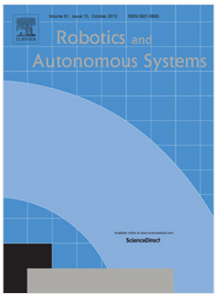
Accepted Manuscript



Bi-objective path planning using deterministic algorithms

Mansoor Davoodi

PII: S0921-8890(16)30075-6
DOI: http://dx.doi.org/10.1016/j.robot.2017.03.021
Reference: ROBOT 2822

To appear in: *Robotics and Autonomous Systems*

Received date : 10 February 2016
Revised date : 10 March 2017
Accepted date : 30 March 2017

Please cite this article as: M. Davoodi, Bi-objective path planning using deterministic algorithms, *Robotics and Autonomous Systems* (2017), http://dx.doi.org/10.1016/j.robot.2017.03.021

Bi-objective Path Planning using Deterministic Algorithms

Mansoor Davoodi[a,b]

[a] *Department of Computer Sciences and Information Technology, Institute for Advances Studies in Basic Sciences, Zanjan, Iran.*
[b] *School of Computer Science, Institute for Research in Fundamental Sciences (IPM)*
mdmonfared@iasbs.ac.ir

Abstract

Path planning has become a central problem in motion planning. The classic version of the problem aims to find an obstacle-free path with the minimum length for a given workspace containing a set of obstacles and two sources and destination points. However, some real world applications consider maximizing the path clearance (i.e., the distance between the robot and obstacles) as the secondary objective. This bi-objective path planning problem has been studied using evolutionary and other heuristic algorithms which do not guarantee achieving Pareto optimal paths. In this paper, we first study this problem using deterministic algorithms. Next, we propose an efficient algorithm for the problem in the grid workspace. We then propose an $O(n^3)$ time algorithm for the problem under the Manhattan distance in a continuous workspace containing n vertical segments as obstacles. Finally, we show the obtained solutions are proper approximation for the problem under the Euclidean distance.

Keywords: Bi-objective optimization, Path planning, Clearance, Pareto optimal, Approximation.

1 Introduction

Robot Path Planning (PP) problem as a practical problem in industrial and daily routine work is an interesting and challenging problem in computer science and robotics. In the classic PP problem, a workspace containing obstacles and two *source* and *destination* points, denoted by *s* and *d,* are given and the goal is to find an obstacle-free path for a robot starting from *s* and ending at *d*, denoted by *s-d-path.* Most of studies in this literature have focused on finding the shortest *s-d*-path [1, 2]. There are numerous parameters like shape, dynamic, abilities and constraints of robot, dimension and characteristic of workspace that are important in the exact definition of PP problem and its complexity. Consequently, various versions of the problem like static\dynamic, holonomic\non-holonomic, continuous\discrete, known\unknown or smart workspace [3] have been studied. With regard to these varieties and the complexity of PP problem which is NP-hard in general, several types of approaches have been proposed for solving it. *Cell decomposition* [4], *sampling roadmaps*, *potential fields* [1, 2, 5] and heuristic

algorithms like particle swarm optimization [6], colony system [7], neural networks [8], and fuzzy controllers [9, 10] can be reviewed as such approaches.

Visibility Graph (VG) of obstacles can be used for converting a PP problem from continuous space to a discrete graph searching problem [11]. VG is a graph whose nodes are obstacle's vertices, and there exists an edge between two nodes if and only if the corresponding vertices are *visible* to each other. If the weight of each edge is equal to its endpoints' distance, VG is a weighted connected graph and the shortest path can be found by a graph searching algorithm (e.g., Dijkstra [12]). The VG of a set of obstacles in the plane with total number of vertices n can be computed in $O(n^2 \log n)$ time by using a search tree structure and ray shooting technique [13]. The shortest path on VG can be computed in $O(n \log n + m)$ time by using Dijkstra's algorithm implemented by Fibonacci heap, where m is the number of edges in VG. Several approaches for computing VG and the shortest path can be found in [11, 14]. Welzl et al. [15] proposed $O(n^2)$ time and space algorithms for computing the VG of n segments, and Hershberger et al. [16] proposed an $O(n \log n)$ time algorithm to solve the PP problem in the plane. While PP problem in two dimensional space is polynomial, it is NP-hard in three dimensional space [17], however, there are polynomial algorithms for special cases of PP problem like *polyhedral terrain* obstacles with Manhattan distance [18].

All the aforementioned studies in PP problem took into account just the objective minimizing the length of *s-d*-path, however the *clearance* objective was also studied in the literature. Clearance is related to the *safeness* of the robot and obstacles, so, the goal is finding a path with maximum clearance, i.e. maximizing the distance between the robot's path and obstacles. Therefore, considering simultaneously both minimizing the length of the path and maximizing the clearance of the path is more practical. Elshamli et al. [19] used a linear combination of the objectives and proposed an evolutionary algorithm for solving it. Castillo et al. [20] studied the PP problem on a grid with the assumption that a predefined difficulty weight for each cell of the grid is given. They presented a genetic algorithm based on a non-dominated ranking procedure. Ahmed and Deb [21] used non-dominated sorting genetic algorithm II (NSGA-II) for solving the PP problem with the objectives travel distance and safety. Davoodi et al. [22, 23] considered the objectives minimizing the length of *s-d*-path and maximizing its clearance and smoothness in the discrete and continuous workspaces. They proposed geometric search operators and algorithms based on NSGA-II framework for finding Pareto optimal solutions. Wein [24] used *Voronoi diagram* and VG, and introduced the combined structure *Visibility-Voronoi Diagram* with clearance c. This structure is able to find a shortest *s-d*-path for a given clearance c. Voronoi diagram is the locus of all points in the plane that have more than one nearest obstacle. Geraerts [25] introduced a structure called *Explicit Corridor Map* to find a short path with a maximum

clearance that can be smoothed after a post-processing phase. Further, it is possible to solve the PP problem with a lexicographic order of the objectives, i.e. first some high clearance *s*-*d*-paths are founded and then by performing a post-processing phase, improve them in terms of length or smoothness [26].

In this paper we study the problem of path planning in the discrete and continuous workspaces and similar to the most of studies in the literature we consider two objectives minimizing the length of the path and maximizing the clearance, that is, maximizing the minimum distance of the path from the nearest obstacles. We formulate the problem in a bi-objective optimization model and propose deterministic algorithms for finding its Pareto optimal fronts. To the best of our knowledge there is no study in the literature that focuses on the complexity of Pareto optimal front(s) using deterministic algorithms (instead of heuristic approaches). Generally, since Pareto optimal solutions are not in polynomial size with respect to the input of PP problem, proposing polynomial algorithms for computing all of them is impossible. There are two general approaches in such situations. First, applying heuristic methods, such as evolutionary algorithms, that attempt to find a small diverse set of Pareto optimal solutions [27, 28], and approximating Pareto optimal solutions using a polynomial size set [29- 32]. As the point of complexity view, the second type of approaches is much more interesting. In this paper we show that the Pareto optimal fronts of the bi-objective PP problem in the grid workspace can be found in polynomial time. Also, we consider the query version of the problem, that is, the destination point of the robot is not as a predefined point; it is given in the query phase. Further, we study the PP problem in continuous workspace under the Manhattan metric, and propose an $O(n^3)$ time algorithm where obstacles are n vertical segments. Finally we discuss about the Euclidean metric and present an approximation solution for it.

This paper is organized in five sections. In Section 2 multi-objective optimization and related concepts is brief introduced. In section 3 an efficient algorithm is proposed for the problem of bi-objective PP in a grid workspace. A limited version of the problem for the vertical segments as obstacles under the Manhattan metric is studied in Section 4. Further, in this section an efficient algorithm for finding all the Pareto optimal *intervals* is proposed, and the approximation for the Euclidean metric is discussed. Finally conclusion and future work are drawn in Section 5.

2 Multi-objective optimization

A multi-objective optimization (minimization) problem can be formulated as the following model [27]:

$$\begin{cases} \textbf{\textit{Minimize}} \quad f_i(X), \qquad \text{for } i = 1,2,\dots,M \\ \\ subject\ to: \\ \quad g_j(X) \le 0, \qquad \text{for } j = 1,2,\dots,J \\ \quad h_j(X) = 0, \qquad \text{for } j = 1,2,\dots,J' \\ \quad X = \langle x_1, x_2, \dots, x_n \rangle \end{cases} . \tag{1}$$

This formulation includes M objectives, and a solution for the problem is an n dimensional vector $X = \langle x_1, x_2, \dots, x_n \rangle$ such that satisfies all J inequality and J' equality constraints. Since there is a confliction among the objectives, there is no unique optimal solution for the problem.

Generally in multi-objective optimization problems, there is no priority knowledge between the objectives (if there is such priority we can combine the objectives in a unique objective function by weighted linear combination), so, three cases may happen for between two feasible solutions X and Y:

i. X *dominates* Y, which means X equals or better than Y in all objectives ($\forall i: f_i(X) \le f_i(Y)$) and X is strictly better than Y in some objectives ($\exists j: f_j(X) < f_j(Y)$),
ii. Y *dominates* X, and
iii. X and Y are *non-dominated*, that means none of them dominates to each other.

