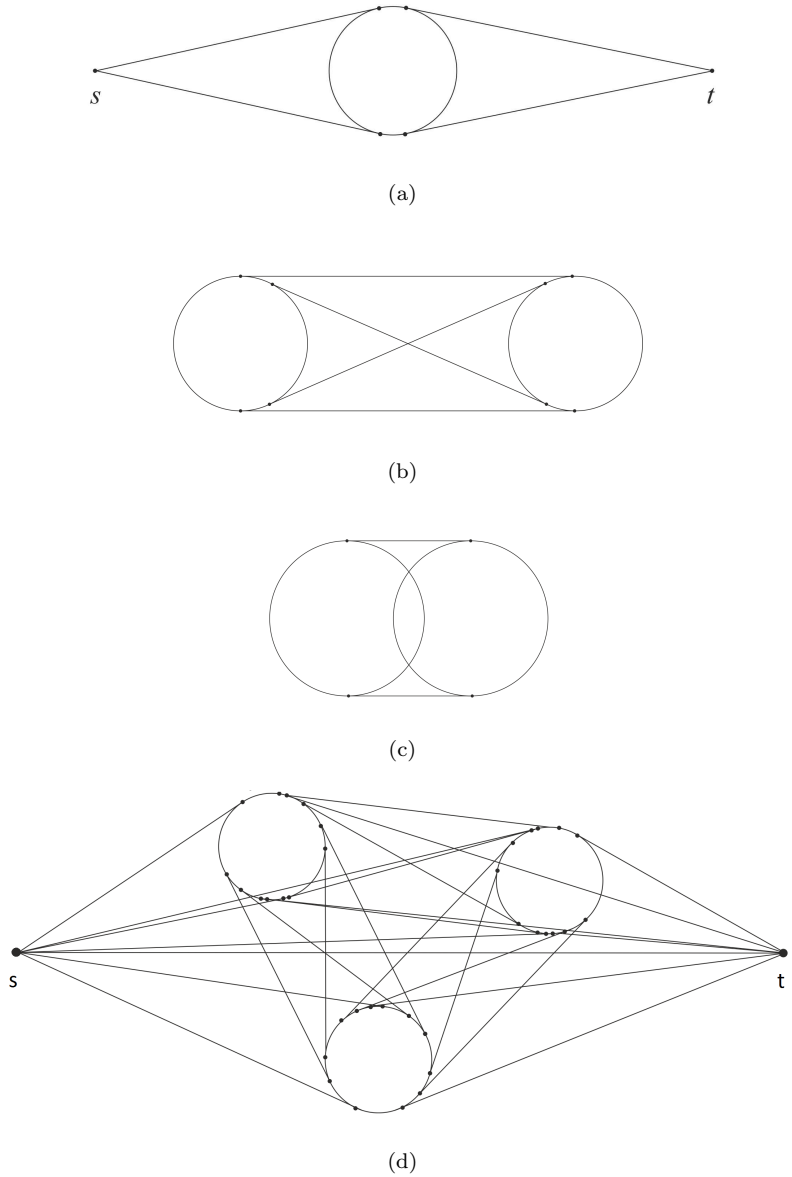

FIGURE 5. Details for creating a TAG.

each disk are created as edges and added to the graph. In order to keep the graph as simple as possible, the number of edges is tried to be kept as minimum as possible. For this, only the arcs between two neighbor vertices (e.g. non-overlapping arcs) are created. It can be easily seen that with combinations of these non-overlapping arc segments, any possible arc between all possible vertex pairs can be spanned. In Figure 5(d), a TAG constructed for an ONP instance with three discs is presented.

TAG modeling requires a construction process which has a complexity dependent on $|D|$. More specifically, the complexity of constructing tangent edges between

disks is $O(|D|^2)$. This is because tangent edge construction involves a constant number of operation for processing every disk pair where there are $|D| * (|D| - 1)/2$ disk pairs. For creating the arc edges, there are $O(|D|)$ number of vertices on each disk which means $O(|D| \times log(|D|))$ number of operations are required in order to sort the vertices clockwise before creating the arcs between them. Since the number of disks is $|D|$, the complexity of creating all arcs is $O(|D|^2 \times log(|D|))$. Thus the complexity of creating a TAG is $O(|D|^2 \times log(|D|))$. A similar algorithm is used in [3] with a complexity of $O(|D|^3 \times log(|D|))$ where they tackle a different problem.

