A Penalty Search Algorithm for the Obstacle Neutralization Problem

Ali Fuat Alkaya[a,*], Vural Aksakalli[b], Carey E. Priebe[c]

[a] *Computer Engineering Department, Marmara University, Istanbul, 34722, TURKEY.*

[b] *Industrial Engineering Department, Istanbul Sehir University, Istanbul, 34662, TURKEY.*

[c] *Applied Mathematics & Statistics Department, Johns Hopkins University, MD, 21218, USA.*

Abstract

We consider a path planning problem wherein an agent needs to swiftly navigate from a source to a destination through an arrangement of obstacles in the plane. We suppose the agent has a limited neutralization capability in the sense that it can safely pass through an obstacle upon neutralization at a cost added to the traversal length. The agent's goal is to find the sequence of obstacles to be neutralized en route that minimizes the overall traversal length subject to the neutralization limit. We call this problem the obstacle neutralization problem (ONP), which is essentially a variant of the intractable weight-constrained shortest path problem in the literature. In this study, we propose a simple, yet efficient and effective suboptimal algorithm for ONP based on the idea of penalty search and we present special cases where our algorithm is provably optimal. Computational experiments involving both real and synthetic naval minefield data are also presented.

Keywords: combinatorial optimization, path planning, weight-constrained shortest path, suboptimal algorithm.

1. Introduction

The problem of finding shortest paths in the presence of a resource constraint is observed in many real life situations. Several such cases are described below:

- The problem of minimizing the time for data to reach destination subject to a given total delay limit in the telecommunications industry [1],
- the problem of finding the path for a military aircraft with minimum probability of being detected by enemy radar subject to fuel constraints [2], and
- the problem of approximating a curve with maximum number of breakpoints subject to storage constraints in computer graphics [3].

*Corresponding author

Email addresses : falkaya@marmara.edu.tr (A. F. Alkaya), aksakalli@sehir.edu.tr (V. Aksakalli), cep@jhu.edu (C. E. Priebe)

Problems such as above are known as the weight-constrained shortest path problem (WCSPP) in the literature, which is NP-Hard except for some trivial cases [4]. In this work, we study a variant of this problem in a path planning setting. Specifically, we consider a problem where the goal is to safely and swiftly navigate an agent from a given source location to a destination through an arrangement of disk-shaped obstacles in the plane. We assume the agent has a neutralization capability in the sense that it can safely pass through an obstacle upon neutralization at a cost added to the traversal length. However, we make the restriction that the number of allowed neutralizations is limited, perhaps due to payload capacity of the agent. The agent's goal is to find the optimal sequence of obstacles to be neutralized en route that minimizes the overall traversal length subject to the number of neutralizations available. We call this problem the obstacle neutralization problem (ONP). It should be noted if the number of allowed neutralizations is unlimited, then the problem reduces to the classical unconstrained shortest path problem and can be solved in polynomial time. Hence, we limit our focus in this study to ONP instances where the optimal solution requires utilization of the entire neutralization capability.

ONP can be defined formally as follows: Let

- the obstacle field Ω be a finite region in $\mathbb{R}^2$,
- the starting location s and the destination t be two points in Ω,
- the obstacle set D be a finite set of disks in Ω with radius r,
- the neutralization cost be denoted by $C \geq 0$, and
- the neutralization limit be denoted by $K \leq |D|$.

An agent needs to traverse from s to t along a continuous curve which is as short as possible in the sense of arclength. The agent cannot enter the obstacle disks but, if and when the agent is located at the boundary of a disk $d_i \in D$, the agent has the option to neutralize it and pass through at a cost C added to the traversal length. The agent's goal is to find a sequence of at most K obstacles to be neutralized en route that minimizes the overall $s-t$ traversal length. Throughout this study, we assume that all disks have the same neutralization cost and the same radius. For simplicity and convenience, an instance of ONP will be denoted by the tuple (s, t, D, C, K).

ONP is a combinatorial optimization problem where among $|D|$ disks, the agent must find the optimal permutation of at most K of the disks for neutralization. Clearly, ONP is a variant of the WCSPP where the weight constraint is the number of neutralizations available. On the other hand, to our knowledge, ONP as defined above has not been studied before in the open literature except for a brief mention in [5].

An instance of ONP is shown in Figure 1 with $s = (10, 20)$, $t = (10, 1)$, and $C = 0.8$ with $r = 3$. In the figure, the optimal paths for $K = 0$, 1, 2 and 3 are given in subfigures 1(a), 1(b), 1(c), and 1(d), respectively. Note that for $K = 1$, only d_6 is neutralized; for $K = 2$, both d_5 and d_6 are neutralized; and for $K = 3$, d_5, d_6, and d_7 are neutralized. Note also that it is not always the case that all K allowable neutralizations are used.

Our goal in this study is to present a simple, fast, efficient, and scalable algorithm for ONP which we call Penalty Search Algorithm (PSA). This algorithm is especially suitable for online applications. State-of-the-art algorithms

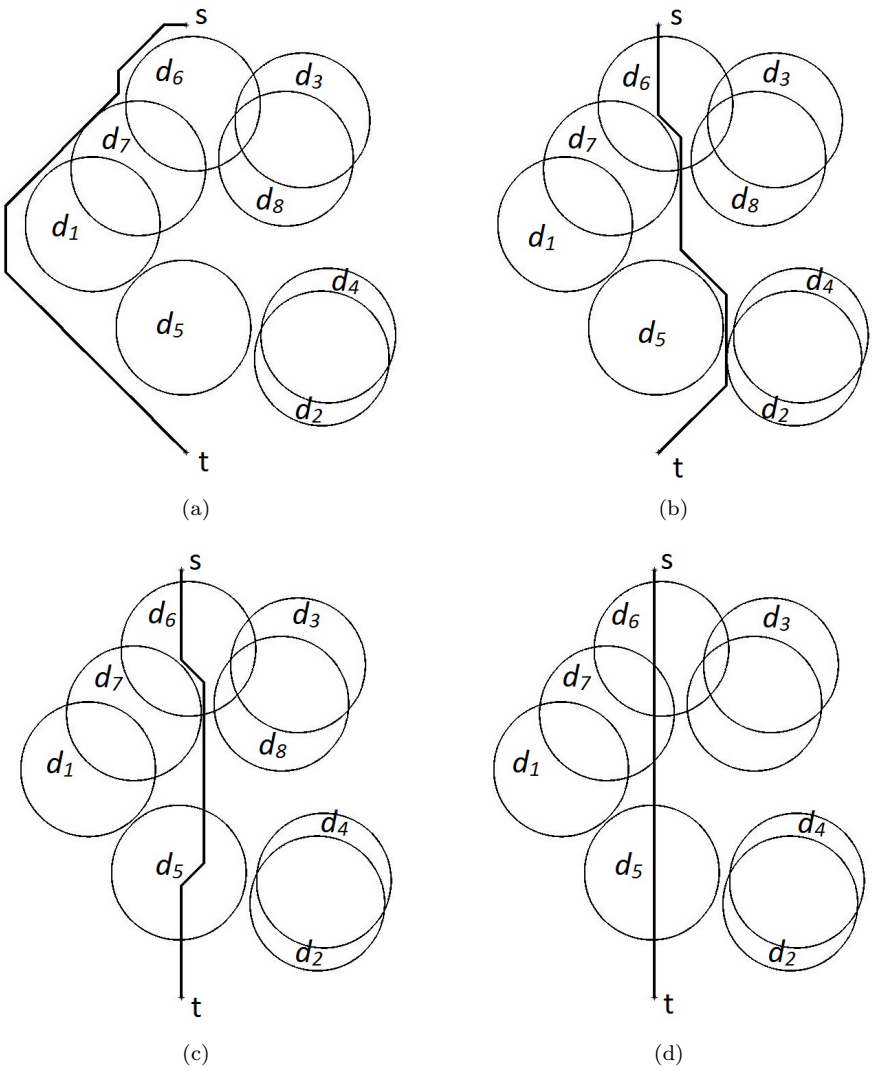


Figure 1: An instance of ONP and the associated optimal paths for $K = 0$, 1, 2, and 3 respectively.

proposed for problems closely related to ONP either require significant run times or show poor performance on ONP instances whereas PSA finds optimal or near-optimal results in short execution times. In particular, we present computational experiments comparing the performance of PSA to two other algorithms on both real and synthetic naval minefield sample data: (1) adaptation of another popular suboptimal WCSPP algorithm to ONP, called Delay-Constrained Unicast Routing (DCUR) Algorithm, and (2) an exact algorithm for ONP. Our results suggest solutions found by PSA compare very favorably to those obtained by the exact algorithm while requiring substantially less computational resources whereas DCUR exhibits relatively poor performance in general.