For a multi-objective optimization problem, the *Pareto optimal solutions* are the set of all non-dominated solutions of the whole search space that map to *Pareto optimal fronts* in the objective space (we use *Pareto solutions* and *Pareto fronts* for the sake of simplicity). Finding the Pareto fronts is the first goal in solving a multi-objective problem. However in most of the problems, the Pareto set is infinite and reporting all of them is impossible. On the other hand, decision maker needs just one Pareto solution. Thus, the second goal of multi-objective optimization emphasizes to diversity among the obtained Pareto solutions in the objective space [28]. Therefore, finding a small set (handful) of Pareto solutions as diverse as possible in the objective space is the main goal in solving multi-objective optimization problems.

Evolutionary algorithms and their improved variations [33, 34], and other heuristic approaches, such as Particle Swarm Optimization, Tabu Search, Simulated Annealing, Ant systems and hybrid heuristic approaches [28], have been extensively applied to solve multi-objective optimization problems. Indeed, evolutionary algorithms are popular general frameworks to solve such problems and find Pareto solutions. However, the important disadvantage of the evolutionary algorithms is that there is no guarantee in achieving Pareto solutions. Hence, it is better to avoid applying evolutionary algorithms and other heuristic approaches if there are deterministic methods to make such a guarantee.

3 Bi-objective Path Planning in Grid workspace

Let G be a grid path planning workspace and s be a starting cell for a robot. Each cell of G is completely *free* (the robot can move there) or occupied by an obstacle —called an *obstacle cell*. The goal is to preprocess the workspace such that for a given destination cell d in the query phase, all the Pareto solutions are computed efficiently. Two usual movement strategies in the grid are *four-points connectivity* (4pc) and *eight-points connectivity* (8pc) [35]. 4pc strategy allows the robot moves to one of the four adjacent cells *up, bottom, left,* and *right* in each step, however, 8pc allows four more adjacent cells *up-left, up-right, bottom-left,* and *bottom-right*. So, for two cells c_i and c_j, the movement $c_i \rightarrow c_j$ (or $c_j \rightarrow c_i$) is *valid* if they are adjacent with respect to 4pc or 8pc. In the rest of the paper we focus on 4pc strategy; however the concepts, algorithms and theorems are hold for 8pc as well.

A feasible and valid path P starting from cell s and ending at cell d can be defined by a sequence of single adjacent movements such as $P:=[s=c_0, c_1, c_2, ..., c_{l-1}, c_l=d]$, where all cells c_i for $i=0,1,...,l$ are obstacle free cells and $c_i \rightarrow c_{i+1}$ is valid respect to the movement strategy. In this case the length of P is l. To compute the clearance of P, we need to first compute the minimum distance between each cell c_i (for $i=0,1,...,l$) from its nearest obstacle cell and then find the minimum one. Let $MinObsDis(c)$ be the minimum distance between cell c and its nearest obstacles cell(s).

We consider two objectives minimizing the length of the path and maximizing the minimum distance between the path and obstacles. Consequently, a (feasible: obstacle-free) path P has two objective values: the length of P, denoted by $L(P)$, which is the number of the single movements, and the clearance of P, denoted by $C(P)$, which is the minimum distance (number of cells) between any cell of P and its nearest obstacle cell(s). Therefore, we can formulate the problem as the following bi-objective model:

$$\begin{cases} P:=[s=c_0, c_1, c_2, \dots c_{l-1}, c_l=d] \\ \\ \boldsymbol{Minimize} \quad L(P)=l \\ \boldsymbol{Maximize} \quad C(P)=\min_{0\le i\le l} MinObsDis(c_i) \\ subject\ to: \\ \quad \textbf{Collision free constraint}: c_i \text{ is an obstacle free cell}, \ 0\le i\le l \\ \quad \boldsymbol{Movement\ constraint}: c_i \rightarrow c_{i+1} \text{is a valid movement}, 0\le i\le l-1 \end{cases} \tag{2}$$

We denote this grid bi-objective path planning problem by *G2objPP*. Clearly, there is no unique optimal solution for G2objPP. In fact, there is a trade-off between $L(P)$ and $C(P)$ in which for finding a better $L(P)$ we need to sacrifice $C(P)$ and vice versa. A path P is a Pareto solution if there is no other solution P' such that $L(P') < L(P)$ and $C(P') \ge C(P)$, or $L(P') \le L(P)$ and $C(P') > C(P)$. Therefore, we are interested in finding all the Pareto solutions of G2objPP problem. Note that it is possible there are many Pareto paths with the same length and clearance. This means, all such paths in the solution space map to the one Pareto solution in the

objective space. In this situation finding just one path in the solution space is sufficient. Also, the destination cell d is not predefined as an input in the preprocessing phase; it is given in the query phase and the goal is to compute all the Pareto *s*-*d*-path efficiently. In the following we propose an efficient approach to solve this problem.

First, for each cell c in G we compute and assign $MinObsDis(c)$ value. To this end, we use an iterative procedure explained as follows. Suppose the initial *MinObsDis* value $+\infty$ for all free cells and *MinObsDis* value zero to all obstacle cells. Clearly, if c is a free neighbor cell of an obstacle its *MinObsDis* is 1. So, we assign 1 as *MinObsDis* value to all free cells which are neighbor of some obstacle cells. Then, in each iteration, if an unassigned cell c has some neighbors with minimum *MinObsDis* value k, we assign $MinObsDis(c) = k+1$. This procedure iterates until a proper value are assigned to all cells. By using a simple *queue-based* approach, we can compute $MinObsDis(c)$ for all cells of G in $O(|G|)$ time, where $|G|$ is the number of cells in G, e.g., if G is an $m \times n$ grid, $|G| = mn$. Figure 1 shows the result of this procedure for a 30×40 grid with respect to 4pc. Also, by using *MinObsDis*, the Voronoi diagram, which is the set of cells that has more than one nearest obstacle cell, can be computed easily. Note that, it is possible in the grid two neighbor cells on Voronoi diagram has the same distance from their nearest obstacles; the diagram contains some edges with thickness 2. Voronoi diagram is useful for finding the extreme Pareto solution that maximizes the clearance, because the path with maximum clearance passes through Voronoi cells.

Lemma 1. Number of Pareto paths (in the objective space) for G2objPP problem in a grid G with size $m \times n$ is at most $MaxC = min\left\{\left\lfloor\frac{m}{2}\right\rfloor, \left\lfloor\frac{n}{2}\right\rfloor\right\}$.

Proof. As mentioned above the maximum clearance path passes through some Voronoi cells. When there is no obstacle cell in G, maximum *MinObsDis* does not exceed $MaxC = min\left\{\left\lfloor\frac{m}{2}\right\rfloor, \left\lfloor\frac{n}{2}\right\rfloor\right\}$ which is related to central cell of G. Consequently, the number of Pareto optimal solutions in the objective space is at most $MaxC$. ■

Proposition 1. As a stronger result, we can conclude that the maximum number of Pareto solutions for G2objPP problem is at most *MaxVD*, which is the maximum $MinObsDis(c)$ among all Voronoi cells c.

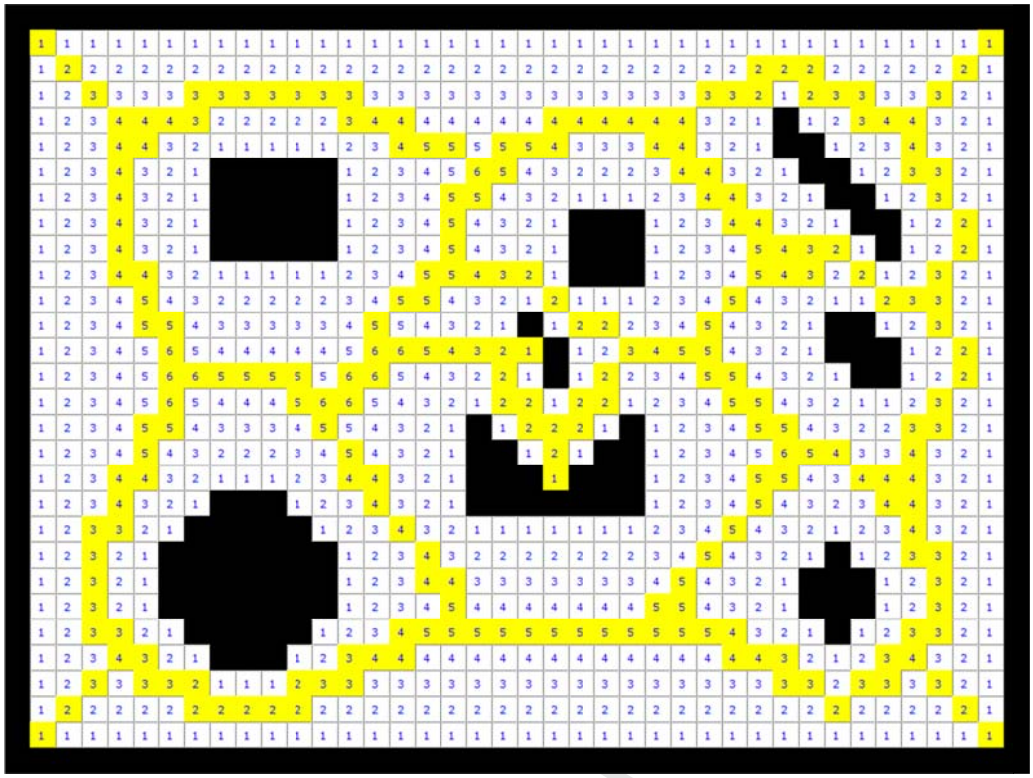


Figure 1. *MinObsDis* values for free cells with 4pc movement strategy. The yellow cells show the discrete Voronoi diagram.