4.2. **TAG contains the optimum path.** Although there are uncountably infinitely many curves over which to minimize, below it is shown that the optimum path for an ONP instance must be a path in the associated TAG. However, the following definitions are needed:

$p^*_{K,D}$: shortest (optimum) path for (s, t, D, C, K) instance
$\phi(p, D)$: set of discs p intersects with in D
$D' = D - \phi(p^*_{K,D}, D)$: set of discs $p^*_{K,D}$ does not intersect with.
In this subsection, the set of discs on which the cost function (τ) will be evaluated is important and therefore overriding the previous definition, it will be used as $\tau(p, D)$. Therefore, $\tau(p, D) = \ell(p) + |\phi(p, D)| \times C$.

Proposition 3. *$p^*_{K,D} = p^*_{0,D'}$ i.e. optimum path for (s, t, D, C, K) and optimum path for (s, t, D', C, 0) are identical (paths neutralizing exactly the same discs and having the same total cost, i.e. parallel paths, are also considered as identical).*

Proof. For a contradiction assume

$$p^*_{K,D} \neq p^*_{0,D'}.$$

Then,

$$\tau(p^*_{0,D'}, D') < \tau(p^*_{K,D}, D').$$

Therefore,

$$\ell(p^*_{0,D'}) + |\phi(p^*_{0,D'}, D')| \times C < \ell(p^*_{K,D}) + |\phi(p^*_{K,D}, D')| \times C.$$

Observe that

$$|\phi(p^*_{0,D'}, D')| = 0$$

because $p^*_{0,D'}$ is the path with 0 neutralizations. On the other hand,

$$|\phi(p^*_{K,D}, D')| = 0$$

because, by definition, D' is the set of discs $p^*_{K,D}$ does not intersect with. Therefore,

$$\ell(p^*_{0,D'}) < \ell(p^*_{K,D}) \tag{11}$$

At the same time one can easily observe that

$$|\phi(p^*_{0,D'}, D)| \leq |\phi(p^*_{K,D}, D)| = L \leq K \tag{12}$$

where L is the number of discs $p^*_{K,D}$ intersects with in D. That is, if one traverses along $p^*_{0,D'}$ in D, then it may intersect at most L discs which are actually the discs $p^*_{K,D}$ intersects with (see how D' is defined above). Then, by firstly multiplying each side by C in Equation (12) and adding Equations (11) and (12) side by side, it is obtained

$$\ell(p^*_{0,D'}) + |\phi(p^*_{0,D'}, D)| \times C < \ell(p^*_{K,D}) + |\phi(p^*_{K,D}, D)| \times C \tag{13}$$

which means that $\tau(p^*_{0,D'}, D) < \tau(p^*_{K,D}, D)$. That is, $p^*_{0,D'}$ is shorter than $p^*_{K,D}$ in D. However, this is a contradiction because by definition $p^*_{K,D}$ must be the optimum path for (s, t, D, C, K). Therefore $p^*_{K,D} = p^*_{0,D'}$ □

So, the optimum path for any ONP instance (s, t, D, C, K) is exactly the same path for some other ONP instance (s, t, D', C, 0) where allowed number of neutralizations is 0 and $D' \subseteq D$. Hence, if it can be shown that the optimum path for the ONP instance (s, t, D', C, 0) lies on the TAG (V', E'), then it can also be shown that the optimum path for the ONP instance (s, t, D, C, K) lies on the TAG (V, E), where $(V', E') \subseteq (V, E)$.

Theorem 4.1. *Optimum path for an ONP instance (s, t, D', C, 0) lies on its associated TAG (V', E').*

Proof. Since the optimum path for (s, t, D', C, 0) does not penetrate any discs, from the triangle inequality, the path must either walk on the boundaries of discs or use straight lines (tangents) between disks. By definition, TAG already consists of the arcs on the boundaries and tangents between discs. Therefore, the optimum path lies on the associated TAG. □

5. **Computational experiments and results.** This section presents the results of the comprehensive computational experiments revealing the performance of the proposed optimal algorithm on real, continuous, discretized, and random data by comparing with an ant system based algorithm and with the well-known SCIP integer programming solver [4, 18]. Since an optimal algorithm is proposed, the basic metric to measure the performance of the proposed algorithm is its run time, RT (in seconds). Number of paths that the kSPA retrieves ($\#RP$) is another data reported in the following tables. Note that a value of 0 for $\#RP$ indicates that the optimal algorithm finds the optimum solution in Phase I. Additionally, the number of neutralizations on the optimal path ($\theta(p^*)$) and its total cost ($\tau(p^*)$) are reported as well. Regarding the ant system based algorithm, percent deviation from the optimal path cost value and run time is provided. For the IP solver, only the run time value is reported since the optimal paths cost is already same with the value found by the proposed optimal algorithm.