The remainder of this manuscript is organized as follows: Section 2 relates ONP to WCSPP and gives a comprehensive overview of current state-of-the-art in WCSPP research. Section 3 provides details of PSA, discusses its properties, and proves its optimality in certain cases. Section 4 presents our computational experiments, and Section 5 summarizes and concludes our work.

2. WCSPP and Previous Work

2.1. WCSPP and ONP

For a given graph $G = (V, E)$, let τ_{ij} and θ_{ij} denote the cost and weight of the edge $(i, j) \in E$ respectively. The objective in WCSPP 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 W_L. WCSPP is an NP-hard problem but it is solvable in polynomial time if all edge weights or edge costs are equal [4].

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 WCSPP is as follows:

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

s.t.

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

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

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

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

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

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.

In our methodology, the obstacle field is represented by an integer lattice (that is, a grid map) with diagonal edges for simplicity and convenience. We also assume that disk radius r is large enough so that a lattice edge intersects an obstacle disk at most once. Thus, should the agent decide to neutralize a particular disk and pass through that disk, then the agent will need to traverse exactly two edges intersecting that disk. This observation

implies that ONP is in fact a WCSPP where $W_L = 2K$. Observe, however, that ONP is in fact a special case of WCSPP because only the edges intersecting obstacle disks shall have a non-zero weight.

A generalization of WCSPP is the resource-constrained shortest path problem (RCSPP) where each edge consumes a number of different resources and there are limits corresponding to each resource type. Both WCSPP and RCSPP are well-studied problems both in theoretical and applied contexts. In the following sections, we first review solution methods for WCSPP and then discuss several areas of application.

2.2. Solving WCSPP

The WCSPP is a generalization of the classical (unconstrained) shortest path problem in the presence of a total edge weight constraint. In WCSPP, a lower bound can be obtained by finding the shortest path using only edge costs and ignoring edge weight constraints. Likewise, an upper bound can be determined by finding the shortest path with respect to edge weights, again ignoring weight constraints. Lagrangian relaxation of the problem can also be solved as an unconstrained shortest path problem for a lower bound. In the literature, methods developed for WCSPP primarily make use of k-th shortest paths, Lagrangian relaxation, dynamic programming, labeling/ enumeration algorithms, or a combination of these; as discussed below.

Handler and Zang [6] use Lagrangian relaxation of the weight constraint and duality theory to solve WCSPP. To close the duality gap, if there is one, they utilize the k-th shortest path algorithm with respect to the edge lengths modified by the Lagrange multipliers. Carlyle and Wood [7] also use Lagrangian relaxation of the weight constraint to obtain a lower bound for the problem. The authors then use an enumeration algorithm to close any duality gaps. Another Lagrangian relaxation based approach using subgradient optimization is presented in [8].

Mehlhorn and Ziegelmann [9], Ziegelmann [10] obtain a linear programming (LP) formulation of the problem by relaxing the integrality constraint of the integer programming formulation. The dual of this LP is then solved and the underlying graph is reduced using the method presented in [8]. To close any duality gaps, a dynamic programming approach or a k-th shortest path method is proposed.

Storandt [11] tackles the same problem when planning routes for bicycles. The problem is to find the route from A to B with the least positive height difference summed over all edges which has length at most D. She shows how speed-up techniques like contraction hierarchies can be adapted to solve this problem efficiently both in run time and space complexity where her experimental setup is based on real-world street networks.

Even though the contraction hierarchies speed-up technique is more convenient for road networks, a recent study [12] shows how to modify them to take car of the grid networks. Moreover, the author also focuses on how to compute canonical optimal paths. With her experimental work, she shows that when contraction hierarchies are applied an acceleration up to two orders of magnitude can be achieved.

Dumitrescu and Boland [13] present an exact method for WCSPP based on weight scaling in a label-setting framework. The weight scaling factor is gradually reduced until lower and upper bounds are tight, or the scaling factor has reached a critical size. The essential value of preprocessing techniques and integration of Lagrangian relaxation within a label-setting framework are demonstrated in detail in [14].

Xue [15] presents a simple primal dual algorithm for computing approximate solutions for the WCSPP. In his algorithm he defines a shortest path problem with edge lengths given by a combination of cost and weight where both of them are weighted by a multiplier in [0,1]. Then he applies a bisection method to find the best multiplier for an approximate solution.

In a recent study [16], the authors combine and expand on the best practices in the literature for WCSPP for an exact solution. Specifically, they fuse the preprocessing and Lagrangian dual solution stages and alternate between preprocessing and solving the Lagrangian dual while aggressively updating upper bounds.

2.3. Applications of WCSPP

We now briefly discuss several applications of WCSPP including the following: (1) routing of signals in telecommunication networks with QoS guarantees, (2) minimum-risk routing of military aircraft, (3) curve approximation, and (4) other optimization problems where WCSPP arises as a subproblem.

Jüttner et al. [17] study the WCSPP within the context of quality of service (QoS) routing under the name Delay Constrained Least Cost Path (DCLC) problem, and they use 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[1]. Reeves and Salama [19] 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 source to destination. While building the path, DCUR chooses edges on LC paths 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 [20] propose 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 use a variant of the Lagrangian relaxation method proposed in [6]. The aforementioned heuristics along with six other ones for DCLC are compared in [1]. This study compares these algorithms with respect to their path cost and computational complexity. Results of simulations on square lattices suggest the DCUR Algorithm provides superior results in general.

Another application of WCSPP is minimum-risk routing of military aircraft in a threat environment where edge costs represent probability of being detected by radar or surface-to-air missiles. The goal is to minimize the total risk on the path taken from source to destination subject to fuel or flight time constraints. Lee [21] considers a variant of the problem with a given fuel consumption limit. Similar to [6], the author makes use of the Lagrangian dual of the problem for a solution. Latourell et al. [22] use genetic algorithms for risk minimization in routing of military aircraft. The authors also consider a limitation on the sharpness of the turns that the aircraft can make. Zabarankin et al. [2] consider both continuous and discrete versions of the minimum-risk aircraft routing problem. The authors demonstrate how an optimal solution can be found in a continuous setting when there is only one

[1] They first prove that this Lagrangean relaxation is polynomial [17, 18] only dependent on $|V| = n$.

radar. For multiple radars, they utilize the algorithm proposed in [13] in a discrete setting. A three-dimensional treatment of the problem is presented in [23]. Royset et al. [24], on the other hand, apply the algorithm in [25] for routing of various types of military aircraft. A general scheme for convergence of network discretizations in such settings is given in [26].

One other application of WCSPP arises in curve approximation. 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 [3]. Such an approximation 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. In both cases, the problem may be formulated as a WCSPP. Dahl and Realfsen [27] present four different algorithms for the curve approximation problem: a combinatorial algorithm, a Lagrangian relaxation based algorithm, a dynamic programming algorithm, and a linear programming algorithm.

WCSPP also appears as a column generation subproblem in many optimization problems. For example, the duty scheduling problem in public transportation systems constructs daily shifts of driving tasks. Borndörfer et al. [28] discuss how WCSPP arises during the column generation phase in duty scheduling. Another example is due to [29] where the authors tackle the aircraft routing problem and solve a pricing subproblem in their column generation method by casting it as a WCSPP.

Lastly, Bekker and Schmid [30] consider a WCSPP problem similar to ONP and propose a genetic algorithm for the problem where they use a fitness function favoring shorter paths with fewer number of resource utilizations. Li [31], on the other hand, studies identification of minimum-risk routes for a ship navigating in a mapped minefield; the author proposes a greedy heuristic for the problem which calls for formation of a prioritized list of mines to be neutralized in order to quickly reduce the risk of minimum-risk paths.