After computing *MinObsDis* values for all free cells, we perform a *Breadth First Search* (BFS) by starting cell s to compute the minimum distance (length of the path) of each free cell from s. We denote such a minimum distance for a cell c by BFS(c). Simply this step is also takes $O(|G|)$ time and space. Figure 2 shows the result of BFS. So, by applying BFS's results for a given query destination cell d, we can simply compute a shortest s-d-path P. Note that, since we use all the cells c with $MinObsDis(c)\geq 1$, P has at least the minimum clearance $C(P)=1$. To compute such a path, it is sufficient to backtrack from d by using BFS's results. More precisely, if BFS(d)=l, there is at least one neighbor of d like d' such that BFS(d')=l-1. Therefor we can go back to d' from d that have the distance l-1 from s, and iterate this process to achieve the start cell s. Clearly, this process is output-sensitive and takes O(l) time to report P in the query phase. Note that, there is no guarantee that P is a Pareto solution in this step. In fact such a simple backtracking procedure is not sufficient to guarantee that the obtained path is a Pareto one, because it is possible there is a path P' with $L(P')=L(P)=l$ and simultaneously $C(P')>C(P)=1$. Hence, in the following we explain how to handle this issue when we aim to find all Pareto paths with clearance more than 1.

Observation 1. If P is a path with $C(P)=k$, there is a path P' with $C(P')=k$-1, however, it is possible P' is not a Pareto solution while P is.

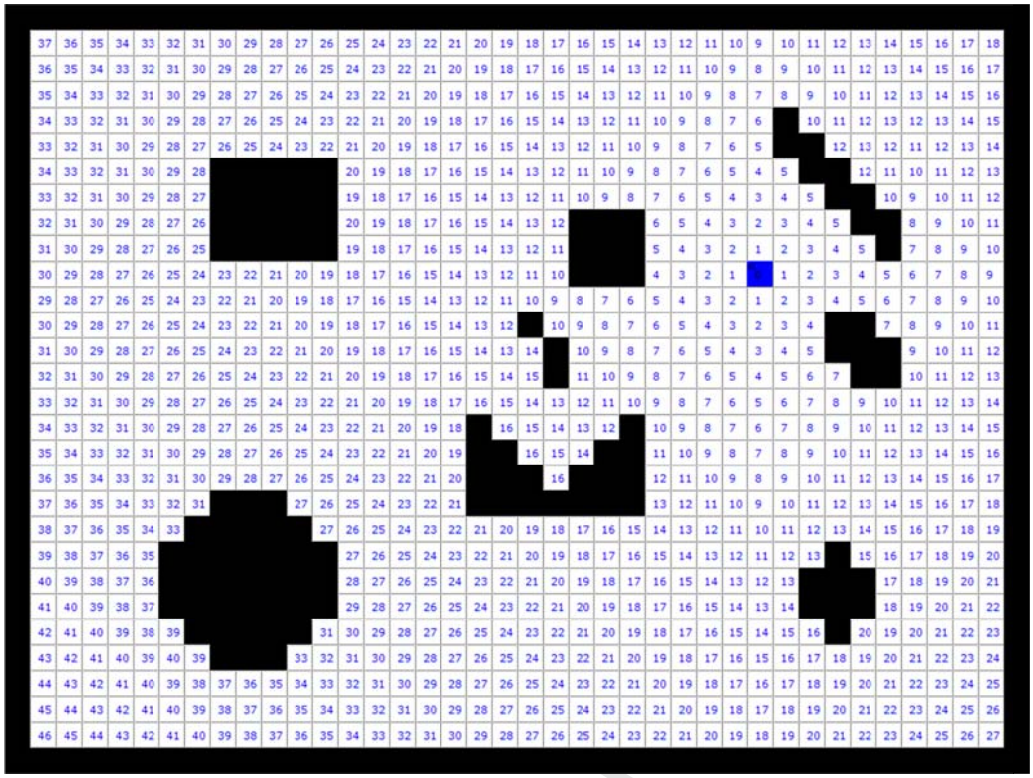


Figure 2. Result of Breadth First Search algorithm. The starting cell is shown by blue cell.

In the preprocessing step, we iterate the BFS procedure for all the cells c that $MinObsDis(c)\geq k$ for k=1,2,3,…, $MaxVD$. We denote the results of BFS for cells c that $MinObsDis(c)\geq k$ by $BFS_k(c)$. Indeed when we compute $BFS_k(c)$, we suppose all cells c' that $MinObsDis(c')<k$ are obstacle. So, it is possible for some probably large k there is no path. Thus, in the worst case the preprocessing phase runs in $O(|G|MaxVD)$ time and space. Let P_1, P_2, …P_K be the shortest path reported by backtracking procedure on BFS_k for k=1,2,..,K (for some $K \leq MaxVD$). See Figure 3 for k=1, 2, 3 and K=4 while $MaxVD$ *is* 6. Obviously $C(P_k)\leq C(P_{k+1})$ for k=1,2,...,K. Thus, we can easily compare $L(P_k)$ with $L(P_{k+1})$ and determine whether P_k is a Pareto solution. Figure 4 shows an example with two Pareto paths with clearance 3 and 4. Consequently, we can report all the Pareto solutions for a given query destination cell d in $O(l\ K)$ time, where l is the maximum path length of the obtained Pareto solutions and $K \leq MaxVD \leq MaxC$. So, we conclude this section with the following theorem and some remarks.

Theorem 1. For an $m\times n$ grid, all Pareto solutions in the objective space of G2objPP problem can be computed and reported in $O(l\ MaxVD)$ query time and $O(mn\ MaxVD)$ preprocessing time and space, where l is the maximum length among the Pareto solutions and $MaxVD = O(\min\{m,n\})$ is the maximum clearance among the Voronoi cells.

Since, all the mentioned running times are polynomial with respect to the size of the input, G2objPP problem belongs to complexity class $\mathcal{P}$.

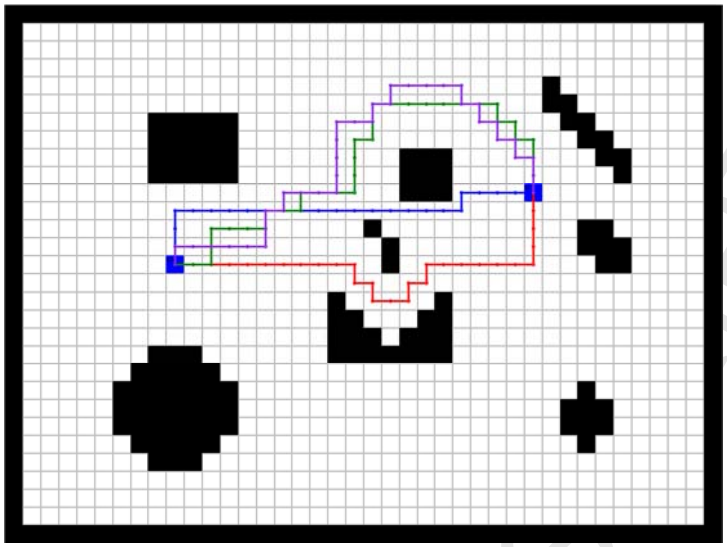


Figure 3. Four Pareto solutions with clearance 1, 2, 3 and 4 are shown by blue, red, green and violet. There is no more Pareto solution.

Remark 1. The concepts and results discussed in this section are hold for the 8pc strategy as well; however, some of them need a few elaborations. The *MinObsDis*, BFS and backtracking procedures are defined and used similarly for 8pc strategy. Voronoi diagram for 4pc is a Manhattan variation of the diagram in the continuous space while 8pc Voronoi diagram is different, however again the polynomial time property of the G2objPP problem is hold as well.

Remark 2. As the practical point of view, the workspace of a robot is a two or three dimensional space, however, some problems in this field need to construct the *configuration space* [1] in which usually its dimension is more than three. The approach presented in this section simply can be extended to three and higher dimensional space but the complexity increases too. For example a cell in three dimensional grid has 6 and 26 neighbors with 4ps and 8ps strategies, respectively.

