ARTICLE IN PRESS

Robotics and Autonomous Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

An optimal algorithm for two robots path planning problem on the grid

Mansoor Davoodi [a,*], Marjan Abedin [b], Bahareh Banyassady [b], Payam Khanteimouri [b], Ali Mohades [b]

[a] Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan, Iran

[b] Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, Iran

HIGHLIGHTS

- Introducing a new concept, *time collision cell*, for coordinating multi-robots.
- Introducing a new concept, *parking cell*, for coordinating multi-robots.
- Proposing an optimal algorithm for solving min–max path planning for two robots.

ARTICLE INFO

Article history:
Received 1 March 2013
Received in revised form
14 July 2013
Accepted 26 July 2013
Available online xxxx

Keywords:
Multi-robot path planning
Min–Max problem
Grid
Optimal algorithm

ABSTRACT

This paper is a study on the problem of path planning for two robots on a grid. We consider the objective of minimizing the maximum path length which corresponds to minimizing the arrival time of the last robot at its goal position. We propose an optimal algorithm that solves the problem in linear time with respect to the size of the grid. We show that the algorithm is complete; meaning that it is sure to find an optimal solution or report if any does not exist.

1. Introduction

Motion planning for a single mobile robot has been extensively studied over the last decades and several approaches have been developed to deal with it for a robot in an environment contained a set of stationary obstacles. Moreover, there are other studies for planning a path for a robot in an environment with moving obstacles which are intractable even in simple cases [1–3]. Based on this fact, motion planning for multiple robots is inherently hard. On the other hand, there are many practical applications of robots such as assembly planning and automated manufacturing which require the use of two or more robots in a common workspace. Also cooperation between robots provides many benefits, and this is why multi-robot motion planning is so appealing.

The multi-robot motion planning problem is generally defined as follows: given a set of m robots in a workspace, each one with start and goal positions; determine the path that each robot should follow to reach its goal while avoiding collisions with obstacles and other robots. Due to the objectives of the robots, there are different assumptions for the problem. For example all the robots may have the same goal from different starts, or each robot is destined to reach its individual goal. The main issue in mobile robots motion planning is to coordinate the motions of the multiple robots interacting in the same workspace to avoid potential collisions.

According to the statement and assumptions of the multi-robot motion planning problem, there are several approaches to deal with. In a general view, the methods are divided into two taxonomies, centralized (called coupled) and decoupled. In the centralized approach, all robots together are considered as a single robot and a path is computed in a composite space [4]. The advantage of the approach is its completeness and the main issue with it, is the time complexity of the path planning method which grows exponentially with respect to the number of robots [5,6]. The algorithm in [5] decomposed the free space into some cells and searches a path in the corresponding graph. The same method was used in [6] for multiple disk robots moving among stationary polygonal obstacles. Classical problems for exact motion planning

* Corresponding author. Tel.: +98 2414155060; fax: +98 2414155071.
E-mail addresses: mdmonfared@iasbs.ac.ir, md_monfared@yahoo.com (M. Davoodi), m.abedin@aut.ac.ir (M. Abedin), bahareh.banyasady@aut.ac.ir (B. Banyassady), p.khanteimouri@aut.ac.ir (P. Khanteimouri), mohades@aut.ac.ir (A. Mohades).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.07.012

usually use the centralized method; randomized and other similar techniques have been applied for faster results [7,8]. Some results by Svestka et al. [9] are probabilistically complete and the method is efficient for a small number of robots. Ryan [10] introduced the concept of subgraph decomposition as a means of reducing the search space in multi-robot planning problems. He showed how partitioning a roadmap into subgraphs of known structure allows first planning at a level of abstraction and then resolving these plans into discrete paths without the need for further searches. Also, he described a kind of subgraph called a *hall* [11], which can be used for planning which occurs much more commonly in real problems. In the paper a naive centralized planner proceeded as follows. First, initialize every robot at its starting position, then select a robot and move it to a neighboring vertex, checking first that no other robot is currently occupying that vertex. Continue in this fashion selecting and moving one of the robots at each step until all robots are at their goals. Each choice presents multiple possibilities and all alternatives have to be searched in some systematic fashion, usually by breadth first search (BFS) or A^* algorithm.

Lack of efficiency in the centralized approach is compensated in the decoupled approach where completeness is sacrificed in favor of the complexity. In the decoupled approach, planning is done in two phases. First for each robot, a path is computed which is collision-free with respect to the obstacles and other robots. Then, time collisions between the robots are resolved in the second phase, see [12,13]. In [12] the trajectory planning problem was transformed to a velocity problem for the robots in the fixed paths designed in the first phase. A similar idea was followed in [13], but the robots are prioritized and the problem is divided into a series of planning problems. There is one moving robot in each problem while other robots with higher priority are considered as moving obstacles. The order of prioritizing is an important issue that influences the ultimate performance of the method; studied in [14]. The coordination diagram introduced in [15] deals with motion planning for two robots and the algorithm in [16] improves the idea to deal with the problem for more than 100 robots in realistic situations. The coordination graph in [17] and incremental planning in [18] are also methods used to avoid collisions in the velocity tuning phase. Furthermore, Alami et al. [19] used the benefits of both centralized and decoupled methods in a hybrid method that is more reliable than solely decoupled and faster than centralized. They constructed a directed acyclic graph which stores a snapshot of the current configuration and the already planned process.