The specific application domain considered in our experiments is naval minefield navigation. A particular instance we make use of is a U.S. Navy minefield data set (called the COBRA data) with 39 obstacle disks that first appeared in Witherspoon et al.[31] and was later referred to in [28, 27, 14, 34, 33, 3, 2, 1, 7]. The COBRA data is illustrated in Figure 6 and tabulated in Table 1.

In addition to the real data explained above, several COBRA-like instances are used in order to fully reveal the performance of the proposed algorithm. Specifically, 100 instances are created for the obstacle field with 100 disks with radius=5 on a [0,100]×[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, the solution of a problem instance (s, t, D, 5, 1) is presented on continuous space and discretized space with three different resolution settings (Figure 7). On continuous space the optimum path (p_1) does not need to make any neutralizations. p_2, p_3 and p_4 are the optimum solutions when discretization is performed with 50×50,

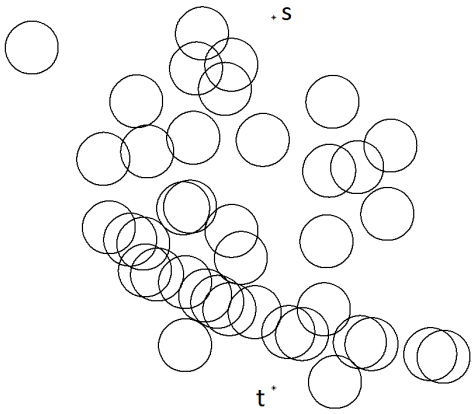

FIGURE 6. An actual naval minefield data set, called the COBRA data.

TABLE 1. Center coordinates of COBRA disks.

X-coordinate	Y-coordinate	X-coordinate	Y-coordinate	X-coordinate	Y-coordinate
321.17	158.27	54.23	201.12	158.17	516.48
215.13	428.31	-145.67	703.06	-151.01	572.15
221.12	557.31	-166.36	299.42	296.16	163.31
163.31	186.14	28.31	205.03	-79.26	709.99
100.40	376.47	-105.75	262.40	185.31	182.18
116.39	110.84	-128.60	274.12	-61.19	345.12
-91.27	664.45	-82.87	348.29	105.47	509.80
-19.93	568.04	-310.23	402.92	-320.73	532.23
-35.11	242.61	-169.99	438.90	95.39	248.12
-78.75	396.14	-245.28	372.05	-166.45	180.33
-134.53	769.27	-258.45	641.03	111.60	640.10
-219.32	313.68	-455.72	742.57	-157.10	441.96
-242.22	321.51	-237.86	546.19	-269.98	379.65

20×20 and 10×10 vertices, respectively. ($\tau(p_1) = 103.45, \tau(p_2) = 110.63, \tau(p_3) = 113.28, \tau(p_4) = 121.57$)

5.1. **Results on COBRA data.** The proposed optimal algorithm is firstly tried on original COBRA data and its discretized version. While discretizing the COBRA data, vertices are created on the coordinates that are multiples of 10 with enough vertices spanning all obstacle field. Therefore, straight edges have a length of 10 whereas the diagonal edges have a length of $10\sqrt{2}$. Results regarding the tests conducted on both original COBRA data and its discretized version are given in Table 2 where several C and K value combinations. Observe that the proposed optimal algorithm finds the optimum result very quickly when compared with the IP solver where proposed optimal algorithm works up to 650 times faster than the IP solver. As the reader can notice easily, run time of the IP solver is provided only for the continuous environment because it can not run it on discretized environment due to the excessive number of vertices and edges that it can handle. Another major observation is that the results are obtained by Phase I of the optimal algorithm. Specifically on continuous setting, when $C \geq 10$, the optimum solution for $K = 1, 2, 3$ is the same path having one neutralization. On discretized setting, the