3. The Penalty Search Algorithm (PSA)

This section introduces the Penalty Search Algorithm (PSA) for ONP. First, recall we assume a lattice edge intersects a given obstacle disk at most once. Thus, half of the neutralization cost is incurred upon entering the disk and the other half is incurred upon exiting. 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 $(\alpha^* C)/2$ 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.

We formally show below that when PSA dictates exactly K neutralizations, then the path returned by PSA is indeed optimal. However, PSA cannot always find a path with exactly K neutralizations, and thus sometimes returns a path with fewer than the maximum allowable number of neutralizations. Nevertheless, this path turns out to be either optimal or very close to optimal in general as illustrated in our computational experiments.

3.1. PSA and Optimality

Before we show our main result, that a PSA path with K neutralizations is optimal, we show that as α is increased, the number of neutralizations dictated by PSA decreases and the total cost increases monotonically. This fact allows us to develop an efficient mechanism for finding α^* using a bisection approach. But first we present the notation that characterizes our model.

- As before, $G = (V, E)$ denotes the lattice discretization of 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).
- $\hat{p}^*$ denote the best path found by PSA.
- For $e \in E$, $\ell(e)$ denotes its Euclidean length, which is assumed to be 1 for straight edges and $\sqrt{2}$ for diagonal edges.
- For $e \in E$, $\theta(e)$ is half the number of disks that edge e intersects. That is, $\theta(e) := (1/2) \times \sum_{d \in D} I_{e \cap d \neq \emptyset}$ where I is the indicator function.
- For $e \in E$, its associated neutralization cost is defined as $c(e) := C \times \theta(e) = (C/2) \times \sum_{d \in D} I_{e \cap d \neq \emptyset}$.
- 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 $\tau(p, \alpha) = \ell(p) + \alpha \times C \times \theta(p)$.
- $\alpha_1 = 1 < \alpha_2 < \ldots < \alpha_n$ denotes a sequence of real-valued choices for α.
- p^*_α denotes the optimal unconstrained path found by, say, A* Algorithm on G using $\tau(e, \alpha)$ as the edge costs.

Proposition 1. *$\tau(p^*_{\alpha_k}, \alpha_j) \geq \tau(p^*_{\alpha_j}, \alpha_j)$ for $j \neq k$.*

Proof. By definition, $p^*_{\alpha_j}$ is the shortest path in total cost for the $\alpha = \alpha_j$ setting. Therefore $p^*_{\alpha_k}$ has a total cost equal to or greater than that of $p^*_{\alpha_j}$ under the $\alpha = \alpha_j$ setting. □

Corollary 2. *$\tau(p^*_{\alpha_j}, \alpha_j) \geq \tau(p^*_{\alpha_k}, \alpha_k)$ for $j > k$.*

Proof. Clearly, $\tau(p^*_{\alpha_j}, \alpha_j) \geq \tau(p^*_{\alpha_j}, \alpha_k)$ because $\alpha_j > \alpha_k$. By Proposition 1, $\tau(p^*_{\alpha_j}, \alpha_k) \geq \tau(p^*_{\alpha_k}, \alpha_k)$. Hence, $\tau(p^*_{\alpha_j}, \alpha_j) \geq \tau(p^*_{\alpha_k}, \alpha_k)$. □

Proposition 3. $\theta(p^*_{\alpha_k}) \geq \theta(p^*_{\alpha_j})$ *for* $j > k$.

Proof. For a contradiction, assume $\theta(p^*_{\alpha_j}) > \theta(p^*_{\alpha_k})$. By Proposition 1, $\tau(p^*_{\alpha_k}, \alpha_j) \geq \tau(p^*_{\alpha_j}, \alpha_j)$. That is, $\ell(p^*_{\alpha_j}) + \alpha_k \times c \times \theta(p^*_{\alpha_j}) \geq \ell(p^*_{\alpha_k}) + \alpha_k \times c \times \theta(p^*_{\alpha_k})$. Then, $\ell(p^*_{\alpha_j}) + \alpha_j \times c \times \theta(p^*_{\alpha_j}) > \ell(p^*_{\alpha_k}) + \alpha_j \times c \times \theta(p^*_{\alpha_k})$ because $\theta(p^*_{\alpha_j}) > \theta(p^*_{\alpha_k})$ and $\alpha_j > \alpha_k$. This means that $\tau(p^*_{\alpha_j}, \alpha_j) > \tau(p^*_{\alpha_k}, \alpha_j)$. But this contradicts the fact that $p^*_{\alpha_j}$ is optimum for α_j. Therefore $\theta(p^*_{\alpha_k}) \geq \theta(p^*_{\alpha_j})$. □

Proposition 4. $\ell(p^*_{\alpha_j}) \geq \ell(p^*_{\alpha_k})$ *for* $j > k$.

Proof. By definition, $\ell(p^*_{\alpha_k}) = \tau(p^*_{\alpha_k}, \alpha_k) - \alpha_k \times c \times \theta(p^*_{\alpha_k})$ and $\ell(p^*_{\alpha_j}) = \tau(p^*_{\alpha_j}, \alpha_k) - \alpha_k \times c \times \theta(p^*_{\alpha_j})$. Then, $\ell(p^*_{\alpha_k}) - \ell(p^*_{\alpha_j}) = \tau(p^*_{\alpha_k}, \alpha_k) - \tau(p^*_{\alpha_j}, \alpha_k) + \alpha_k \times c \times \theta(p^*_{\alpha_j}) - \alpha_k \times c \times \theta(p^*_{\alpha_k})$. Observe that $\tau(p^*_{\alpha_k}, \alpha_k) - \tau(p^*_{\alpha_j}, \alpha_k) \leq 0$ by Proposition 1 and $\theta(p^*_{\alpha_j}) - \theta(p^*_{\alpha_k}) \leq 0$ by Proposition 3. Thus, $\ell(p^*_{\alpha_j}) \geq \ell(p^*_{\alpha_k})$. □

Theorem 5. $\tau(p^*_{\alpha_j}, \alpha_1) \geq \tau(p^*_{\alpha_k}, \alpha_1)$ *for* $j > k$.

Proof. For a contradiction, assume that $\tau(p^*_{\alpha_j}, \alpha_1) < \tau(p^*_{\alpha_k}, \alpha_1)$ for $j > k$. Then,

$$\ell(p^*_{\alpha_j}) + \alpha_1 \times c \times \theta(p^*_{\alpha_j}) < \ell(p^*_{\alpha_k}) + \alpha_1 \times c \times \theta(p^*_{\alpha_k}). \tag{7}$$

On the other hand,

$$(\alpha_k - \alpha_1) \times c \times \theta(p^*_{\alpha_j}) < (\alpha_k - \alpha_1) \times c \times \theta(p^*_{\alpha_k}) \tag{8}$$

because $\theta(p^*_{\alpha_j}) \leq \theta(p^*_{\alpha_k})$ for $j > k$ by Proposition 3. Adding (7) and (8) yields $\ell(p^*_{\alpha_j}) + \alpha_k \times c \times \theta(p^*_{\alpha_j}) < \ell(p^*_{\alpha_k}) + \alpha_k \times c \times \theta(p^*_{\alpha_k})$. But this means that $\tau(p^*_{\alpha_j}, \alpha_k) < \tau(p^*_{\alpha_k}, \alpha_k)$, which contradicts the fact that $p^*_{\alpha_k}$ is optimum for α_k. Hence, $\tau(p^*_{\alpha_j}, \alpha_1) \geq \tau(p^*_{\alpha_k}, \alpha_1)$ for $j > k$. □

Propositions 3 and 4 and Theorem 5 show monotonicity of ℓ, θ, and τ in α. PSA makes use of this result and searches for the largest value of α that yields the best path $\hat{p}^*$, which we denote by α^*. In this context, the best path refers to the shortest unconstrained path with the largest number of neutralizations, that is, $\theta(p)$, such that $\theta(p) \leq K$. Pseudo-code of PSA is presented in Figure 2. In the figure, A*(α) denotes execution of the A* Algorithm for finding the unconstrained optimal $s - t$ path using $\tau(e, \alpha)$ as the edge costs. PSA starts by setting $\alpha = 1$. If A*($\alpha = 1$) returns a path with number of 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. Observe that run time of PSA is dominated by the number of A* calls, which depends on ε. In the worst case, run time is $\mathcal{O}(log(\ell(\bar{p})/\varepsilon))$ where $\bar{p}$ is shortest $s - t$ path without any neutralizations.