Standley [20] presented an optimal and complete algorithm for the problem of minimizing the sum of all time steps that each agent spends away from its goal. He used *independence detection* to decompose instances into independent subproblems and only used a more costly but admissible search algorithm, *operator decomposition*, when no such independence can be found. Although this combination is both complete and optimal its running time is prohibitively expensive for many real-time applications. Another try [21] followed online static route computations combined with load balancing techniques. The paper studied whether the static approaches can meet the performance of the dynamic approaches while being collision and deadlock-free.

Among the problems studied in the multi-robot motion planning area, it is a challenging issue to minimize the last robot's arrival time (makespan). We call the problem min–max multi-robot path planning and it is generally stated as follows. Given m start–goal positions for m mobile-robots in a continuous/discrete workspace containing stationary obstacles; minimize the arrival time of the last robot while all the other robots are in their goals. Related works for optimal motion planning are done in [22,23]. Shiller et al. [22] presented a time-optimal velocity planning algorithm for a robot moving along a fixed path with acceleration bounds to the axis. A method to plan collision-free time-optimal motions for two robots with limited actuator torques and velocities in a continuous workspace was proposed in [23], and the minimal-time criterion method under the B-Spline assumption of the Cartesian path was used in [24].

In addition to the applications of the problem in multi-robot motion planning, min–max is an interesting topic in network packet routing problems where some packets at possibly different start positions are given; minimize the arrival time when the last packet arrives to its destination is the issue [25]. The packets should avoid colliding with each other during the movements similar to the movements of robots in the multi-robot planning problem. The most similar work in the graph context to multi-robot planning is the problem of finding disjoint paths in the graph. Li et al. [26] studied the problem of finding two disjoint paths between two given nodes such that the length of the longer path is minimized. They showed that both the problems of vertex-disjoint and edge-disjoint paths are strongly NP-complete.

As mentioned above, the problem of path planning for multiple robots is NP-hard, and it is exponential in the number of robots. Therefore studying the problem for a small number of robots and proposing efficient algorithms is interesting. In this paper, the min–max problem for two robots with equal constant velocities is studied in a grid workspace. However, a straightforward approach for solving the problem would construct the composite configuration space, resulting in an $O(n^2)$ time algorithm, where n is the number of free cells in the grid. We propose a complete algorithm that solves the problem in $O(n)$ time. The algorithm has two phases: in the first phase a minimum-length obstacle-avoiding path is computed for each robot, and in the second phase the (time-) collisions between the robots are resolved. This phase, which handles possible deadlocks, is the key step of our approach. We define two types of collisions and handle them efficiently based on a new concept which is a Voronoi diagram in terms of time and distance.

This paper is organized as follows. In section 2, the problem statement and the preliminaries needed throughout the paper are studied. The algorithm for the problem of min–max path planning for two robots is proposed in section 3. We follow in section 4 with conclusions and suggestions for future works.

2. Problem statement

As we described in the previous section we are going to deal with the problem of path planning for two robots in a grid workspace. So, we are given a rectangular workspace with an $m_1 \times m_2$ grid on it. Also, some cells are marked as obstacles which the robots should not collide with. Let n denote the number of free cells over the whole cells of the grid, so $n \leq m_1 * m_2$. Moreover, a pair of start and goal positions for each robot i, (s_i, g_i), is determined such that $s_i \neq s_j$ and $g_i \neq g_j$ for all $i \neq j$.

There are five possibilities for a robot's movement in one time step: *four-points connectivity* in which each cell has four neighbors, left, right, up and bottom to move or to be stationary in its current cell. A path for the first robot from its start position to its goal is a sequence $P = \{p_0 = s_1, p_1, p_2, \ldots, p_l = g_1\}$ such that each p_i is a free cell and a successor of a valid movement from p_{i-1} as described. Moreover the length of a path P is the number of steps, l, rather than the number of cells contained in P. Note that it is possible that a robot stays in one cell for one or more steps, which means $p_i = p_{i+1} = \cdots = p_{i+k}$ for some i and k. A particular case of such a situation happens when one of the robots has reached its goal and the other ones are still moving. It means that the robot stays on its goal cell.

In addition to avoiding collisions with obstacles, the robots are not allowed to be in collision with each other. As the path P for the first robot, suppose $Q = \{q_0 = s_2, q_1, q_2, \ldots, q_{l'} = g_2\}$ is a path

Please cite this article in press as: M. Davoodi, et al., An optimal algorithm for two robots path planning problem on the grid, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.07.012

for the second robot. We define two kinds of collisions for robots: both robots cannot move to the same cell simultaneously, meaning $p_i \neq q_i$ for all i, and the robots cannot swap their positions in one time step. More precisely, a pair of paths P and Q in which $p_i = q_{i+1}$ and $p_{i+1} = q_i$ is invalid. The problem of *Min–Max two robots path planning* (MM2rPP) is to minimize the $\max(l, l')$.