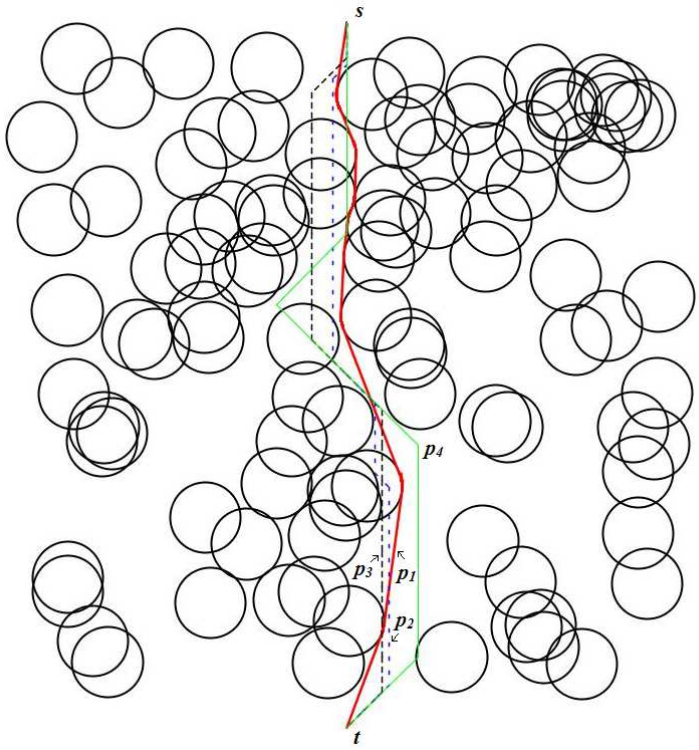

FIGURE 7. An example depicting the solutions on continuous space and discretized space at three different resolution settings.

optimum solution for $K = 1, 2, 3$ is the same path having one neutralization when $C \geq 50$.

5.2. **Performance on random COBRA-like obstacle fields on continuous graphs.** As described in the previous subsection, 100 random COBRA-like obstacle fields are created with 100 disks in each. The performance of the optimal algorithm is tried on those instances with various C and K values and compared with the AS algorithm developed in [4]. Tables 3 and 4 present the performance comparison of the optimal algorithm and AS algorithm both in terms of solution quality and run time where each figure is the average of 100 obstacle fields. In Table 3 and 4, $\%Dev.$ column presents the deviation of the AS solution from the optimal and RT column presents its run time in seconds.

It can be easily observed that $\#RP$ and RT are proportional with each other which means that almost all of running time is due to kSPA. Another observation is the effectiveness of the optimal algorithm when K increases. This is because the algorithm firstly uses a heuristic to find a provably suboptimal solution. If it can find one, then it stops. The same observation is valid for various C values in Table 4. When compared with the run time of the IP solver, proposed optimal algorithm is faster for each C and K value where the performance gain is up to 640 times. Therefore, one can easily see that the proposed optimal algorithm is robust for high C and K values. On the other hand, one can see that AS presents a poor

TABLE 2. Result on original and discretized COBRA data where several C and K value combinations are tried.

C	**K**	**Continuous Env.**					**Discretized Env.**			
		Proposed Optimal Algo.				**IP Solver**	**Proposed Optimal Algo.**			
		$\theta(p^*)$	$\tau(p^*)$	**#RP**	**RT (s)**	**RT (s)**	$\theta(p^*)$	$\tau(p^*)$	**#RP**	**RT (s)**
1	0	0	977.54	0	**0.055**	0.36	0	1043.26	0	0.078
	1	1	708.97	0	**0.055**	3.34	1	758.99	0	0.051
	2	2	704.83	0	**0.042**	4.38	2	726.85	0	0.086
	3	3	703	0	**0.009**	4.04	3	711.28	0	0.034
5	0	0	977.54	0	**0.041**	0.48	0	1043.26	0	0.055
	1	1	712.97	0	**0.018**	4.10	1	762.99	0	0.033
	2	2	712.83	0	**0.008**	3.9	2	734.85	0	0.052
	3	2	712.83	0	**0.010**	3.76	3	723.28	0	0.095
10	0	0	977.54	0	**0.034**	0.34	0	1043.26	0	0.049
	1	1	717.97	0	**0.009**	3.03	1	767.99	0	0.031
	2	1	717.97	0	**0.010**	4.24	2	744.85	0	0.061
	3	1	717.97	0	**0.006**	3.94	3	738.28	0	0.014
20	0	0	977.54	0	**0.039**	0.42	0	1043.26	0	0.045
	1	1	727.97	0	**0.011**	2.90	1	777.99	0	0.032
	2	1	727.97	0	**0.010**	3.79	2	764.85	0	0.014
	3	1	727.97	0	**0.009**	3.85	2	764.85	0	0.016
50	0	0	977.54	0	**0.020**	0.39	0	1043.26	0	0.043
	1	1	757.97	0	**0.011**	3.04	1	807.99	0	0.015
	2	1	757.97	0	**0.008**	3.85	1	807.99	0	0.014
	3	1	757.97	0	**0.009**	3.61	1	807.99	0	0.020

performance both in terms of path cost and run time. When compared with the proposed optimal algorithm, AS finds the solution inferior more than 9% in up to 200 times more time than the optimal algorithm. The poor performance in total cost is due to its stochastic nature whereas its high runtime is due to its iterative design while the ants try to build paths.