The following theorem shows 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).

Input: ONP instance (s, t, D, C, K)

Output: The shortest path p with the largest $\theta(p)$ such that $\theta(p) \leq K$

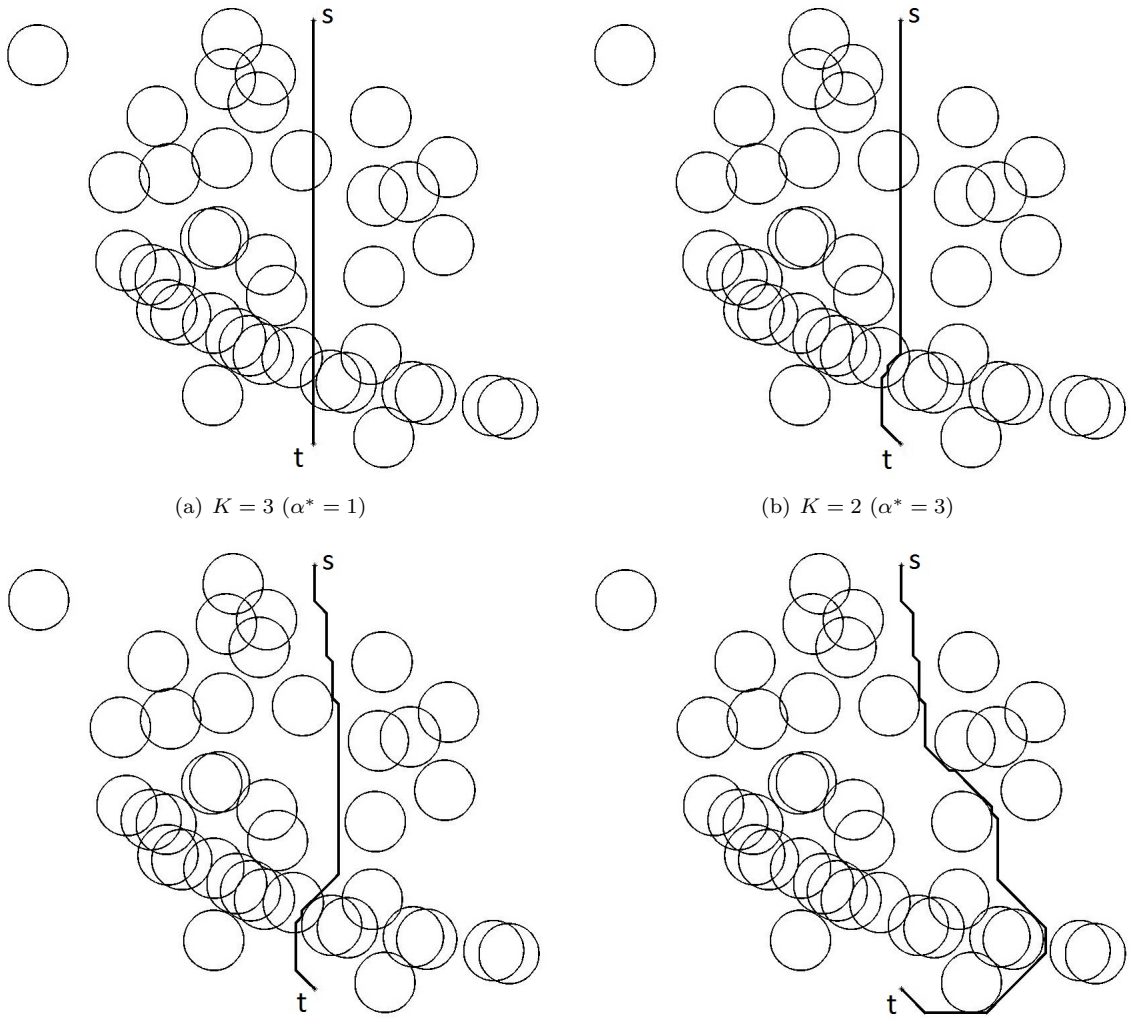
1. $\alpha = 1$
2. $p = \text{A*}(\alpha)$
3. **if** $(\theta(p) \leq K)$ return p **end if**
4. **while** $(\theta(p) > K)$
5. $\quad \alpha_{min} = \alpha$
6. $\quad \alpha = \alpha \times 10$
7. $\quad p = \text{A*}(\alpha)$
8. $\quad$ **if** $(\theta(p) == K)$ return p **end if**
9. **end while**
10. $\alpha_{max} = \alpha$
11. **while** $((\alpha_{max} - \alpha_{min}) > \varepsilon)$
12. $\quad \alpha = (\alpha_{max} - \alpha_{min})/2$
13. $\quad p = \text{A*}(\alpha)$
14. $\quad$ **if** $(\theta(p) \leq K)$
15. $\quad\quad \alpha_{max} = \alpha$
16. $\quad\quad \hat{p}^* = p$
17. $\quad\quad \alpha^* = \alpha_{max}$
18. $\quad$ **else**
19. $\quad\quad \alpha_{min} = \alpha$
20. $\quad$ **end if**
21. $\quad$ **if** $(\theta(p) == K)$ return p **end if**
22. **end while**
23. return $\hat{p}^*$

Figure 2: PSA Pseudocode

Theorem 6. *Suppose that for α_x, $p^*_{\alpha_x}$ is the shortest path having K neutralizations. Then, $p^*_{\alpha_x}$ is the optimum path for the (s, t, D, C, K) problem. That is, $\tau(p^*_{\alpha_x}, \alpha_1) = \tau(p^*, \alpha_1)$.*

Proof. For a contradiction assume that $\tau(p^*_{\alpha_x}, \alpha_1) > \tau(p^*, \alpha_1)$. Then $\tau(p^*_{\alpha_x}, \alpha_x) > \tau(p^*, \alpha_x)$ because both $p^*_{\alpha_x}$ and p^* require the same number of neutralizations. But, this contradicts the fact that $p^*_{\alpha_x}$ is the optimum path associated with α_x. □

As an example, Figure 3 illustrates paths found by PSA with $C = 1$ on a discretized actual naval minefield data set with 39 disks, called the COBRA data (this data set is explained in more detail in Section 4). Figures 3(a), 3(b), 3(c), and 3(d) show the paths found by PSA for $K = 3, 2, 1$, and 0 respectively. Note that K, the number of neutralizations in ONP, corresponds to the number of resources in WSCPP. In this particular case, when $C = 1$, it turns out PSA finds the optimal path regardless of K.



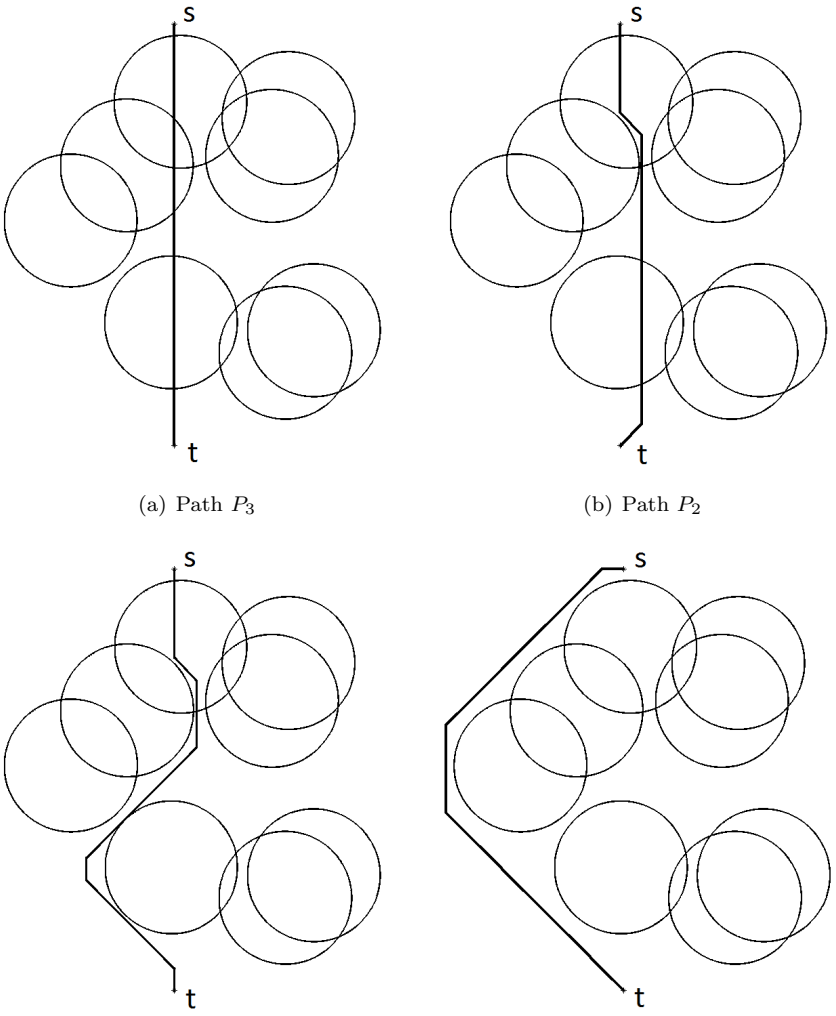
(a) $K = 3$ ($\alpha^* = 1$) (b) $K = 2$ ($\alpha^* = 3$)