A naive $O(n^2)$ algorithm for solving the MM2rPP problem is constructing the composite configuration space of all possible pairs of cells (p, q). So, the problem can be reduced to a graph search problem on the configuration space. In the next section we propose an optimal $O(n)$ time algorithm that solves the problem of MM2rPP.

3. Optimal algorithm for Min–Max two robots path planning problem

For simplicity we show the workspace grid by a graph G with nodes corresponding to free cells of the grid and there is an edge between two nodes if and only if the corresponding free cells are neighbors with respect to the four-points connectivity constraint. Let n be the number of nodes in G. Since the connectivity of G can be determined in $O(n)$ time, without loss of generality we assume G is one connected component. Before explaining the algorithm, we investigate the existence of a feasible solution to the problem.

Definition. A Parking node in G is a node which has degree greater than two.

A parking node (shortly a parking) in G, is a node of degree three or four. So, by using a parking, two robots can exchange their positions without any deadlock. Thus, a sufficient condition for existence of a feasible solution to the MM2rPP problem is that G contains at least one parking node. Otherwise, if G does not contain any parking, all the nodes have degree one or two. According to the assumption that G is one connected component, G is either a *simple cycle* (all the degrees are two) or a *simple chain* (two nodes with degree one and $n-2$ nodes with degree two). Since there are (at most) two clockwise and counter clockwise paths for each robot on the simple cycle, there is always a feasible solution to the problem in this case. However, it is possible that there is no feasible solution in the simple chain when s_1–g_1-path and s_2–g_2-path are in collision (s–g-path is one of the possible shortest paths from s to g). Therefore, not only the existence of a solution can be checked in $O(n)$ time for both the simple cycle and simple chain cases, but also the MM2rPP problem can be solved in $O(n)$ time using BFS by checking the constant number of feasible solutions.

A challenging problem is an upper bound on the length of a solution, which is the maximum number of time steps required for the last robot to arrive at its goal cell. The length of a feasible solution to the MM2rPP problem is at most $n-1$ if G is a simple chain or simple cycle. In Theorem 1 we show that in the general case where G contains some parking nodes, the length of a solution is at most $2n-5$ and also there is an example with this length of solution.

Theorem 1. *If G contains a solution to the problem of MM2rPP, its length is at most $2n-5$. Also $2n-5$ is the tight upper bound.*

Proof. When G has no parking, the upper bound for the length of a feasible solution is $n-1$. So, we consider G has some parking nodes. In the worst case, the s_1–g_1-path and s_2–g_2-path are in collision and the robots have to use a parking to exchange their positions to avoid the collision. Fig. 1 shows such an example. Since the distance of the furthest parking from s_1 or s_2 is $n-3$ steps, and robots can exchange their paths in one time step, and follow their goals in at most $n-3$ steps, so $(n-3)+1+(n-3) = 2n-5$ steps are sufficient in the worst case. Also, this bound is tight. Fig. 1 illustrates an example that needs exactly $2n-5$ steps in the best case. ■

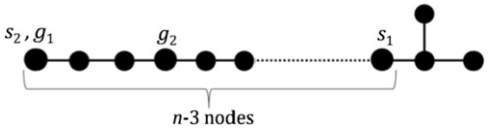

Fig. 1. An example of the MM2rPP problem that needs $2n-5$ steps.

The algorithm for solving the MM2rPP problem has two general phases. In the first phase, we find two s_1–g_1-path and s_2–g_2-path, denoted by P_1 and P_2, by using BFS. If P_1 and P_2 are not in collision, the problem is solved; otherwise, we handle the collision in the second phase based on the type of the collision. Let L_1 and L_2 be the length of P_1 and P_2. It is possible to find the collision cell in $O(L_1+L_2)$ time. So, if there exists a collision between P_1 and P_2, it can be found in $O(n)$ time based on Theorem 1. For handling the collision, we introduce two kinds of collisions: *head-to-head* collision and *next-to-next* collision. There are three types of head-to-head collisions which are defined as follows.

Type 1. One of the robots lies in cell c' at time t_0-1, in cell c at time t_0 and in cell c'' at t_0+1; and in reverse, the other robot lies in cell c'' at time t_0-1, in cell c at time t_0 and in cell c' at t_0+1. In this type of head-to-head collision, the cell c is called a *collision cell* for both P_1 and P_2.

Type 2. One of the robots lies in cell c' at time t_0-1, in cell c at time t_0 and in cell c'' at t_0+1; and in reverse, the other robot lies in cell c'' at time t_0-1, in cell c at time t_0 and in cell c''' at t_0+1, where $c''' \neq c'$. In this type of head-to-head collision, the cell c is called a *collision cell* for both P_1 and P_2.

Type 3. One of the robots that walks on P_1 lies in cell c at time t_0 and in cell c' at time t_0+1; the other one that walks on P_2 lies in cell c' at time t_0 and in cell c at time t_0+1. In this type, c is the collision cell for P_2 and c' is the collision cell for P_1.