TABLE 3. Average results of 100 random COBRA-like obstacle fields for various K values ($C = 1$).

K	**Proposed Optimal Algorithm**				**IP Solver**	**Ant System Algorithm**	
	$\theta(p^*)$	$\tau(p^*)$	**#RP**	**RT (s)**	**RT (s)**	**% Dev.**	**RT (s)**
1	0.96	115.01	29.4	**15.217**	166.95	7.24%	258.87
2	1.80	108.06	10.5	**5.444**	167.96	6.90%	262.03
3	2.56	105.32	1.7	**0.972**	138.02	3.41%	181.97
4	3.04	104.37	0.7	**0.490**	86.10	0.84%	77.88
5	3.26	104.22	0.6	**0.345**	67.23	0.08%	31.50
6	3.35	104.16	0.01	**0.118**	63.64	0.00%	8.03
7	3.36	104.15	0.00	**0.088**	47.48	0.00%	5.33
8	3.36	104.15	0.00	**0.084**	53.81	0.00%	5.32
9	3.36	104.15	0.00	**0.085**	51.16	0.00%	5.33

TABLE 4. Average results of 100 random COBRA-like obstacle fields for various C values ($K = 2$).

C	Proposed Optimal Algorithm				IP Solver	Ant System Algorithm	
	$\theta(p^*)$	$\tau(p^*)$	#RP	RT (s)	RT (s)	% Dev.	RT (s)
0.1	1.99	106.39	11.9	**6.734**	264.838	3.21%	1072.552
0.5	1.90	107.14	11.3	**6.482**	247.793	9.45%	678.499
1	1.80	108.06	10.5	**5.444**	228.303	6.90%	262.030
2	1.59	109.78	8.9	**5.292**	179.448	5.67%	128.304
5	1.24	113.94	4.7	**3.139**	117.311	1.62%	49.419
10	0.86	119.15	3.1	**2.100**	58.123	0.07%	23.278
20	0.46	125.81	0.0	**0.076**	35.616	0.00%	12.800

5.3. **Performance on random COBRA-like obstacle fields on discretized graphs.** In order to reveal the performance of the proposed optimal algorithm on discrete graphs, the random COBRA-like obstacle fields are discretized on three different resolutions where each vertical and horizontal edge has a length of 10, 5 and 2. Thus, an obstacle field of [0,100]×[0,100] will have 10×10, 20×20, and 50×50 vertices from its coarsest resolution to its finest resolution. Results regarding the performance with various K values on different resolutions are given in Table 5. Observe in Table 5 that for any K value, as the resolution gets finer, #RP and RT measures increase fast which due to the increase in edges and therefore in number of paths. Another interesting finding is observed when Table 3 and 5 are compared. See that even the #RP results on worst resolution (10×10 discrete setting) is greater than the one for continuous setting. However, even though run time on 10×10 setting is significantly less than continuous setting, the run time on 20×20 and 50×50 settings get considerably large.

The reason why the #RP value get larger on finer resolution graphs is the parallel identical paths that occur more likely on discretized graphs. Due to its inherent design, the kSPA saves every instance of path that it checks until it finds the optimum path and there may be many parallel paths. Consider the following example in Figure 8. In the figure, suppose the path $s-v-x-y-z-t$ is the shortest path and we are trying to find the second shortest path (excluding the parallel paths). Let us count how many parallel identical paths are there of the shortest path. There are $\binom{5}{1} = 5$ parallel paths from s to v (because they are five edges away vertically and one edge away horizontally from each other). Similarly, there are $\binom{5}{2} = 10$ parallel paths from x to y, and $\binom{4}{3} = 4$ parallel paths from z to t. Thus, even in this small graph $5 \times 10 \times 4 = 200$ parallel identical paths are observed. Therefore memory footprint for the kSPA may grow in an unexpected way on the discretized graphs due to identical parallel paths.