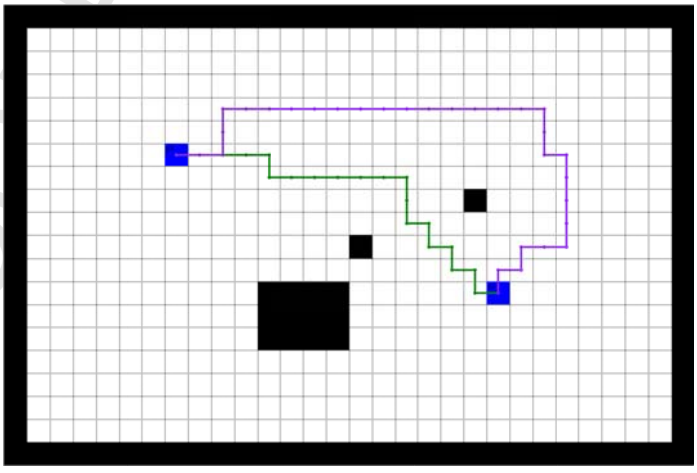


Figure 4. Two Pareto solutions with clearance 3 and 4. There is no other Pareto solution.

Remark 3. As mentioned before, there have been many studies that have proposed heuristic algorithms for single and multi-objective PP problems in (two and three dimensions) grid workspaces. For some examples see [19-23]. As proved in this section this problem is not an NP-hard problem (at least in the fixed dimensional grids). Hence there is no (anymore) justification for solving this problem by such approaches.

Remark 4. Bi-objective Breadth First Search Algorithm

In this remark, we improve the simple BFS algorithm and customize it for finding Pareto solutions of G2objPP problem efficiently. As aforementioned there is no guarantee for finding a Pareto path when we apply the backtracking procedure on the result of BFS. In fact, the computed path is a shortest *s*-*d*-path but there is no guarantee that it has the maximum possible clearance. To make such guarantee, it is sufficient we construct BFS tree by considering both the objectives length and clearance. Indeed the result of BFS is a directed tree with the root of the starting cell s and the nodes free cells. In this tree a node c is a parent for adjacent node c' if $L(c')= L(c)+1$. So, it is possible a node c' has more than one parent. In such a situation we consider the parent with the maximum clearance, and if they have same clearance, we arbitrary choose one of them as the parent. The following pseudocode shows such a Bi-objective BFS. Also, Figure 5 shows the result of this code for an example[1].

Bi-objective BFS

Input: Grid G and the start cell s.

Output: A shortest length path value for each free cell with regarding the maximum possible clearance.

Step 1: Compute *MinObsDis*(c) for all free cells c.

Step 2: Set BFS(c)=+∞ for all free cells and BFS(s)=0.

Step 3: Set PC(c)=0 for all free cells and BFS(s)= *MinObsDis*(s).

Step 4: Insert s to queue **Q** and repeat the following steps utile **Q** is not empty.

a) c=**Delete(Q)**. Let l=BFS(c) and k=PC(c).

b) For all neighbors of c such as c', if BFS(c)<BFS(c') then do the one of the following steps:

b1) If BFS(c')= +∞, BFS(c')= BFS(c)+1 and $PC(c') = \boldsymbol{min}\{ MinObsDis(c'), PC(c)\}$.

b2) If BFS(c')≠ +∞, BFS(c')= $\boldsymbol{min}\{$ BFS$(c'),$ BFS$(c)+1\}$ and

$PC(c') = \boldsymbol{max}\{ PC(c'), PC(c)\}$.

c) **Insert** c' to **Q**.

[1] The source code of Bi-objective BFS can be found at http://www.iasbs.ac.ir/~mdmonfared/projects.html.

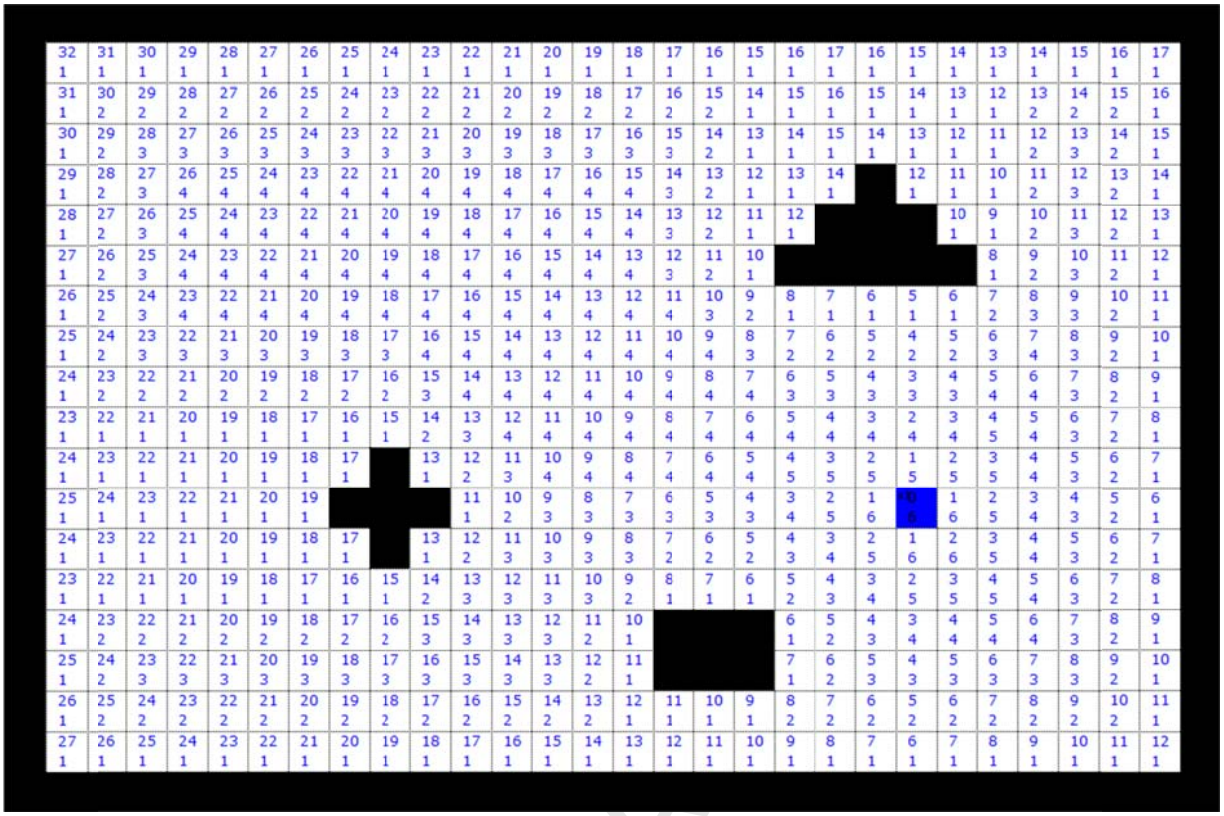


Figure 5. Result for Bi-objective Breadth First Search Algorithm. The blue cell is the starting cell *s*. There are two numbers in each cell *c*; the upper number shows the minimum length for the shortest path between *s* and *c*, BFS(*c*), and the bottom one shows the maximum achievable clearance for *c*, PC(*c*).

4. Continuous Space Path Planning with Manhattan Metric

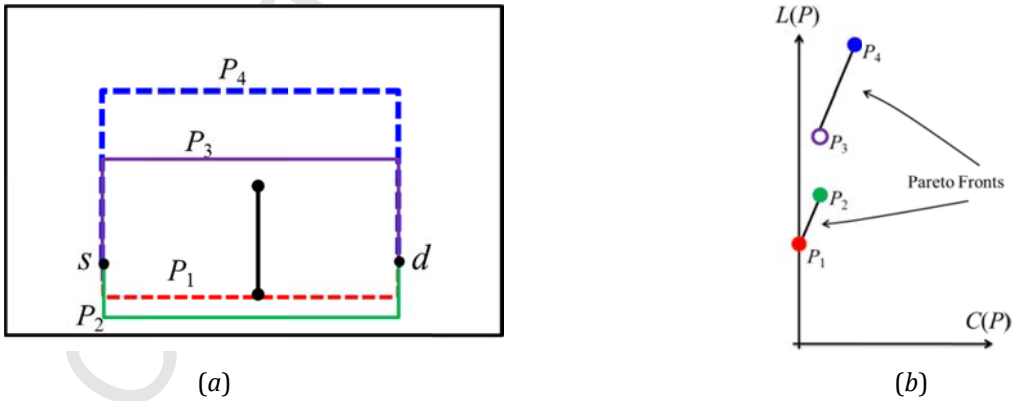
Similar to the problem of G2objPP we define the continuous version of the PP problem. Let W be a two-dimensional workspace containing polygonal obstacles o_1, o_2, ..., o_n and robot's source point s. The goal is a proper preprocessing of W such that for a given query destination point d, all Pareto s-d-paths are computed efficiently. A feasible path in W is a polygonal obstacle-free point-wise linear chain that can be represented by a set of its sequential break points. There are two objectives: minimizing the length of the path and maximizing the clearance which is the minimum distance between the path and obstacles. We denote this problem by C2objPP[2]. As aforementioned the clearance of the path is related to safeness of the robot and obstacles. Also, it can come from in uncertainty source in the real world [36]. Note that any metric such as Manhattan or the Euclidean can be used for the definition of the distance function. First, we apply the Manhattan metric and then we discuss about the Euclidean one.