We call all the remaining collisions next-to-next collisions. Fig. 2 shows these types. First, we explain how to handle a head-to-head collision.

Head-to-head collision

As aforementioned, there are three types of head-to-head collisions. In the following, we consider the first type of head-to-head collision. All the lemmas and procedures hold for the other types too.

The following lemma shows that if P_1 and P_2 have a head-to-head collision, there are no other collisions between them.

Lemma 1. *Let P_1 and P_2 be s_1–g_1-path and s_2–g_2-path which have a head-to-head collision in cell c at time t_0. The cell c is the only collision between P_1 and P_2.*

Proof. By contradiction, assume P_1 and P_2 have at least one other collision, e.g. in cell sc at time $t_0+1+\Delta t$ for a positive value of Δt (see Fig. 3). Regarding this fact that P_1 and P_2 are shortest paths and any subpath of a shortest path is the shortest one, it concludes that subpaths s_1–sc-path of P_1 and s_2–sc-path of P_2 are also the shortest subpaths to reach sc from s_1 and s_2 with length $t_0+1+\Delta t$. However, there is a shorter path for each of the robots with length $t_0-1+\Delta t$ that does not pass the collision cell c (the solid subpath in Fig. 3), which contradicts the shortest length of P_1 and P_2. A similar proof is used for the other types of head-to-head collisions. Also, the proof is similar for the negative values of Δt. ■

Suppose P_1 and P_2 have a head-to-head collision in cell c. Based on the lemma, this collision is the only (time) collision between them. So, if it is handled by using one of the nearest parking nodes to c, the pair of P_1 and P_2 will be a possible feasible solution. That is by starting from c, moving backward and forward through the graph G along P_1 and P_2, so two nearest parking nodes can be found in linear time. Fig. 4 shows such nodes. If c is a node of degree

Please cite this article in press as: M. Davoodi, et al., An optimal algorithm for two robots path planning problem on the grid, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.07.012

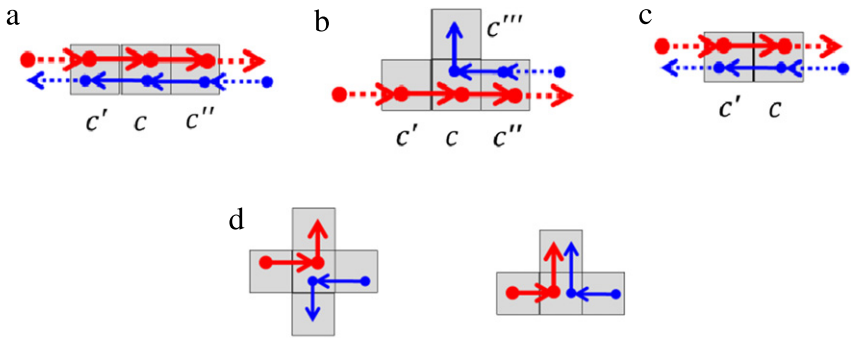

Fig. 2. Schematic view of all types of collisions for two paths red and blue. (a) Head-to-head, type 1, (b) head-to-head, type 2, (c) head-to-head, type 3, and (d) next-to-next collisions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

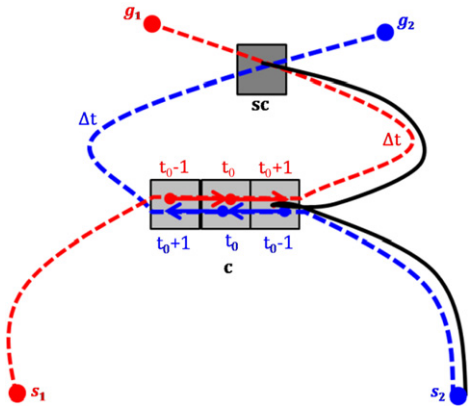

Fig. 3. Counterexample used in proving Lemma 1. If P_1 and P_2 have more than one collision such that one of them is a head-to-head collision, P_1 or P_2 are not shortest paths.

three, the nearest backward and forward parking is c itself. Also, it is possible that only one of the parking nodes does exist (see the example shown in Fig. 1). By using the parking nodes it is possible to find a feasible strategy for the robots.

There are two general strategies for handling the collision using the parking nodes. First, one of the robots follows its path while the other one moves to one of the parking nodes. Therefore, there are two solutions, and the min–max solution can be found in $O(n)$ time. Note that the collision cell c is a parking node in the second type of head-to-head collision (this is exactly why we need to separate type 1 and type 2 collisions). See Fig. 2(b). To handle such a collision the red robot can stay one time step in cell c' before the collision cell c. In fact, in this strategy we accept both the paths and just schedule them.

The second strategy to handle the head-to-head collision between P_1 and P_2 is to accept one of them and try to find another path such that it is not in collision with the accepted path. To this end, we accept P_1 (P_2) and find another s_2–g_2-path (s_1–g_1-path) like Q_2 (Q_1) using BFS while the collision cell is considered as a hypothetical obstacle. Finally, the procedure for handling head-to-head collisions has the following four steps.