5.4. **Complexity for building TAG vs. Discretized graphs.** In this study, graph modeling of the environment is performed in two different ways: TAG and discretization. The first model solves ONP in continuous environment and therefore can find optimal solution, whereas the second approach handles the problem in discrete space and can only find a sub-optimal solutions. From practical point of view, some discussions about the practical preference of building continuous and discretized graphs should be included. As it is shown in previous section, the

TABLE 5. Results of proposed optimal algorithm on 50 random COBRA-like obstacle fields for various K values ($C = 1$)

K	[10]×[10]				[20]×[20]				[50]×[50]			
	$\theta(p^*)$	$\tau(p^*)$	#RP	RT (s)	$\theta(p^*)$	$\tau(p^*)$	#RP	RT (s)	$\theta(p^*)$	$\tau(p^*)$	#RP	RT (s)
1	0.88	173.70	45.6	0.027	0.54	157.68	4866.1	70.044	0.62	137.32	3800.0	546.432
2	1.84	161.31	134.7	0.086	1.50	138.83	3138.3	48.120	1.60	122.26	2800.0	409.321
3	2.78	147.07	300.7	0.455	2.50	127.72	1657.8	24.312	2.46	116.06	3189.2	192.275
4	3.76	134.59	158.4	0.164	3.64	119.29	504.9	8.294	3.16	113.31	4079.5	162.626
5	4.82	124.43	18.9	0.012	4.62	114.55	73.1	0.180	3.98	111.02	3416.2	139.945
6	5.28	119.65	7.3	0.007	5.32	113.01	66.4	0.118	4.54	110.51	3157.9	132.476
7	6.04	116.68	4.9	0.005	6.04	111.76	22.2	0.046	4.90	110.05	2616.8	104.004
8	6.48	114.57	7.6	0.007	6.76	110.96	29.3	0.055	5.56	109.65	1201.6	77.306
9	7.10	113.06	4.7	0.005	7.28	110.59	31.7	0.059	6.04	109.35	400.0	7.138

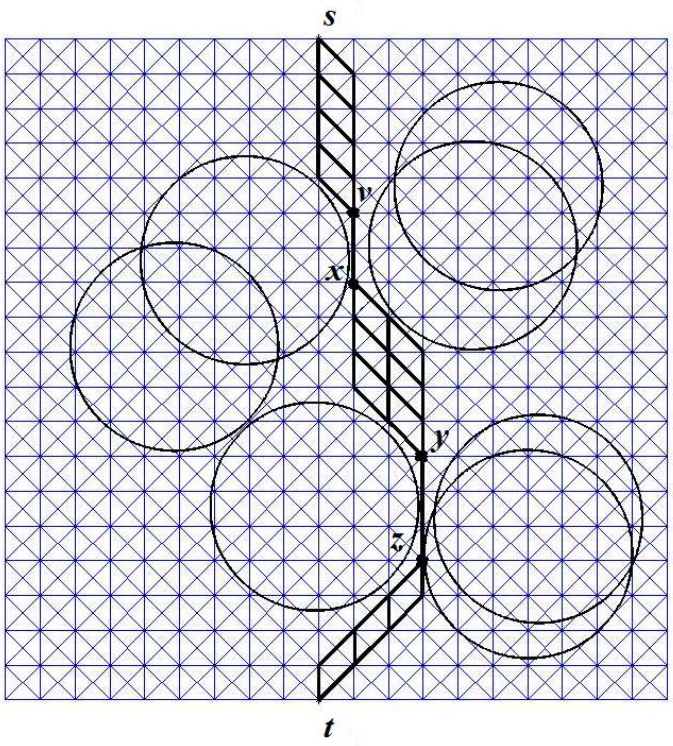

FIGURE 8. An example how there occurs many parallel paths on a discretized minefield

complexity of creating a TAG is $O(|D|^2 \times log(|D|))$. Note that this complexity is independent of the map size. However, when discretization is used the complexity of the construction of the lattice graph depends on the map size. Grid construction, as opposed to TAG, is not affected by the cardinality of $|D|$. Specifically, if x denotes the number of vertices on a coordinate axis (assume for simplicity there are equal number of vertices on both coordinates), the complexity of grid construction is $O(x^2)$ which is not dependent on $|D|$. Therefore, in small graphs (in terms of $|D|$) using TAG modeling is more efficient which also guarantees to find the optimal solution in the continuous space. In large graphs (with high number of disks), on

the other hand, the TAG construction time increases, whereas the creation of the grid can take considerably shorter times.