(c) $K = 1$ ($\alpha^* = 4$) (d) $K = 0$ ($\alpha^* = 29$)

Figure 3: Paths found by PSA on a discretized naval minefield data set for 4 different neutralization limits. For each one of these paths, the number of neutralizations required is the same as the neutralization limit, and therefore they are optimal.

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. As an illustration, a simple discretized ONP instance with 8 disks is depicted in Figure 4 along with the optimum paths for K ranging from 0 to 3, with $C = 0.8$. Fine-tuning of the α values is shown in Table 1 for $K = 1$. As the table illustrates, no matter how fine α is tuned, PSA cannot find a path with exactly one neutralization.

We now offer some intuition why as to this is the case. In Figure 4, let P_K denote the optimal path that requires exactly K neuralizations. Lengths of these paths are $\ell(P_3) = 19.0$, $\ell(P_2) = 19.8284$, $\ell(P_1) = 23.1421$ and $\ell(P_0) = 26.2132$ respectively. Observe that PSA returns P_2 as the best path for α satisfying $\ell(P_3) + \alpha \times C \times \theta(P_3) = \ell(P_2) + \alpha \times C \times \theta(P_2)$. Solving that equation yields $\alpha = 1.0355$. Thus, for $\alpha > 1.0355$, P_2 is the best path. Similarly, to find the α value that gives P_1 as the best path, we solve $\ell(P_2) + \alpha \times C \times \theta(P_2) = \ell(P_1) + \alpha_k \times C \times \theta(P_1)$. So, P_1



(a) Path P_3 (b) Path P_2

(c) Path P_1 (d) Path P_0

Figure 4: A simple discretized ONP instance with 8 disks and the optimal paths for 4 different neutralization limits. P_K denotes the optimal path for K neutralizations. PSA cannot find a path with exactly one neutralization for this case. See Table 1.

is expected to become the best path for $\alpha > 4.1421$. However, observe that $\tau(P_1, 4.1421) = 26.4558 > 26.2132 = \tau(P_0, 4.1421)$ which means that P_0 is the best (for $\alpha > 3.9905$). Hence, there are no intervals that manifests P_1 as the best path.

4. Computational Experiments

This section presents comprehensive computational experiments comparing the performance of PSA to two other algorithms on both real and synthetic data: (1) adaptation of another popular suboptimal WCSPP algorithm to ONP, called Delay-Constrained Unicast Routing (DCUR) Algorithm [19], and (2) an exact algorithm for ONP. Our choice of the DCUR Algorithm is due to its proven success in similar problem environments [1]. The exact WCSPP algorithm we utilize is the one proposed in [16].

Table 1: Fine-tuning α for the ONP instance depicted in Figure 4 with $K = 1$. No matter how fine α is tuned, PSA cannot find a path with exactly one neutralization.

α	$\tau(p, \alpha)$	$\theta(p, \alpha)$
1	21.8284	2
10	26.2132	0
6	26.2132	0
4	26.2132	0
3	21.8284	2
3.5	26.2132	0
3.25	26.2132	0
3.125	21.8284	2
3.1875	21.8284	2
3.21875	26.2132	0
3.203125	26.2132	0
3.1953125	26.2132	0
3.19140625	21.8284	2
3.193359375	26.2132	0
3.1923828125	21.8284	2
3.19287109375	26.2132	0
3.192626953125	26.2132	0
3.1925048828125	26.2132	0
3.19244384765625	26.2132	0
3.19241333007812	26.2132	0
3.19239807128906	26.2132	0
3.19239044189453	26.2132	0
3.19238662719726	21.8284	2
3.19238853454589	26.2132	0

The specific application domain we consider 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. [32] and was later referred to in [33, 34, 35, 36, 37, 38, 39, 40]. The COBRA data is illustrated in Figure 5 and tabulated in Table 2. For simplicity, original data coordinates were scaled and shifted so that disk centers are inside the region $[10, 90] \times [10, 90]$. The starting point is $s = (54, 80)$ and the destination point is $t = (54, 10)$ with disk radius taken as $r = 5$. Throughout our experiments, we work with an 8-regular lattice discretization of the obstacle field for convenience where straight edge lengths are taken as 1 and diagonal edge lengths are taken as $\sqrt{2}$.

4.1. Outline of Experiments and Performance Metrics

An outline of our computational experiments is presented below.

- Section 4.2 compares the three algorithms on the COBRA data set for various C and K combinations.
- Section 4.3 compares the algorithms on simulated COBRA-like instances, again for different C and K combinations. In these instances, there are 100 disks with $r = 5$ on a [0,100]×[0,100] obstacle field where $s = (50, 100)$ and $t = (50, 1)$.

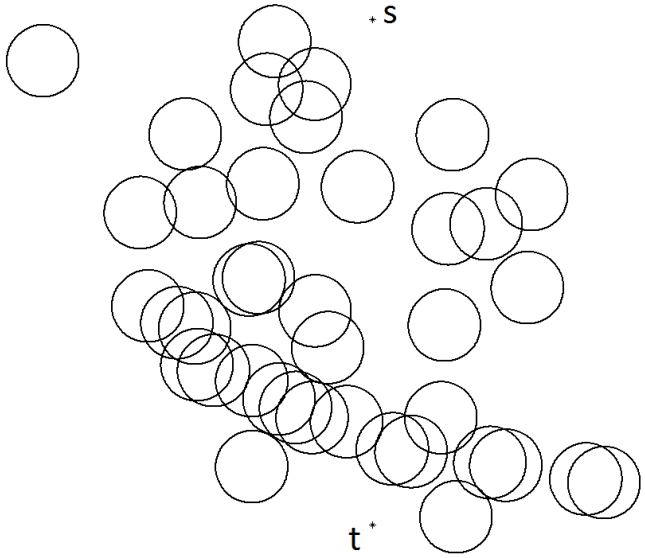


Figure 5: An actual naval minefield data set, called the COBRA data.

Table 2: Scaled and shifted center coordinates of COBRA disks.

X-coordinate	**Y-coordinate**	**X-coordinate**	**Y-coordinate**	**X-coordinate**	**Y-coordinate**
46.13	39.61	50.49	24.26	83.62	16.33
30.21	54.62	56.83	20.50	44.87	66.45
47.88	34.51	40.55	76.93	43.43	26.22
21.93	53.22	69.82	51.65	65.64	11.08
37.36	29.94	29.47	37.21	59.42	20.11
38.90	57.22	32.07	31.37	45.71	24.83
86.12	15.83	52.01	56.80	41.14	27.41
8.43	74.26	37.00	43.89	72.53	18.22
22.98	40.29	70.33	18.61	29.78	32.15
63.54	24.81	64.04	37.65	27.00	37.97
46.07	71.00	65.16	64.01	37.36	18.03
39.43	70.31	75.51	42.83	76.11	55.73
38.29	44.20	28.16	64.10	64.55	50.98

- Section 4.4 gives a comparison on three different simulated instance types: (1) 200 disks with $r = 7$ on [0,200]×[0,200] where $s = (100, 200)$ and $t = (100, 1)$, (2) 400 disks with $r = 10$ on [0,400]×[0,400] where $s = (200, 400)$ and $t = (200, 1)$, and (3) 1000 disks with $r = 16$ on [0,1000]×[0,1000] where $s = (500, 1000)$ and $t = (500, 1)$.