Head-to-head collision handling procedure.

Step 1: Use the nearest forward and backward parking nodes to the collision cell and find the best scheduling strategy for P_1 and P_2.

Step 2: Accept P_1 and find Q_2 as an s_2–g_2-path considering the collision cell of P_2 as an obstacle cell.

Step 3: Accept P_2 and find Q_1 as an s_1–g_1-path considering the collision cell of P_1 as an obstacle cell.

Step 4: Select the min–max solution among the solutions obtained with the above steps.

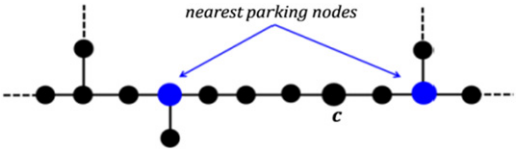

Fig. 4. The forward and backward nearest parking nodes from the collision cell c.

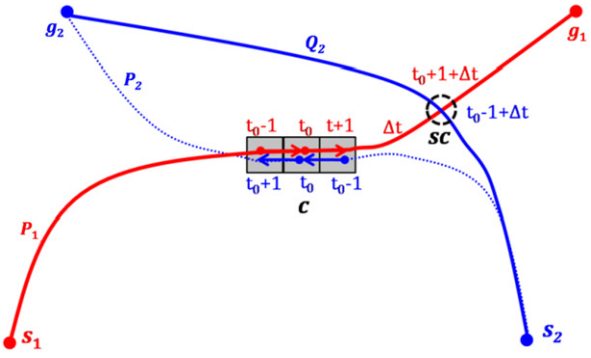

Fig. 5. Concepts used in proof of Lemma 2. Cell sc, the crossing cell between P_1 as the original s_1–g_1-path and Q_2 as an s_2–g_2-path after considering the collision cell c as an obstacle, is not a real (time) collision.

Since the collision between robots is a time collision and paths P_1 and P_2 are the shortest ones, therefore if only one of the robots changes its path and finds another shortest path which does not pass the collision cell, a feasible solution to the problem will be found. The following lemma shows that P_1 and Q_2, and also P_2 and Q_1 which are computed in step 2 and step 3 of the handling procedure are not in collision.

Lemma 2. *Let P_1 and P_2 be two s_1–g_1-path and s_2–g_2-path with a head-to-head collision in cell c at time t_0. Also, let Q_2 be an s_2–g_2-path obtained using BFS after considering c as an obstacle. Then P_1 and Q_2 are not in collision anymore.*

Proof. By contradiction assume that P_1 and Q_2 are in collision at cell sc at time $t_0 + 1 + \Delta t$ for a positive value of Δt. See Fig. 5. So, the minimum time step for the s_1–sc-path is $t_0 + 1 + \Delta t$. Since Q_2 is an s_2–g_2-path subjected not to passing cell c, sc can be reached at most in $t_0 - 1 + \Delta t$ time steps from s_2. So, robots do not lie in sc simultaneously; otherwise it contradicts with the shortest length of Q_2. The proof is similar for negative values of Δt and also for the other types of head-to-head collisions. ■

Based on Lemma 2, not only two pairs of solutions P_1 and Q_2; and P_2 and Q_1 are feasible but also if an optimal solution like Q_1 and Q_2 exists, the procedure will successfully find it or an alternative solution such that the maximum length remains the same. Note

Please cite this article in press as: M. Davoodi, et al., An optimal algorithm for two robots path planning problem on the grid, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.07.012

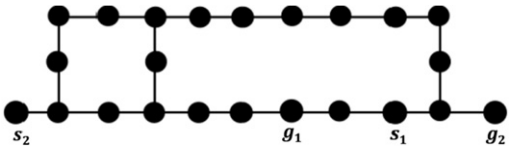

Fig. 6. An exception case of head-to-head collision arises when the collision cell is one of the goal cells. This case can also be handled by checking parking and neighboring cells of the collision cell.

that the lengths of P_1 and P_2 are equal to or less than Q_1 and Q_2, respectively. Thus, if Q_1 and Q_2 is an optimal solution to the MM2rPP problem, then so to are P_1 and Q_2, or P_2 and Q_1.

All the four steps of the head-to-head collision handling procedure run in $O(n)$ time. Therefore, we can conclude this part by the following theorem.

Theorem 2. *If P_1 and P_2 are s_1–g_1-path and s_2–g_2-path with a head-to-head collision, the head-to-head collision handling procedure finds the optimal solution to the MM2rPP problem in $O(n)$ time, where n is the number of free cells in the grid.*

There is an exception in the head-to-head collision when the collision cell is one of the goal cells, g_1 or g_2. See Fig. 6. This case can also be handled similar to the explained procedure except in steps 2 and 3 it is meaningless that the collision cell is considered as an obstacle cell to find an alternative path, because BFS does not find any feasible path when the goal lies on obstacle regions. For handling such an exception, we use neighboring cells of the collision cell to find the alternative solution.