[2] Note that discretization is a general approach to convert a continuous space to a discrete one. To this end, it is sufficient by gridding the workspace with a predefined size ε and applying approach presented in section 3, all the Pareto optimal solutions with the different clearance at most ε are computed. So, this is a good approximation approach but there is no guarantee in polynomial running time of it, because it is sensitive to ε.

However, discretization is a general approach for handling path planning problem in the continuous space, there is an essential difference between the discrete and continuous versions of the bi-objective PP problem, that is, the number of Pareto solutions is infinite in the continuous workspace. Figure 6(a) shows a simple instance of C2objPP under the Manhattan distance. The complexity of such an input is O(1), while there are many Pareto paths. So, it is impossible to report more than O(1) Pareto path using a polynomial algorithm. Therefore, as a theoretical and practical point of view finding extreme Pareto paths and all *topologically different paths* are more interested. By "topologically different paths", we mean the paths that lie on different Pareto fronts in the objective space. Figure 6(a) shows four special paths. P_1 and P_2, and P_3 and P_4 are topologically same and there are many infinite paths between them. In fact $[P_1, P_2]$ and $(P_3, P_4]$ are the set of Pareto intervals (generally the first one is closed interval and the other ones are half-open intervals) contains all the Pareto solutions. We call such intervals *Critical Pareto Optimal Intervals* and denote by CPOI. The projection of the CPOI in the objective space is exactly Pareto fronts and the critical paths are exactly the endpoints of such fronts (see Figure 6(b)). Note that these endpoint solutions are nice candidate Pareto solutions which are as diverse as possible. Therefore, finding all such intervals' endpoints can be more interesting instead of finding all Pareto solutions. Also, by having CPOI, we able to answer the following two main questions efficiently.

1. For a given query length l, what is the Pareto path P with L(P)= l under Manhattan metric?
2. For a given query clearance λ, what is the Pareto path P with C(P)= λ?

In the following we discuss on a limited version of PP problem and propose an efficient algorithm for finding all CPOIs.



(a) (b)

Figure 6. A simple bi-objective PP problem in the continuous workspace. (a) There is just one obstacle (the vertical segment). Two extreme Pareto paths are shown by dashed red (P_1) and blue (P_4) under Manhattan metric, however, there are many infinite Pareto solutions among them. Two special topologically equal with red and blue are shown by solid green (P_2) and violet (P_3). The clearance of the green and violet paths is equal, but the violet one is not a Pareto solution. All the Pareto solutions between P_1 and P_2, and between P_3 and P_4 are topologically eaual. (b) The objective space and the Pareto fronts of Figure 6(a).

4.1 Bi-objective Path planning amidst a set of vertical segments

In this subsection we consider Manhattan distance and study the problem of bi-objective PP in a workspace containing a set of n vertical segments. We discuss on the query version of the problem, that is, the destination point d is not as a given in the preprocessing phase.

Let Obs={s_1, s_2,..., s_n} be the ordered (left to right) set of the vertical obstacle segments. In the classic single-objective PP problem for a set of n vertical segments the shortest path (probably a path with clearance C(P)= $\lambda = 0$) can be found in O(n log n) time [17]. However, for C2objPP problem we need to compute all topologically different Pareto paths, that is, finding all the topologically different shortest paths when λ increases 0 to $+\infty$. Clearly, for $\lambda = 0$ the break points of the shortest path belong to the vertices of the obstacle. Now, to compute a Pareto solution with C(P)= λ, we can fat the obstacles. That is, performing the *Minkowski sum* of the obstacles with a square with length 2λ. See Figure 7. Let $Obs(\lambda)$={$s_1(\lambda)$, $s_2(\lambda)$,..., $s_n(\lambda)$} be the resulted fatted segments.

Observation 2. The break points of the shortest path for any given clearance λ belongs to $Obs(\lambda)$.

Observation 2 shows that by computing the shortest path for $Obs(\lambda)$ we can find the Pareto path P with $C(P) = \lambda$. So, the bi-objective path planning problem C2objPP can be seen as a classic shortest path problem when λ increases continuously. As aforementioned, this is impractical to implement, thus we focus on CPOI.

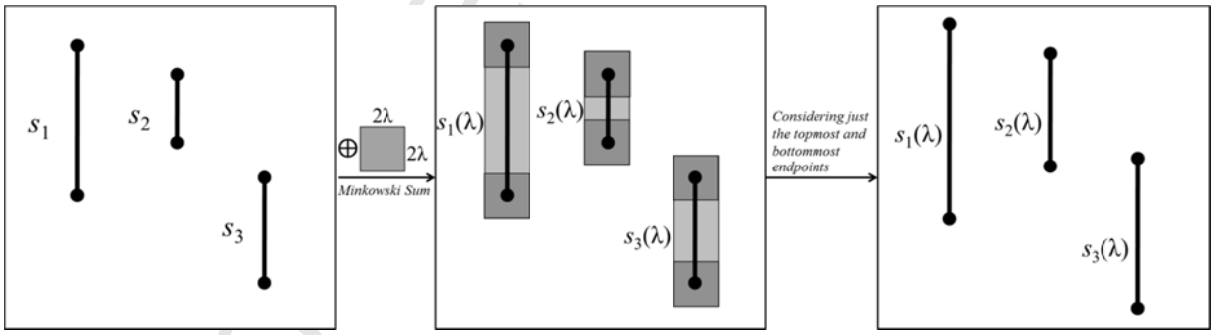


Figure 7. Fatting the obstacle by Minkowski sum with a square, and considering just the topmost and the bottommost endpoints of the fatted obstacles.

To compute CPOI, we must take into account the critical paths that are the endpoints of the Pareto fronts. While increasing λ does not change the topology of the shortest path, we stay on one component of the Pareto fronts. As increasing λ changes topologically some break points, an *event* does happen. For efficient handling the events occur while λ changes, first we propose an algorithm for finding the shortest path amidst a set of vertical segments. Then we handle the

events based on this algorithm. We construct a data structure that able to handle the events and construct the shortest path, or report the shortest path for any given query clearance λ.

For a set of vertical obstacle segments $Obs=\{s_1, s_2,..., s_n\}$, we denote the i-th segment by $s_i=(p_{2i-1}, p_{2i})$. Since the shortest path is an x-monotone path, for a given source point s on the left of s_1, we incrementally construct an efficient data structure tree T in $O(n \log n)$ time and $O(n)$ space to compute the shortest path.

We initialize T with the root s and two child nodes p_1 and p_2 (endpoints of the first segment), and two edges (pointers) from s to p_1 and p_2. Let W_1 and W_2 be the length of the shortest path (in Manhattan) from s to p_1 and p_2, respectively — called *initial weights* for p_1 and p_2. By such weights we can partition the right half-space of s_1 to two regions R_1 and R_2 such that a point p in the right half-space of s_1 belongs to R_1 if and only if $W_1+D_1(p_1,p)<W_2+D_1(p_2,p)$, where $D_1(.,.)$ is the Manhattan distance. Similarly, R_2 contains points closer to p_2. Other points correspond with $s=p_0$ and lies in its region, denoted by R_0. Let *bisector* B_1 be the set of all points like p in the right half-space of s_1 such that $W_1+D_1(p_1,p)=W_2+D_1(p_2,p)$. If the y-coordinate of s lies between the endpoints of s_1, B_1 is an orthogonal line to s_1. Figure 8(a) shows the explained concepts and notations. Note that the x-position of a segment like s_1 is not important for determining B_1, and for two segments s_1 and s_2 which s_1 lies on the left of s_2, we can construct T incrementally only by using their endpoints' y-coordinate (note that this property does not hold for the Euclidean distance metric).