- Section 4.5 compares the algorithms on simulated instances with 100, 200, 400, and 1000 disks respectively with $r = 5$ on a [0,100]×[0,100] obstacle field where $s = (50, 100)$ and $t = (50, 1)$.

The performance metrics we report are as follows:

- Total Path Cost (TotCost): The total cost of the optimal $s - t$ path found by the exact algorithm. It is comprised of the sum of Euclidean lengths of all the edges on the path and the total cost of neutralizations.

- Lower Bound and Upper Bound (%DualDev and %PriDev): Percent deviation of the lower and upper bound of the LP relaxation obtained by an LP solver.
- Number of Neutralizations (#Neut): This value is reported for all three algorithms.
- Deviation From the Optimal Path Cost (%Dev): Percent deviation from the optimal path total cost value. This value is reported for PSA and the DCUR Algorithm.
- Number of A* Calls (#A*): This value is reported only for PSA.
- Run Time (RT): Execution time of the algorithms in seconds (RT of the exact algorithm corresponds to the one in [16]).

In each section and in each comparison, each reported performance metric is the average over 100 random instances. After a set of preliminary tests we observed that using 0.5 for ε is best. The tests were performed on a personal computer with an Intel Xeon E5 Processor running at 3.1 GHz clock speed and with 128 GB memory. The MATLAB codes of the algorithm as well as the tested examples can be accessed from `http://mimoza.marmara.edu.tr/~falkaya/research.htm`.

4.2. Comparison on the COBRA Data

In this section we compare the performance of the algorithms on COBRA data. The results for various C values are given in Table 3. The number of neutralizations ($\#Neut$) for both PSA and DCUR is $K = 1$. Observe that for $C = 0.2$, 0.5, 1 and 2, even though DCUR makes only 1 neutralization, its path is far from optimal which means that it does not neutralize the correct disc. However for $C = 5$, it chooses the correct disc for neutralization and uses a path very close to optimal. On the other hand, PSA neutralizes the correct disc for all values of C and finds the optimum path in less time than DCUR and exact algorithm.

Table 3: Comparison of algorithms for various C on COBRA data for K=1. TotCost denotes total path cost, #Neut denotes number of neutralizations, %Dev denotes deviation from the optimal, #A* denotes number of A* calls, and RT denotes run time in seconds.

C	**Exact**			**PSA**			**DCUR**	
	TotCost	**#Neut**	**RT**	**%Dev**	**#A***	**RT**	**%Dev**	**RT**
0.2	75.999	1	2.61	0.00%	3	0.090	58.48%	0.095
0.5	76.299	1	2.55	0.00%	2	0.071	58.51%	0.090
1	76.799	1	2.52	0.00%	2	0.070	57.48%	0.089
2	77.799	1	2.51	0.00%	2	0.069	56.74%	0.090
5	80.799	1	2.40	0.00%	1	0.039	1.03%	0.090

Results for various values of K are given in Table 4. The results indicate that PSA finds the optimum easily whereas DCUR is far away from the optimum for $K = 0$ and $K = 1$. Overall, poor performance of DCUR can perhaps be attributed to its inherent greedy design of choosing least cost paths over least weight ones.

Table 4: Comparison of algorithms for various K on COBRA data for C=1.

K	Exact			PSA				DCUR		
	TotCost	#Neut	RT	%Dev	#Neut	#A*	RT	%Dev	#Neut	RT
0	104.3259	0	2.45	0.00%	0	3	0.089	18.14%	0	0.080
1	76.7990	1	2.52	0.00%	1	2	0.070	57.48%	1	0.080
2	74.4853	2	2.55	0.00%	2	5	0.130	0.00%	2	0.081
3	73.0	3	2.57	0.00%	3	1	0.036	0.00%	3	0.081

4.3. Comparison with Different C and K Values on COBRA-Like Data

This section compares the algorithms with respect to various C and K combinations on COBRA-like instances with 100 disks. In particular, $s = (50, 100)$, $t = (50, 1)$, $r = 5$, and 100 disk centers were generated randomly from the uniform distribution on [5,95]×[5,95]. The average results for 100 random instances are given in Table 5 for $K = 5$[2]. We observe that PSA outperforms DCUR and runs in much less time than the exact algorithm. Another observation is the decrease in number of neutralizations on the paths for increasing neutralization cost values. This is reasonable since for higher neutralization costs, paths with fewer number of neutralizations are preferred. Similarly, for both algorithms' total cost increases as neutralization cost increases.

In our computational results, we provide the percent deviation of the lower and upper bounds of the LP-relaxation using the well-known SCIP solver [41]. In Table 5 and subsequent tables, it can be seen that the percent deviation of the lower and upper bounds from the optimal are significantly greater than that of PSA. Furthermore, the required run time for obtaining these bounds are drastically greater than PSA run time, which are 5, 213, 5038, 115223 seconds on the average for [0,100]×[0,100], [0,200]×[0,200], [0,400]×[0,400], [0,1000]×[0,1000] sized obstacle fields, respectively. As can be seen in the same tables, %PriDev is greater that %DualDev in general. An intuition behind this difference is that the primal integer programs (IPs) have fewer variables and more constraints compared to their duals. Consequently, linear programming (LP) relaxations of the primal IPs tend to deviate more from the optimal when compared to the corresponding duals.

Table 5: Comparison of algorithms for various C on COBRA-like instances with 100 disks for K=5.

C	Exact					PSA				DCUR		
	TotCost	%DualDev	%PriDev	#Neut	RT	%Dev	#Neut	#A*	RT	%Dev	#Neut	RT
0.1	105.08	6.16%	40.76%	4.86	3.13	1.09	3.89	7.16	0.12	54.57	4.93	0.065
0.2	105.57	5.12%	51.90%	4.86	3.17	1.00	3.89	5.37	0.11	55.34	4.87	0.064
0.5	107.01	4.74%	54.13%	4.76	3.03	0.75	3.81	5.36	0.12	47.93	4.84	0.061
1	109.29	2.64%	62.55%	4.19	2.78	0.43	3.73	4.31	0.11	22.44	4.43	0.062
2	113.13	2.50%	63.28%	3.45	2.48	0.15	3.31	1.93	0.08	6.58	3.53	0.064
5	120.89	2.02%	63.43%	2.08	2.41	0.04	2.04	1.06	0.07	0.29	2.08	0.064
Mean	110.16	3.86%	56.01%	4.03	2.83	0.56	3.45	4.2	0.10	30.16	4.11	0.063

In Table 6 results regarding the performance of the algorithms with respect to various K are given where $C = 1$. The total cost performance of all algorithms get better with increasing K values and when $K = 10$ all of the

[2]For a relatively low K value, the optimum path mostly walks around the disks. For a relatively high K value, on the other hand, the optimum path turns out to be almost a straight line. We observed that $K = 5$ stands as the most challenging value for the algorithms to show their performance.

algorithms find the optimal path for all of the 100 minefield instances. This is because of the fact that when it is allowed to make 10 neutralizations the shortest path from s to t has fewer neutralizations and hence it is the optimal path. On the average, PSA outperforms DCUR and runs in much less time than the exact algorithm. In addition, it can be observed in Tables 5 through 8 that the solutions obtained by PSA are very close to the optimal in general, if not optimal.

Table 6: Comparison of algorithms for different K values on COBRA-like instances with 100 disks for $C = 1$.