Next-to-next collision

This subsection explains how to handle next-to-next collisions between P_1 and P_2 as s_1–g_1-path and s_2–g_2-path, respectively. Let L_1 and L_2 be the length of P_1 and P_2. Unlike head-to-head collisions it is possible that P_1 and P_2 have more than one next-to-next collision. If one of the robots stays one time step before the first collision, all the other next-to-next collisions will be handled automatically. Note that by so handling the collision both the robots walk consecutively on their paths without a new collision happening. Consequently, if $L_1 \neq L_2$ the procedure for handling the next-to-next collision is simple; the robot with the shorter path stays for one time step before the first next-to-next collision. So, the MM2rPP problem has a solution with a length of $\max(L_1, L_2)$ which is optimal. Thus, in the case of $L_1 = L_2$ it is possible that there exists a min–max solution with length $L = L_1 = L_2$ or in the worst case with length $L + 1$. Therefore, in the following we concentrate on the case $L_1 = L_2$ to find an optimal solution with length L if there exists one; otherwise we find an optimal solution with a length of $L + 1$.

To handle the case $L = L_1 = L_2$, we define a new concept *time collision* (TC) set which is the set of all cells like c such that if P_1 and P_2 with length L enter to c, they lie simultaneously in it and a next-to-next collision between robots occurs. Formally, TC is defined as follows:

$$TC = \left\{ c \in G \left| \begin{array}{l} d_{sp}(s_1, c) = d_{sp}(s_2, c) \text{ and} \\ d_{sp}(c, g_1) = d_{sp}(c, g_2) \text{ and} \\ d_{sp}(s_1, c) + d_{sp}(c, g_1) = d_{sp}(s_1, g_1) \end{array} \right. \right\}, \quad (1)$$

where $d_{sp}(c, c')$ is the length of the shortest path between c and c' in G. In fact, node c belongs to TC if and only if its shortest distances from robots' sources are equal and its shortest distances from robots' goals are also equal and c places on one of the s_1–g_1-path and s_2–g_2-path. By using four BFS starting from sources and goals, TC can be computed in $O(n)$ time. Also, we define $\overline{TC} = G \backslash TC$. Regarding the definition of TC, if P_1 crosses P_2 in a cell $c \in TC$, it will be a time collision, while crossing in $\overline{TC}$ cells is not a real collision because the robots do not reach the cell simultaneously.

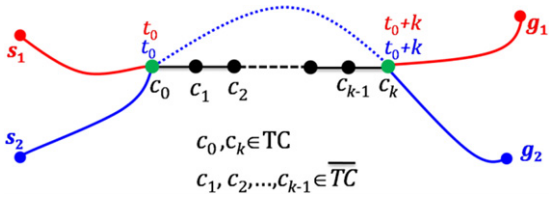

Fig. 7. Concepts used in proving Lemma 4. If $P_1 \cap TC$ is more than one connected component it implies P_1 or P_2 is not the shortest path.

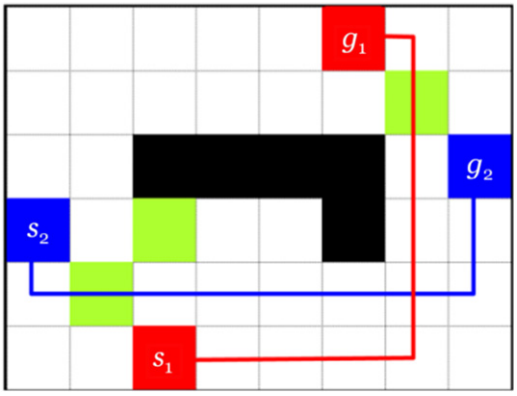

Fig. 8. An example with three disjoint TC components. The lengths of s_1–g_1-path and s_2–g_2-path are 10. The red path uses the top-right component of TC and the blue one uses the bottom-left component.

Lemma 3. *Let P_1 and P_2 be s_1–g_1-path and s_2–g_2-path with the same length that have a next-to-next collision in cell c, then c belongs to TC.*

Proof. According to the fact that any subpath of P_1 and P_2 has the shortest length and the assumption that P_1 and P_2 have a collision in cell c so $d_{sp}(s_1, c) = d_{sp}(s_2, c)$ and $d_{sp}(c, g_1) = d_{sp}(c, g_2)$. Since c lies on P_1 and P_2, based on the definition, c belongs to TC. ■

Corollary 1. *If there exists an s_1–g_1-path like P_1 that does not pass through the TC set, the MM2rPP problem has a solution with length L.*

Based on Corollary 1 if there exists an s_1–g_1-path P_1 (s_2–g_2-path P_2) such that $P_1 \cap TC = \emptyset (P_2 \cap TC = \emptyset)$, there will exist an s_2–g_2-path P_2 (s_1–g_1-path P_1) that does not collide with $P_1(P_2)$. It is true because if there is a cross between P_1 and P_2, it happens in $\overline{TC}$ so it is not a real collision (see Fig. 8 as an example). This fact guides us to find an s_1–g_1-path (or s_2–g_2-path) that has the minimum intersections with TC (Minimize $|P_1 \cap TC|$) or in other words, maximize its intersection with $\overline{TC}$ (Maximize $|P_1 \cap \overline{TC}|$). Before explaining it in detail, we investigate some properties of the TC set.