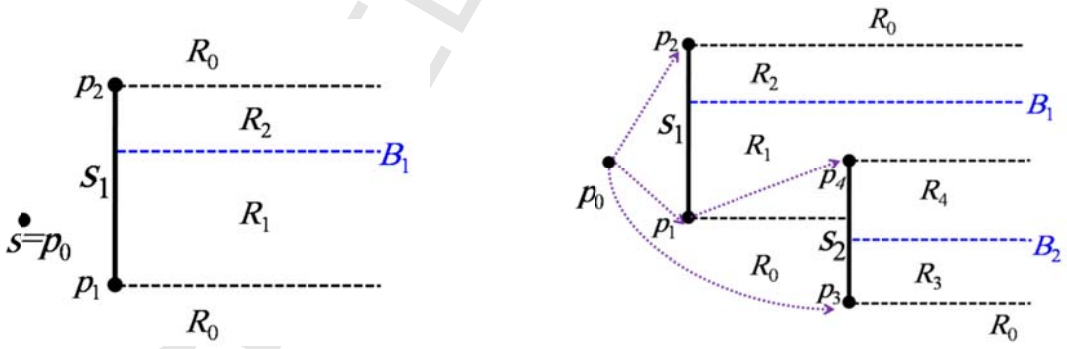


Figure 8. (a) Partitioning of the right half space of s_1 based on the shortest path length. (b) structure T and partitioned regions for two segments.

In fact, the mentioned partitioning is the Manhattan Voronoi region of p_1 and p_2 with respect to their shortest distance from s (the initial weights W_1 and W_2). So, for a point p in R_1, e.g., for endpoints of the next segments, the shortest path from s to p does not pass through p_2. This simple observation helps us to construct the shortest path efficiently. To see this, consider constructing T for the next level, $s_2= (p_3,p_4)$. If p_3 (and similarly p_4) lies in region R_k, for k=0,1 or

2, we simply insert it into T and make an edge from p_k to it. Then we compute bisector B_2 with respect to the initial weights W_3 and W_4, and partition the right half-space of s_2. Figure 8(b) shows T and the regions. Assume we construct T up to level i-1, and now we want to handle segment s_i with endpoints p_{2i-1} and p_{2i}. If p_{2i-1} (and p_{2i}) lies in region R_k, for some k=0,1,…,2i-2, we simply insert it to T and make an edge from p_k to it. Then we compute the initial weights W_{2i-1} and W_{2i} and bisector B_i for partitioning right half-space of s_i to R_{2i-1} and R_{2i}. The correctness of this process can be proved by the following facts.

Observation 3. The shortest path from s to the endpoints of segments is an x-monotone path.

Lemma 2. The shortest path between an endpoint p and any points located in the Voronoi region of p, completely lies in the Voronoi region of p.

Since we construct T incrementally and the Voronoi regions are semi-unbounded axis-aligned boxes, a point q lies exactly in the Voronoi region of one of endpoints of a segment s_k if and only if s_k is the first segment intersecting horizontal ray emanating from q to the left (we assume an enough large vertical segment s_0 passing through the source point s). Based on this observation we can find the parent of an endpoint inserted in each level. We first find the intersecting segment and then by a simple comparison of the y-coordinate and the initial weight of its endpoints the parent is determined.

Theorem 2. Manhattan shortest path from the source point s to all endpoints of a set of n vertical segments can be found in O(n log n) time and O(n) space.

Proof. First, we sort the segments with respect to their x-coordinate. In level i of the explained process, when we want to handle the segment s_i with endpoints p_{2i-1} and p_{2i}, we do following steps. We find the first intersecting segment with the horizontal ray emanating from p_{2i-1} (and p_{2i}) to the left. Such a segment can be found in O(log i) query time and O(n) preprocessing space by a dynamic ray shooting technique [37]. After finding the parent, we insert node p_{2i-1} and the corresponding edge to T. Therefore, we handle segment s_i in O(log i). So, the total complexity time is O(n log n). Also, clearly the constructed structure T has 2n+1 nodes and 2n edges (pointers). So the space complexity is O(n). ■

Since the constructed structure T is a tree, each node has exactly one parent, the shortest path for a given destination query point d can be found in O(log n +m) time, where m is the number of break points. First we find the parent of d by the ray shooting technique in O(log n) time, then by using a backward manner from d to s the path can be found in O(m) steps. Also, the path can be extracted for the workspace by a horizontal movement (to the left) and then a vertical movement (towards the parent) from each node to its parent.

Now we extend the mentioned concepts to the fatted segments for finding a Pareto path with clearance λ. Let $S(\lambda)=\{s_1(\lambda), s_2(\lambda),..., s_n(\lambda)\}$ be a set of n ordered fatted vertical obstacle segments. Because of the ordered property of $S(\lambda)$, the shortest path for any destination point d is an x-monotone chain. Since in constructing the structure T, only the y-coordinate of the obstacles' endpoints are used (the topmost and bottommost endpoints of the fatted segment), the shortest path for a clearance value λ, can be defined by these two critical points. These critical endpoints can be defined by two linear functions topmost $t_i(\lambda)$ and bottommost $b_i(\lambda)$ for the fatted segment $s_i(\lambda)$, for $i=1,2,...,n$. Similarly, initial weights for endpoints and bisectors are linear functions denoted by $W_i(\lambda)$ and $B_i(\lambda)$. While λ increases, three types of events may occur:

1. Two endpoints functions $t_i(\lambda)$ and $b_j(\lambda)$ reach a same y-coordinate, $t_i(\lambda)=b_j(\lambda)$.
2. An endpoint function $t_i(\lambda)$ or $b_i(\lambda)$ intersects a bisector $B_j(\lambda)$, $t_i(\lambda)= B_j(\lambda)$ or $b_i(\lambda)= B_j(\lambda)$.
3. Two segments are joined (there is no anymore clearance path between them).

We discuss about the first and second types of the events. The third one is rather straightforward such that two segments are joined and we assume them as a one segment with the proper topmost, bottommost and bisector functions. Since the obstacle segments are ordered from left to right, there are $O(n)$ such events when λ increases. We can compute all such critical λs in the beginning and handle each event (in the number of their children) in the worst case $O(n)$.

The first type of events occurs when two linear functions $t_i(\lambda)$ and $b_j(\lambda)$ intersect at a critical λ. Since they are linear, there are at most $O(n^2)$ intersections for the first type while clearance value λ increases. Let Λ be the sorted list of such critical λs.

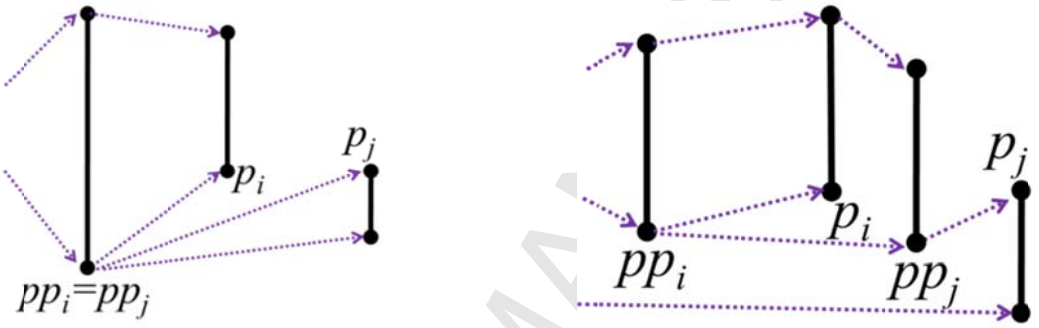
The second type of events occurs when a function $t_i(\lambda)$ or $b_i(\lambda)$ intersects with a bisector $B_j(\lambda)$. This event needs to handle if the endpoint $p_i(\lambda)$ lies on the Voronoi region of one of endpoints of $s_j(\lambda)$. Otherwise, the shortest path does not change. For a fixed λ there are at most $O(n)$ such critical λs indicating intersection point between each endpoint and its corresponding bisector. Since these values change dynamically, we compute such intersections for λ=0 and save them in a heap structure H, and update it during the algorithm. H can be constructed in $O(n)$ time and space, and supports insertion and deletion in $O(\log n)$ time. So, the next upcoming critical λ is determined between the minimum value in H and the next critical λ in Λ.

The structure T remains unchanged between two sequential critical λs. So, we construct T for λ=0, and update it when an event occurs. Changing T indicates a topologically change in the shortest path, so, we should report it (or the shortest path if the destination point is also given

in the preprocessing step) and the corresponding interval of λ. Now we explain how to handle events and update T when an event occurs.

1. Two endpoints p_i and p_j reach a same y-coordinate. One of p_i or p_j is the top endpoint and the other one is the bottom endpoint.

Assume p_i lies on the left of p_j. Let pp_i and pp_j be the parent of p_i and p_j, respectively. If $pp_i = pp_j$, it means both p_i and p_j lie on the Voronoi region of pp_i (or pp_j) before the event occurs, and after that p_j lies on the Voronoi region of p_i. Thus, we update T by removing edge (pp_i,p_j) and inserting edge (p_i,p_j). We, also update the initial weights for the endpoints in the subtree with root p_j. Otherwise, if $pp_i \neq pp_j$, T remains unchanged (see figure 9).



(a) (b)

Figure 9. The first type of events. (a) parent of p_i and p_j is the same, (b)parent of p_i and p_j is different.