K	**Exact**					**PSA**				**DCUR**		
	TotCost	**%DualDev**	**%PriDev**	**#Neut**	**RT**	**%Dev**	**#Neut**	**#A***	**RT**	**%Dev**	**#Neut**	**RT**
1	126.46	7.10%	26.58%	0.99	2.79	1.77%	0.72	5.04	0.14	60.31%	1	0.077
2	117.52	6.06%	36.57%	1.88	2.83	2.19%	1.48	4.93	0.14	56.13%	1.91	0.076
3	112.33	5.03%	46.65%	2.74	2.86	0.65%	2.38	4.47	0.12	52.38%	2.82	0.076
4	110.07	3.94%	58.19%	3.56	2.87	0.45%	3.15	4.37	0.12	38.94%	3.69	0.077
5	109.29	2.64%	62.55%	4.19	2.78	0.43%	3.73	4.31	0.11	22.44%	4.43	0.078
6	108.71	2.36%	65.94%	4.82	2.80	0.27%	4.4	3.21	0.09	10.68%	5.03	0.078
7	108.36	1.94%	67.42%	5.24	2.67	0.20%	4.91	2.33	0.08	6.68%	5.42	0.079
8	108.19	0.66%	68.09%	5.53	2.56	0.08%	5.4	1.48	0.06	1.35%	5.64	0.079
9	108.16	0.50%	69.08%	5.64	2.51	0%	5.61	1.12	0.06	0.03%	5.73	0.080
10	108.15	0.43%	69.41%	5.75	2.50	0%	5.75	1.00	0.05	0%	5.75	0.080
Mean	111.72	3.06%	57.05%	4.03	2.73	0.64%	3.75	3.23	0.10	25.85%	4.14	0.078

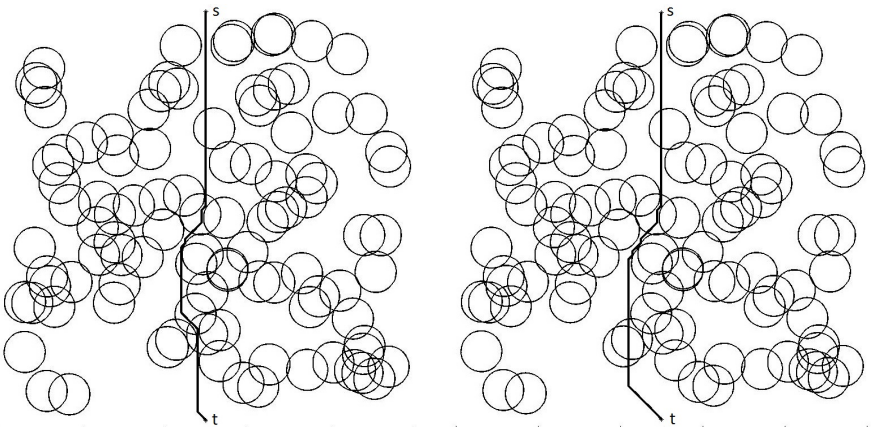
Figure 6 shows the outputs of the algorithms on one of the random instances. Observe that PSA finds a path with four neutralizations (Figure 6(b)) whose total cost is only 0.60% away from optimum (Figure 6(a)). On the other hand, DCUR finds a path with five neutralizations, yet 10% away from optimal total cost (Figure 6(c)).

4.4. Comparison with Different Obstacle Field Sizes

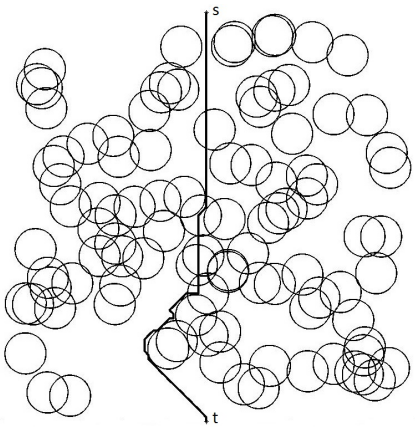
The third performance comparison is with respect to three different instance types: (1) 200 disks with $r = 7$ on [0,200]×[0,200], (2) 400 disks with $r = 10$ on [0,400]×[0,400], and (3) 1000 disks with $r = 16$ on [0,1000]×[0,1000]. 100 random instances are generated for each combination of instance type and values of C. Comparison results are given in Table 7. Observe that the results are similar to the those with 100 disks. However an important observation here is the scalability of PSA when compared with DCUR and exact algorithm. Execution time of DCUR and exact algorithm increases faster than PSA for larger graphs as seen in the table.

4.5. Comparison with Different Disk Quantities

The last comparison of the algorithms is with respect to different number of disks on a [0,100]×[0,100] obstacle field with $r = 5$, $C = 1$, and $K = 5$. We generate 100 random instances with 200, 300, 400, and 1000 disks respectively. The results are presented in Table 8. In the table, mean deviation from optimal for PSA is 2.87%, mean deviation for DCUR is 28.46%. Notice that average total path cost for all algorithms increase as disk quantity increases. For instances with 1000 disks, the paths have a total cost close to 200 because with such a high obstacle quantity, the optimal path tends to be the one with no neutralizations at all. That is, the optimal path tends to be the one that traverses along the outer boundaries of the obstacle field without any neutralizations.



(a) Optimal path (b) PSA path



(c) DCUR path

Figure 6: Paths found by the algorithms on a COBRA-like instance with 100 disks.

Table 7: Comparison of algorithms with 200, 400, and 1000 disks on [0,200] × [0, 200], [0, 400] × [0, 400] and [0, 1000] × [0, 1000] obstacle fields respectively with $K = 5$.

Size	**C**	**Exact**					**PSA**			**DCUR**		
		TotCost	**%DualDev**	**%PriDev**	**#Neut**	**RT**	**%Dev**	**#Neut**	**RT**	**%Dev**	**#Neut**	**RT**
[200]×[200]	0.2	212.86	4.02%	7.41%	4.99	12.9	0.87%	4.14	0.27	58.48%	4.93	0.156
	0.5	214.36	3.99%	8.54%	4.97	13.0	0.7%	4.16	0.24	57.38%	4.93	0.156
	1	216.81	3.96%	9.13%	4.85	13.1	0.61%	4.08	0.28	39.23%	4.94	0.156
	2	221.5	3.15%	10.12%	4.43	11.6	0.41%	3.85	0.24	18.57%	4.61	0.156
	5	232.04	3.13%	13.44%	2.7	9.6	0.08%	2.62	0.13	1.76%	2.71	0.156
	Mean	219.51	3.65%	9.73%	4.39	12.0	0.52%	3.77	0.23	34.42%	4.42	0.156
[400]×[400]	0.5	430.09	5.22%	5.20%	4.99	56.4	0.52%	4.24	0.76	68.72%	4.92	0.857
	1	432.58	4.87%	6.39%	4.96	57.2	0.5%	4.15	0.77	61.1%	4.96	0.865
	2	437.49	4.50%	7.03%	4.83	57.6	0.36%	4.09	0.95	43.65%	4.89	0.879
	5	450.86	3.56%	7.87%	3.84	46.9	0.08%	3.56	0.45	6.85%	3.9	0.890
	10	465.91	2.02%	9.21%	2.51	43.0	0%	2.51	0.21	0%	2.51	0.877
	Mean	443.39	4.03%	7.14%	4.23	52.2	0.29%	3.71	0.63	35.26%	4.24	0.874
[1000]×[1000]	1	1089.12	5.13%	4.13%	5	411.7	0.46%	4.5	5.75	65.91%	5	9.731
	2	1094.12	5.01%	4.57%	5	415.3	0.41%	4.5	5.20	60.37%	5	9.797
	5	1109.12	4.74%	6.60%	5	432.3	0.29%	4.4	6.54	29.56%	5	9.915
	10	1133.1	4.68%	7.77%	4.6	372.3	0.14%	4	4.64	11.86%	4.6	9.943
	Mean	1106.36	4.89%	5.77%	4.9	407.9	0.32%	4.35	5.53	41.59%	4.9	9.846

Table 8: Comparison of algorithms with respect to disk quantities of 200, 300, 400, and 1000 on a [0,100]×[0,100] obstacle field with $r = 5$, $C = 1$, and $K = 5$.