Lemma 4. *Let P_1 and P_2 be s_1–g_1-path and s_2–g_2-path. $P_1 \cap TC$ is empty or one connected component. It means that P_1 enters into and exits from TC at most once.*

Proof. By contradiction assume $P_1 \cap TC$ is more than one component. See Fig. 7. Suppose first two components. Let c_0 at time t_0 be the last cell of the first component, and c_k at time $t_0 + k$ be the first cell of the second component. So, $c_1, c_2, \ldots, c_{k-1}$ are a set of cells in P_1 that lie in $\overline{TC}$. Since $c_0, c_k \in TC$, $d_{sp}(s_1, c_0) = d_{sp}(s_2, c_0) = t_0$, $d_{sp}(c_0, g_1) = d_{sp}(c_0, g_2)$, $d_{sp}(s_1, c_k) = d_{sp}(s_2, c_k) = t_0 + k$ and $d_{sp}(c_k, g_1) = d_{sp}(c_k, g_2)$. And the length of the shortest path between c_i and c_k is $k-i$, for $i = 1, 2, \ldots, k-1$. So, $d_{sp}(s_1, c_i) = t_0 + i$. Therefore, $d_{sp}(s_2, c_i) \leq d_{sp}(s_2, c_0) + d_{sp}(c_0, c_i) = t_0 + i$, for $i = 1, 2, \ldots, k-1$. On the other hand if $d_{sp}(s_2, c_i) < t_0 + i$, so do $d_{sp}(s_2, c_k) < t_0 + k$ which contradicts with the assumption of

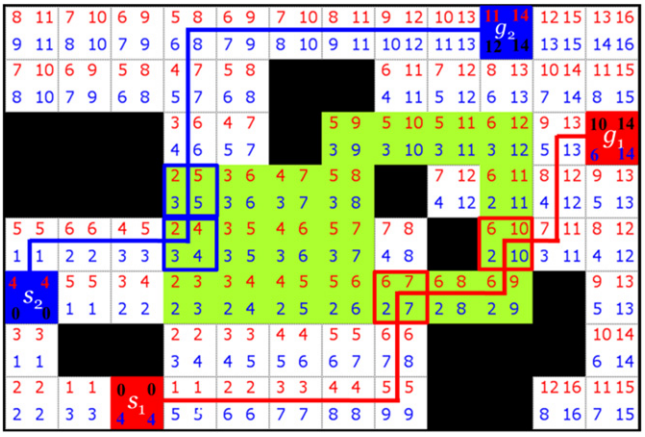

Fig. 9. An example of the MM2rPP problem and a solution for it that has maximum common cells with $\overline{TC}$ (white cells) and uses only the boundary cells of TC. There are four numbers in each cell. Right numbers show the BFS's results for s_1 and s_2 and the left ones show the cv_1 and cv_2. The four solid cells are fs_1, fg_1, fs_2 and fg_2.

$d_{sp}(s_2, c_k) = t_0+k$. Thus $d_{sp}(s_2, c_i) = t_0+i$, that means $d_{sp}(s_1, c_i) = d_{sp}(s_2, c_i)$. Similarly it is proved that $d_{sp}(c_i, g_1) = d_{sp}(c_i, g_2)$, thus all c_i, for $i = 1, 2, \ldots, k-1$, belong to the TC set. Therefore, c_0 and c_k belong to one connected component of TC. ■

Lemma 4 shows that $P_1 \cap TC$ is one connected component; however, TC may be a set of disjoint components. Fig. 8 shows such an example. Based on the lemma, in this case there exist two s_1–g_1-path and s_2–g_2-path that do not collide each other; because we can use one of the components of TC as a middle subpath for s_1–g_1-path and another component as a middle subpath for s_2–g_2-path (this is true because based on Lemma 3, a cross between P_1 and P_2 in $\overline{TC}$ is not a time collision).

Therefore, in the rest of this paper we assume that TC is one connected component which means the shortest path between any pair of cells in TC totally lies interior of TC. In this case we find an s_1–g_1-path P_1 (and similarly an s_2–g_2-path P_2) which satisfies the following conditions:

1- Maximizes $|P_1 \cap \overline{TC}|$
2- Uses only the boundary cells of TC.

We show that if such P_1 and P_2 exist, which are not in collision, the MM2rPP problem has a solution with length L; otherwise the optimal solution to the problem has length $L + 1$. To find P_1 (and P_2) which satisfies the mentioned conditions, we define a value c_{v1} (and similarly c_{v2}) for each cell c which indicates the maximum number of cells belonging to $\overline{TC}$ that lie in any s_1–c-path. More precisely,

$$c_{v1} = \text{Max}\left|s_1\text{–}c\text{-path} \cap \overline{TC}\right|, \quad \text{for all } s_1\text{–}c\text{-path} \tag{2}$$

$$cv_2 = \text{Max}\left|s_2\text{–}c\text{-path} \cap \overline{TC}\right|, \quad \text{for all } s_2\text{–}c\text{-path}. \tag{3}$$

After computing the TC set, values of c_{v1} for each cell c can be computed by using an iterative process (like the BFS procedure) in $O(n)$ time based on Eq. (2). Now we are able to find a path that satisfies the conditions by using BFS's results and c_{v1} values, simultaneously. Also, boundary cells of TC can be determined in $O(n)$ time using a CW turn from the left-bottommost cell of TC to the right-topmost cell of TC to find the upper part of the boundary and similarly a CCW to find the lowest part of the boundary. Fig. 9 shows an example and a solution that satisfies the conditions.