6. **Summary and conclusions.** In this study, the obstacle neutralization problem (ONP) is tackled. The goal of ONP is to safely and swiftly navigate an agent from a given source location to a destination through an arrangement of obstacles in the plane. The agent is capable of neutralizing a limited number of disks en route at a cost. The main issue is how to optimally use this neutralization capability for the shortest $s-t$ traversal. A two-phased optimal algorithm is presented to solve this problem and its correctness is proved.

The performance of the algorithm is tried both on real minefield examples and synthetic but realistic obstacle fields. Furthermore, the algorithm is tried both on continuous and discrete spaces. The computational results suggest that the algorithm performs better in continuous space, which is a desirable property. The inferior performance in discrete space is due to large number of identical parallel paths that may a burden for Phase II of the algorithm. The run time performance of the algorithm is compared with an IP solver and as a result the proposed optimal algorithm runs up to 650 times faster than the IP solver. As a future work, in order to improve the performance of the optimal algorithm, a kSPA that can deal with parallel paths can be worked on. Furthermore, we believe that the proposed algorithm can be applied to several problem domains considering optimal navigation including ships neutralizing ice zones and military air vehicles neutralizing danger zones.

Acknowledgments. Work of Ali Fuat Alkaya was supported by The Scientific and Technological Research Council of Turkey (TUBITAK), Project No. 114M069. Work of Dindar Oz was supported by the Marmara University Scientific Research Committee, Project No. FEN-C-DRP-090414-0103.

REFERENCES

[1] V. Aksakalli and I. Ari, Penalty-based algorithms for the stochastic obstacle scene problem, *INFORMS Journal on Computing,* **26** (2014), 370–384.

[2] V. Aksakalli and E. Ceyhan, Optimal obstacle placement with disambiguations, *Annals of Applied Statistics,* **6** (2012), 1730–1774.

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

[4] R. Algin, A. F. Alkaya, V. Aksakalli and D. Oz, 2013. An ant system algorithm for the neutralization problem, *Advances in Computational Intelligence,* Volume 7903 of the series Lecture Notes in Computer Science, (2013), 53–61.

[5] R. Algin and A. F. Alkaya, Solving the obstacle neutralization problem using swarm intelligence algorithms, *Proceedings of 7th International Conference on Soft Computing and Pattern Recognition,* (2015), 187–192.

[6] A. F. Alkaya and R. Algin, Metaheuristic based solution approaches for the obstacle neutralization problem, *Expert Systems with Applications,* **42** (2015), 1094–1105.

[7] A. F. Alkaya, V. Aksakalli and C. E. Priebe, A penalty search algorithm for the obstacle neutralization problem, *Computers and Operations Research,* **53** (2015), 165–175.

[8] J. F. Bekker and J. P. Schmid, Planning the safe transit of a ship through a mapped minefield, *Journal of the Operations Research Society of South Africa,* **22** (2006), 1–18.

[9] W. M. Carlyle, J. O. Royset and R. K. Wood, Lagrangian relaxation and enumeration for solving constrained shortest-path problems, *Networks,* **52** (2008), 256–270.

[10] Costal Battlefied Reconnaissance and Analysis - (COBRA), `http://www.navy.mil/navydata/fact_display.asp?cid=2100&tid=1237&ct=2`, Last access: September 1, 2014.

[11] G. Dahl and B. Realfsen, *Curve Approximation and Constrained Shortest Path Problems,* International Symposium on Mathematical Programming (ISMP97), 1997.