No.	Exact					PSA				DCUR		
	TotCost	%DualDev	%PriDev	#Neut	RT	%Dev	#Neut	#A*	RT	%Dev	#Neut	RT
100	109.29	2.64%	62.55%	4.19	2.78	0.43%	3.73	4.31	0.11	22.44%	4.43	0.068
200	153.43	11.14%	16.64%	4.92	4.61	8.90%	1.36	6.04	0.20	57.51%	4.54	0.091
300	176.97	10.60%	9.16%	4.62	3.55	3.23%	0.51	6.01	0.22	35.32%	4.40	0.105
400	182.32	5.72%	5.06%	3.90	2.99	1.75%	0.40	6.52	0.29	25.11%	4.25	0.131
1000	188.74	0.62%	0.72%	2.05	2.67	0.04%	1.69	2.62	0.18	1.93%	2.52	0.185

5. Summary and Conclusions

We study the obstacle neutralization problem (ONP) where the goal is to safely and swiftly navigate an agent from a given source location s to a given destination t through an arrangement of disk-shaped 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 utilize this neutralization capability for the shortest $s - t$ traversal. We show that ONP is a variant of the intractable weight constrained shortest path problem (WCSPP) and we provide a detailed literature review on solutions and applications of WCSPP. Then we propose the Penalty Search Algorithm (PSA) for ONP which is a simple, effective, and scalable suboptimal algorithm for ONP. PSA is guaranteed to give the optimal path in case it manages to fully utilize the neutralization capability. Performance of PSA is compared to ONP adaptation of another high-performing heuristic for WCSPP as well as an exact method on real and synthetic ONP instances from a naval minefield application domain. We present comprehensive computational experiments which suggest that PSA finds solutions that compare very favorably to the optimal paths with minimal computational resources, whereas the other heuristic method struggles to find good solutions in general. An attractive feature of PSA is that it lends itself to easy adaptation to general instances of WCSPP. Thus, as future work, we intend to adapt PSA to other variants of WCSPP and benchmark against competing methods.

Acknowledgments

Work of A.F. Alkaya was supported by The Scientific and Technological Research Council of Turkey (TUBITAK), Project No. 1059B191100310. Work of V. Aksakalli was supported by TUBITAK Grant No. 111M541. The authors would like to thank Dr. Ranga Muhandiramge for several helpful discussions and also for sharing with us his computer code for exact solution of ONP. The authors would like to thank the anonymous reviewer and the area editor for dedicating much time and effort for a comprehensive review process that resulted in a significant improvement of this manuscript.

References

[1] F. Kuipers, T. Korkmaz, M. Krunz, P. V. Mieghemt, Performance evaluation of constraint-based path selection algorithms, IEEE Network 18(1) (2004) 16–23.

[2] M. Zabarankin, S. Uryasev, P. Pardalos, Optimal Risk Path Algorithms, volume 66, Kluwer Academic, Dordrecht, 2002.

[3] G. Dahl, B. Realfsen, Curve approximation and constrained shortest path problems, in: International Symposium on Mathematical Programming (ISMP97), 1997.

[4] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Co, 1979.

[5] R. Muhandiramge, Maritime manouvering optimization: path planning in minefield threat environments, Ph.D. thesis, University of Western Australia, 2008.

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

[7] W. Carlyle, R. Wood, Lagrangian relaxation and enumeration for solving constrained shortest-path problems, in: 38th Ann. ORSNZ Conf., Hamilton, New Zealand, University of Waikato, 2003.

[8] J. Beasley, N. Christofides, An algorithm for the resource constrained shortest path problem, Networks 19 (1989) 379–394.

[9] K. Mehlhorn, M. Ziegelmann, Resource constrained shortest paths, in: 8th Annual European Symposium on Algorithms (ESA), LNCS 1879, 2000, pp. 326–337. `http://www.ziegelmann.info/CNOP.htm`.

[10] M. Ziegelmann, Constrained Shortest Paths and Related Problems, Ph.D. thesis, Fakultät für Informatik, Saarbrücken, Germany, 2001.

[11] S. Storandt, Route planning for bicycles - exact constrained shortest paths made practical by construction hierarchy, in: 22th ICAPS, 2012, pp. 234–242.

[12] S. Storandt, Contraction hierarchies in grid graphs, in: 36th Annual Conference on Artificial Intelligence, volume LNCS 8077, Koblenz, Germany, 2013, pp. 236 – 247.

[13] I. Dumitrescu, N. Boland, Algorithms for the weight constrained shortest path problem, Intl. Trans. in Op. Res. 8 (2001) 15–29.

[14] I. Dumitrescu, N. Boland, Improved preprocessing, labelling and scaling algorithms for the weight-constrained shortest path problem, Networks 43 (2003) 135–153.

[15] G. Xue, Primal dual algorithms for computing weight-constrained shortest paths and weight-constrained minimum spanning trees, in: 19th IEEE Performance, Computing and Communications Conference (IPCCC), 2000, pp. 271 – 277.

[16] R. Muhandiramge, N. Boland, Simultaneous solution of lagrangean dual problems interleaved with preprocessing for the weight constrained shortest path problem, Networks 53 (2009) 358–381.

[17] A. Jüttner, B. Szviatovski, I. Mécs, Z. Rajkó, Lagrange relaxation based method for the QoS routing problem, in: Proc. 20th Ann. Joint Conf. IEEE Comput Commun Societies, volume 2, 2001, pp. 859–868.

[18] A. Jüttner, Parametric Problems in Combinatorial Optimization and Their Applications in Telecommunications, Ph.D. thesis, Eötvös University, Faculty of Science, Budapest, Hungary, 2006.

[19] D. Reeves, H. Salama, A distributed algorithm for delay-constrained unicast routing, IEEE/ACM Trans. Net. 8 (2000) 239–250.

[20] L. Guo, I. Matta, Search space reduction in QoS routing, Computer Networks 41 (2003) 73–88.

[21] S. Lee, Route Optimization Model for Strike Aircraft, Master's thesis, Naval Postgraduate School, Monterey, California, 1995.

[22] J. Latourell, B. Wallet, B. Copeland, Genetic algorithm to solve constrained routing problem with applications for cruise missile routing, in: Proceedings of SPIE, 3390, 1998, pp. 490–500.

[23] M. Zabarankin, S. Uryasev, R. Murphey, Aircraft routing under the risk of detection, Nav. Res. Logist. 53 (2006) 728–747.

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

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

[26] R. Muhandiramge, N. Boland, S. Wang, Convergent network approximation for the continuous euclidean length constrained minimum cost path problem, SIAM Journal on Optimization 20 (2009) 54–77.

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

[28] R. Borndörfer, M. Grötschel, A. Löbel, Scheduling duties by adaptive column generation, Technical Report (ZIB-Report) 01-02, Konrad - Zuse - Zentrum für Informationstechnik Berlin (ZIB), Berlin, 2001.

[29] C. Barnhart, N. Boland, L. Clarke, E. Johnson, G. Nemhauser, R. Shenoi, Flight string models for aircraft fleeting and routing, Transportation Science 32 (1998) 208–220.

[30] J. Bekker, J. Schmid, Planning the safe transit of a ship through a mapped minefield, ORION 22 (2006) 1–18.

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

[32] N. Witherspoon, J. Holloway, K. Davis, R. Miller, A. Dubey, The coastal battlefield reconnaissance and analysis (COBRA) program for minefield detection, in: Proceedings of the SPIE: Detection Technologies for Mines and Minelike Targets, Orlando, Florida, volume 2496, 1995, pp. 500–508.

[33] C. Priebe, T. Olson, D. H. Jr., Exploiting stochastic partitions for minefield detection, in: Proceedings of the SPIE, 3079, 1997, pp. 508–518.

[34] C. Priebe, D. Fishkind, L. Abrams, C. Piatko, Random disambiguation paths for traversing a mapped hazard field, Nav. Res. Logist 52 (2005) 285–292.

[35] D. Fishkind, C. Priebe, K. Giles, L. Smith, V. Aksakalli, Disambiguation protocols based on risk simulation, IEEE Trans. on Systems, Man, and Cybernetics, Part A 37(5) (2007) 814–823.

[36] X. Ye, C. Priebe, A graph-search based navigation algorithm for traversing a potentially hazardous area with disambiguation, Internat. J. Oper. Res. and Information Sys. 1 (2010) 14–27.

[37] X. Ye, D. Fishkind, L. Abrams, C. Priebe, Sensor information monotonicity in disambiguation protocols, J. Oper. Res. Society 62 (2011) 142–151.

[38] V. Aksakalli, D. Fishkind, C. Priebe, X. Ye, The reset disambiguation policy for navigating stochastic obstacle fields, Nav. Res. Logist. 58 (2011) 389–399.

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

[40] V. Aksakalli, I. Ari, Penalty-based algorithms for the stochastic obstacle scene problem, INFORMS Journal on Computing, DOI: 10.1287/ijoc.2013.0571 (2013).

[41] T. Koch, Rapid Mathematical Prototyping, Ph.D. thesis, Technische Universität Berlin, 2004.