2. An endpoint p_i intersects a bisector B_j.

B_j is the bisector of s_j with endpoints p_{2j-1} and p_{2j}. When p_i intersects B_j, the Voronoi region that p_i lies changes. Suppose it changes from R_{2j-1} to R_{2j}. So, p_{2j-1} is the parent of p_i when the event occurs (see figure 10). We update T by removing edge (p_{2j-1},p_i) and inserting edge (p_{2j},p_i). We also update the initial weights for the endpoints in the subtree with root p_i.

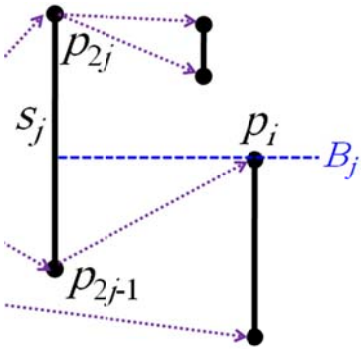


Figure 10. The second type of events for updating T, p_{2j-1} is parent of p_i.

After each iteration of the above process, if the initial weights of some endpoints are changed, we compute the new bisectors for them and update H. Finally, the next upcoming event is determined between the minimum in H and the next element in Λ. If two events occur simultaneously, we handle the left one at first. For example, if y-coordinate of pairs p_i and p_j, and $p_{i'}$ and $p_{j'}$ reach a same value coincidently, we first handle the left event, one of the parent that has the smaller x-coordinate. It is true because our algorithm update T incrementally from left to right. In both types of the events, the initial weights and bisectors can be updated in $O(n)$ time in the worst case. Based on the type of query, there are different approaches to report the shortest path.

Complexity Analysis. As aforementioned, because of the ordered property of $S(\lambda)$ the shortest path is x-monotone. In the proposed algorithm there exists exactly one parent for each endpoint in the structure T. So, by starting a backward manner from endpoint p_{2i}, at most i-1 break points lie on the shortest s-p_{2i}-path. If the destination point d is given and m is the number of break points on the shortest s-d-path, we can report the shortest path in $O(m)$ time and space. Also, its length is a linear function based on the clearance value between any sequential critical λs. Note that, in this case there is no need to report T completely; we only report the break points as the shortest path and weight of them. Therefore, all the shortest paths can be defined as a piecewise linear function with at most $O(n^2 + k)$ break points in $O(mn^2 + nk)$ time, where k is the number of insertion to and deletion from H which is $O(n^2)$ in the worst case, and m is at most n, e.g., for an orthogonal zigzag path. Considering the worst case of parameters k and m, we conclude our algorithm finds all CPOI, Pareto fronts' endpoints in $O(n^3)$ time.

If d is not given in the preprocessing step, after preprocessing and constructing T, we find the parent of d for λ=0 using the ray shooting technique in $O(\log n)$ time. By increasing λ it is possible the set of parents changes at most $O(n^2)$ times. So, we can construct the functions in $O(n^3)$ time and space. Finally, if both the destination point d and clearance are query, regarding to theorem 2 the shortest path can be computed in $O(n)$ time and space. Using the preprocessing, we can find the corresponding structure T among all $O(n^2)$ candidate structures by a binary search in $O(\log n)$ time (in the preprocessing step the structure T is reported in order while λ increases). Then we do the ray shooting technique for finding the parent of d. As the final result, by having CPOI the shortest path can be found in $O(\log n + m)$ time and $O(m)$ space for a given destination query, where m is number of breakpoints of the shortest path. Note that as it will be shown in Theorem 3, computing the Manhattan shortest path for a set of vertical segments belongs to $\Omega(n \log n)$. Further, it is easy to see we can construct a workspace contains n vertical segments such that while λ increases, the shortest path changes $\Omega(n)$ times.

Theorem 3. Finding the Manhattan shortest path from a given source point s to a destination point d among a set of n vertical segments belongs to $\Omega(n \log n)$ in the worst case.

Proof. The proof is by a reduction from the sorting problem. Let $x_1, x_2, \ldots, x_n$ be a set of n real positive numbers. In linear time, we convert these numbers to $2n$ vertical segments and locate the source point s on the left of them and the destination point d on the right of them. We show that in such a workspace the Manhattan shortest path passes through the endpoints of the segments in increasing order of $x_1, x_2, \ldots, x_n$. For i=1, 2, …, n; we define s_i a vertical segment with the endpoints $(x_i, x_i + \varepsilon)$ and $(x_i, +\infty)$ and s_i' a vertical segment with the endpoints $(x_i, x_i - \varepsilon)$ and $(x_i, -\infty)$ for some small ε. So there is a path with clearance ε between s_i and s_i'. Also, define the source point $s = (\min_i x_i - \varepsilon, 0)$ and destination point $d = (\max_i x_i + \varepsilon, 0)$. By such a configuration the Manhattan s-d-path first meets the left segment which corresponds with the minimum real number $\min_i x_i$, and then meets the other ones in order till the right one which is the maximum real number $\max_i x_i$. Therefore by having the break points of such an s-d-path we have the sorted order of $x_1, x_2, \ldots, x_n$. ■

4.2 Discussion

We can improve the sorting phase for the defined set Λ by using an approach presented for geometric duality [38]. In this technique a point (a,b) in the workspace can be uniquely transformed to the line y=ax-b in the dual space and vice versa. As a result of such a duality, for a set of n points in the plane, the sorted list of slope of lines passing any pair of points can be obtained in O(n^2) time. This result helps us to obtain the sorted set Λ in O(n^2) time. Also, we can use a simple array with length $2n$ to store the value of parents for each endpoint. This simple O(n) space structure supports inserting and deleting an edge in constant time in the algorithm because the parent of each endpoint is unique and the shortest path is monotone, so, it can be found by a backward manner starting at destination point and follow the parents up to reach the source point.

We suggested an efficient algorithm for finding Pareto solutions under the Manhattan metric. Indeed, we showed that when the obstacles are n vertical segments, there are at most O(n^2) topologically different Pareto solutions, more precisely, the size of CPOI is O(n^2). This is not clear if we apply the Euclidean metric. In the following we show that our algorithm's result is a proper approximation for the Euclidean metric.

Definition 1. Let Π be a bi-objective minimization problem with objectives f_1 and f_2. Solution X is an (α, β)-*approximation* solution for Π if there is no solution Y such that $f_1(X) \geq \alpha f_1(Y)$ and $f_2(X) > \beta f_2(Y)$, or that $f_1(X) > \alpha f_1(Y)$ and $f_2(X) \geq \beta f_2(Y)$.

Definition 1 is an extension of approximation solution for single objective optimization [39] to bi-objective optimization. So, an (α, β)-*approximation* solution X means that there is no Pareto solution such that dominates to X if we improve $f_1(X)$ and $f_2(X)$ by factor α and β, respectively.

Theorem 4. The CPOIs obtained by the algorithm explained in subsection 4.1 for the problem of bi-objective PP amidst vertical segments under Manhattan metric are $(\sqrt{2}, 1)$-*approximation* for the problem under the Euclidean metric.

Proof. Let P be a Pareto path in Manhattan metric with length l and clearance λ. Assume we fat the obstacles with an square with length 2λ, so, P which is a piece-wise linear chain containing vertical and horizontal segments, touches the obstacles in some points $p_1, p_2,\ldots,p_m$. Let P'=[s, p_1, $p_2,\ldots,p_m$, d] be the Euclidean path from s to d (not necessarily obstacle-free). See figure 11. By using triangle inequality, it is rather easy to see the length of P is at most $\sqrt{2}$ times of the length of P', $L(P)\leq\sqrt{2}\ L(P')$. Since P passes through $p_1, p_2,\ldots,p_m$, its clearance is at most λ. So, P is an $(\sqrt{2}, 1)$-*approximation* of P'. Thus, if P' is the shortest Euclidean path with clearance λ, the proof is complete, otherwise, there is an Euclidean path Q' with clearance λ and we can approximate it using a Manhattan path Q such that $L(Q)\leq\sqrt{2}\ L(Q')$, and since P is a shortest Manhattan path, we have $L(P)\leq L(Q)$. Consequently, $L(P)\leq\sqrt{2}\ L(Q')$, and the proof is complete. ■

It is notable that Theorem 4 does hold in any PP workspace with arbitrarily obstacles. However for vertical obstacles, as we obtain a polynomial algorithm for C2objPP problem under the Manhattan metric, the solution is an $(\sqrt{2}, 1)$-approximation for the Euclidean metric as well.

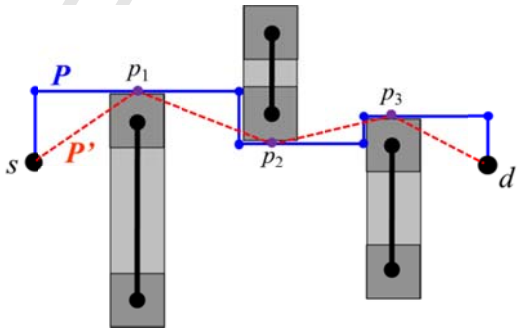


Figure 11. $(\sqrt{2}, 1)$-*approximation* of the Euclidean path with Manhattan path.