[12] G. Dahl and B. Realfsen, Curve approximation constrained shortest path problems, *Networks,* **36** (2000), 1–8.
[13] I. Dumitrescu and N. Boland, Algorithms for the weight constrained shortest path problem, *International Transactions in Operational Research,* **8** (2001), 15–29.
[14] D. E. Fishkind, C. E. Priebe, K. Giles, L. N. Smith and V. Aksakalli, Disambiguation protocols based on risk simulation, *IEEE Transactions on Systems, Man, and Cybernetics, Part A,* **37** (2007), 814–823.
[15] L. Guo and I. Matta, Search space reduction in QoS routing, *Computer Networks,* **41** (2003), 73–88.
[16] G. Y. Handler and I. Zang, A dual algorithm for the constrained shortest path problem, *Networks,* **10** (1980), 293–309.
[17] A. Jüittner, B. Szviatovski, I. Mecs and Z. Rajko, Lagrange relaxation based method for the QoS routing problem, *Proceedings of 20th Annual Joint Conference of the IEEE Computer Communications Societies,* **2** (2001), 859–868.
[18] T. Koch, *Rapid Mathematical Prototyping*, Ph.D. Thesis, Technische Universität Berlin, 2004.
[19] F. Kuipers, T. Korkmaz, M. Krunz and P. Van Mieghemt, Performance evaluation of constraint-based path selection algorithms, *IEEE Network,* **18** (2004), 16–23.
[20] J. Latourell, B. Wallet and B. Copeland, Genetic algorithm to solve constrained routing problem with applications for cruise missile routing, *Proceedings of SPIE,* **3390** (1998), 490–500.
[21] S. H. K. Lee, *Route Optimization Model for Strike Aircraft*, Master's thesis, Naval Postgraduate School, Monterey, California, 1995.
[22] P. C. Li, *Planning the Optimal Transit for a Ship Through a Mapped Minefield*, Master's thesis, Naval Postgraduate School, Monterey, California, 2009.
[23] Y. M. Marghi, F. Towhidkhah and S. Gharibzadeh, A two level real-time path planning method inspired by cognitive map and predictive optimization in human brain, *Applied Soft Computing,* **21** (2014), 352–364.
[24] C. Mou, W. Qing-xian and J. Chang-sheng, A modified ant optimization algorithm for path planning of UCAV, *Applied Soft Computing,* **8** (2008), 1712–1718.
[25] R. Muhandiramge, N. Boland and S. Wang, Convergent network approximation for the continuous euclidean length constrained minimum cost path problem, *SIAM journal on Optimization,* **20** (2009), 54–77.
[26] R. Nygaard, J. HusZy and D. Haugland, Compression of image contours using combinatorial optimization, *Proceedings of the International Conference on Image Processing-ICIP98,* **1** (1998), 266–270.
[27] C. E. Priebe, D. E. Fishkind, L. Abrams and C. D. Piatko, Random disambiguation paths for traversing a mapped hazard field, *Naval Research Logistics,* **52** (2005), 285–292.
[28] C. E. Priebe, T. E. Olson and D. M. Healy Jr. Exploiting stochastic partitions for minefield detection, *Proceedings of the SPIE,* **3079** (1997), 508–518.
[29] D. S. Reeves and H. F. Salama, A distributed algorithm for delay-constrained unicast routing, *IEEE/ACM Transactions on Networking,* **8** (2000), 239–250.
[30] J. O. Royset, W. M. Carlyle and R. K. Wood, Routing military aircraft with a constrained shortest-path algorithm, *Military Operations Research,* **14** (2009), 31–52.
[31] N. H. Witherspoon, J. H. Holloway, K. S. Davis, R. W. Miller and A. C. Dubey, The coastal battlefield reconnaissance and analysis (cobra) program for minefield detection, *Proceedings of the SPIE: Detection Technologies for Mines and Minelike Targets, Orlando, Florida,* **2496** (1995), 500–508.
[32] B. Yang, Y. Ding, Y. Jin and K. Haho, Self-organized swarm robot for target search and trapping inspired by bacterial chemotaxis, *Robotics and Autonomous Systems,* **72** (2015), 83–92.
[33] X. Ye, D. E. Fishkind and C. E. Priebe, Sensor information monotonicity in disambiguation protocols, *Journal of the Operational Research Society,* **62** (2011), 142–151.
[34] X. Ye and C. E. Priebe, A graph-search based navigation algorithm for traversing a potentially hazardous area with disambiguation, *International Journal of Operations Research and Information Systems,* **1** (2010), 14–27.
[35] J. Y. Yen, Finding the k shortest loopless paths in a network, *Management Science,* **17** (1971), 712–716.
[36] M. Zabarankin, S. Uryasev and R. Murphey, Aircraft routing under the risk of detection, *Naval Research Logistics,* **53** (2006), 728–747.

[37] M. Zabarankin, S. Uryasev and P. Pardalos, Optimal risk path algorithms, *Cooperative Control and Optimization (R. Murphey and P. Pardalos ed.), Kluwer Academic, Dordrecht,* **66** (2002), 273–298.

[38] Q. Zhu, J. Hu, W. Cai and L. Henschen, A new robot navigation algorithm for dynamic unknown environments based on dynamic path re-computation and an improved scout ant algorithm, *Applied Soft Computing,* **11** (2011), 4667–4676.

Received April 2015; revised December 2015.

E-mail address: falkaya@marmara.edu.tr
E-mail address: dindar.oz@yasar.edu.tr