5 Conclusion and Future work

The main focus of this paper is on the complexity of bi-objective PP problem with minimizing the length and maximizing the clearance of the path. This problem has been extensively studied in the literature using heuristic approaches, particularly by evolutionary algorithms. We proposed an efficient polynomial algorithm for finding all Pareto optimal fronts of the problem

in the grid workspace. Further, we studied a limited version of the problem in the continuous space where the obstacles are vertical segments. Since the number of Pareto optimal solutions in this problem is inherently infinite, we focused on the topologically different Pareto optimal paths, specially, we concentrated on the endpoint of Pareto optimal fronts in the objective space. We considered this problem under Manhattan metric and showed that the number of such Pareto optimal solutions is polynomial, and proposed an efficient algorithm for computing them. Finally, we discussed approximating the Euclidean metric of the problem.

The tight complexity analysis of the bi-objective path planning problem and proposing efficient algorithm for finding its Pareto optimal solutions remain as open problems. Also, the approximation approach presented in subsection 4.2 can be extended to other problems that complexity of them depends on which distance metric is chosen. Generally, there are many multi-objective optimization problems that are NP-hard under the Euclidean distance while they may belong to the complexity class $\mathcal{P}$ under Manhattan distance. So, a general approach is first, finding the Pareto optimal solutions of such problems under the Manhattan distance, and then approximate the Pareto optimal solutions of the Euclidean distance.

Acknowledgment

This research was in part supported by a grant from IPM under a project named "*Approximating Multi-objective Optimization Problems*".

Reference

[1] S. M. Lavalle, Planning Algorithms, Cambridge University press, 2006.

[2] H. Choset, Principles of robot motion: Theory, Algorithms, and Implementation, MIT press, 2005.

[3] M. Arndt, S. Wille, L. De Souza, V. F. Rey, N. Wehn, K. Berns, Performance evaluation of ambient services by combining robotic frameworks and a smart environment platform, Robotics and Autonomous Systems, 61(11), pp. 1173-1185, 2013.

[4] J. O'Rourke, Computational Geometry in C (2nd ed.), Cambridge University Press, New York, 1998.

[5] R. Raja, A. Dutta, K. S. Venkatesh, New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover. Robotics and Autonomous Systems, 72, pp. 295-306, 2015.

[6] W. Li, G. Y. Wang, Application of improved PSO in mobile robotic path planning, Intelligent computing and integrated Systems (ICISS), pp. 45-48, 2010.

[7] N. Buniyamin, N. Sariff, W.A.J. Wan Ngah, Z. Mohamad, Robot global path planning overview and a variation of ant colony system algorithm, International journal of mathematics and computers in simulation, 1(5), pp. 151-159, 2011.

[8] U. A. Syed, F. Kunwar, M. Iqbal, Guided Autowave Pulse Coupled Neural Network (GAPCNN) based real time path planning and an obstacle avoidance scheme for mobile robots, Robotics and Autonomous Aystems, 62(4), pp. 474-486, 2014.

[9] R. E. Precup, R. C. David, E. M. Petriu, S. Preitl, M. B. Rădac, Novel adaptive charged system search algorithm for optimal tuning of fuzzy controllers. Expert Systems with Applications, 41(4), pp.1168-1175, 2014.

[10] R. E. Haber, J. R. Alique, Fuzzy logic-based torque control system for milling process optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 37(5), pp. 941-950, 2007.

[11] M. De Berg, O. Cheong, M. van Kreveld, M. Overmans, In Computational Geometry: Algorithms and Applications, 3rd. rev. ed., Springer-Verlag, Berlin, 2008.

[12] E. W. Dijkstra, A note on two problems in connexion with graphs. Numerische Mathematik, 1, pp. 269–271, 1959.

[13] S. K. Ghosh, D. M. Mount, An output sensitive algorithm for computing visibility graphs, Proc. 28th symp. On Foundations of computer science. Los Angeles, pp. 11-19, 1987.

[14] M. H. Overmars, E. Welzl, New Methods for Computing Visibility Graphs, Proceedings of the fourth annual symposium on Computational geometry, 1988.

[15] E. Welzl, constructing the visibility graph for n line segments in $O(n^2)$ time, Information processing letters, 20, pp. 167-171, 1985.

[16] J. Hershberger, S. Suri, An optimal algorithm for Euclidean shortest paths in the plane. SIAM Journal on Computing, 28(6), pp. 2215–2256, 1999.

[17] J. F. Canny, J. H. Reif, New lower bound techniques for robot motion planning problems. In Proc. 28th IEEE Sympos. Found. Computer Science, pp. 49–60, 1987.

[18] H. Zarrabi-Zadeh, Path Planning above a Polyhedral Terrain, In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida, United States, pp. 873-876, 2006.

[19] A. Elshamli, H. A. Abdullah, S. Areibi, Genetic Algorithm for Dynamic Path Planning, CCECE 2004 – CCGEI 2004, Niagara Falls, May 2004 IEEE.

[20] O. Castillo, L. Trujillo, P. Melin, Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots, Soft Computing, 11, pp. 269-279, 2007.

[21] F. Ahmed, K. Deb, Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms, Soft Computing, 17(7), pp. 1283-1299, 2013.

[22] M. Davoodi, F. Panahi, A. Mohades, S. N. Hashemi, Multi-objective path planning in discrete space. Applied Soft Computing, 13(1), pp. 709-720, 2013.

[23] M. Davoodi, F. Panahi, A. Mohades, S. N. Hashemi, Clear and smooth path planning. Applied Soft Computing, 32, pp. 568-579, 2015.

[24] R. Wein, The Integration of Exact Arrangements with Effective motion planning, PhD thesis, Tel-Aviv University, 2007.

[25] R. Geraerts, Planning short paths with clearance using explicit corridors. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 1997-2004, 2010.

[26] R. Geraerts, M. H. Overmars, Creating high-quality paths for motion planning, The International Journal of Robotics Research 26.8, pp. 845-863, 2007.

[27] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, Wiley, 2001.

[28] C. C. A. Coello, D. A. Van Veldhuizen, G. B. Lamont, Evolutionary algorithms for solving multi-objective problems. Vol. 242. New York: Kluwer Academic, 2002.

[29] V. Roostapour, I. Kiarazm, M. Davoodi, Deterministic Algorithm for 1-Median 1-Center Two-Objective Optimization Problem. In Topics in Theoretical Computer Science, Springer International Publishing, pp. 164-178, 2016.

[30] S. Vassilvitskii, M. Yannakakis. Efficiently computing succinct trade-off curves. Theoretical Computer Science 348, pp. 334–356, 2005.

[31] C.H. Papadimitriou, M. Yannakakis. On the Approximability of Trade-offs and Optimal Access of Web Sources. In Proc. FOCS, pp. 86–92, 2000.

[32] A. Warburton. Approximation of Pareto Optima in Multiple-Objective Shortest Path Problems. Operations Research, 35, pp. 70–79, 1987.

[33] A. Cardon, T. Galinho, J. P. Vacher, Genetic algorithms using multi-objectives in a multi-agent system. Robotics and Autonomous Systems, 33(2), pp. 179-190, 2000.

[34] M. Z. Ali, N. H. Awad, R. M. Duwairi, Multi-Objective Differential Evolution Algorithm with a New Improved Mutation Strategy. International Journal of Artificial Intelligence™, 14(2), pp. 23-41, 2016.

[35] M. Davoodi, M. Abedin, B. Banyassady, P. Khanteimouri, A. Mohades, An optimal algorithm for two robots path planning problem on the grid, Robotics and Autonomous Systems, 61(12), pp. 1406-1414, 2013.

[36] M. Davoodi, A. Mohades, F. Sheikhi, P. Khanteimouri, Data imprecision under λ-geometry model, Information Sciences, pp. 126-144, 2015.

[37] Y. Giora, H. Kaplan, Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. In Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms, pp. 19-28, 2007.

[38] B. Chazelle, L. Guibas, D. T. Lee, The power of geometric duality, in: Prec. 24th Ann. IEEE Symp. On Foundations of Computer Science, pp. 217-225, 1983.

[39] V. V. Vazirani, Approximation algorithms. Springer Science & Business Media, 2013.

Mansoor Davoodi Monfared received his B.S. degree in computer science from Vali-Asr University, Rafsanjan, Iran, in 2005, and M.S. Ph.D degrees in Amirkabir (Tehran Polytechnic University) University in 2007 and 2012 respectively. He is currently Assistant Professor in the Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan, Iran. His main research area includes: computational geometry, imprecise data modeling and multi-objective optimization.

Mansoor Davoodi Monfared

Highlights:

- Introducing a bi-objective length and clearance path planning (bi-PP) model
- Proposing an algorithm for solving the bi-PP in grid and finding Pareto fronts
- Proposing an algorithm for solving the bi-PP in continuous Manhattan space
- Approximating Pareto solutions of the bi-PP under the Euclidean distance