To find an s_1–g_1-path that maximizes $|(s_1\text{–}g_1\text{-path}) \cap \overline{TC}|$, it is necessary for s_1 to be connected to the furthest (with respect to s_1) boundary cell of TC, call it fs_1, in which $(s_1\text{–}fs_1\text{-path}) \cap TC = fs_1$ and similarly g_1 be connected to the furthest (with respect to g_1) boundary cell of TC, call it fg_1, in which $(g_1\text{–}fg_1\text{-path}) \cap TC = fg_1$.

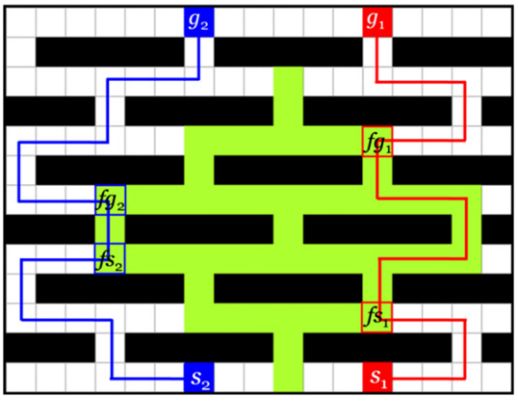

Fig. 10. An example contains TC set (green cells) and the four furthest cells fs_1, fg_1, fs_2 and fg_2. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Finally fg_1 should be connected to the fs_1 using boundary cells of TC. See Fig. 9. Note that we only considered the case that there is no s_1–g_1-path (or s_2–g_2-path) that completely lies inside $\overline{TC}$, and also TC is one connected component, because as mentioned before, in these cases there is a feasible solution with length L based on Lemma 4.

Regarding the definition of TC, it can be interpreted as the intersection of two Voronoi diagrams with Manhattan distance metric on the grid: the Voronoi diagram of s_1 and s_2, and the Voronoi diagram of g_1 and g_2. Also, each cell in TC must lie as a middle subpath of an s_1–g_1-path (or an s_2–g_2-path). According to the property of Voronoi regions, the s_1–fs_1-path lies in the Voronoi region of s_1, similarly the s_2–fs_2-path lies in the Voronoi region of s_2 (except fs_1 and fs_2 which lie on the Voronoi diagram). In the case that there exists a solution with length L, there exists exactly one cell fs_1 (respectively fg_1, fs_2 and fg_2). If there exist two (or more) furthest cells c and c', one of the paths s_1–c-path or s_1–c'-path lies in the Voronoi region of s_2 which contradicts with the definition of fs_1. Therefore, by using cells fs_1, fg_1, fs_2 and fg_2, and boundary cells of TC we can construct paths s_1–fs_1–fg_1–g_1-path and s_2–fs_2–fg_2–g_2-path with length L. Otherwise the length of the optimal solution is $L + 1$. Figs. 9 and 10 show two examples with four fs_1, fg_1, fs_2 and fg_2 cells and the solution such that it satisfies the conditions mentioned in Eqs. (2) and (3). Formally we illustrate these facts in the following lemma.

Lemma 5. *If there is a feasible solution, s_1–g_1-path and s_2–g_2-path, with length L, there exist two paths P_1 and P_2 with length L such that they have maximum intersection with $\overline{TC}$ and use only the boundary cells of TC.*

Proof. Let Q_1 and Q_2 as s_1–g_1-path and s_2–g_2-path be two optimal paths with length L, and let fs_1 and fg_1 be the furthest boundary cells of TC such that $(s_1\text{–}fs_1\text{-path}) \cap TC = fs_1$ and $(g_1\text{–}fg_1\text{-path}) \cap TC = fg_1$. As mentioned before, fg_1 lies on the Voronoi region of g_1 with respect to the Voronoi region of g_2, hence, the subpath g_1–fg_1 lies on the Voronoi region of g_1. Since Q_1 and Q_2 are the feasible shortest paths with the same length, they are not in collision (we assume TC is one component, otherwise as aforementioned we can find the solution by considering a distinct component for each robot path). Therefore, Q_1 partitions the boundary of TC into two parts, the part that contains Q_2 and the opposite one (see Fig. 11(a)). The main idea for proving this lemma is finding P_1 in the opposite part which follows the fact that the feasibility of Q_1 and Q_2 implies the feasibility of P_1 and Q_2. See figure (b). A similar discussion is true for P_2 and Q_1, and consequently, P_1 and P_2 will be a feasible solution (see Fig. 11(f)).

Please cite this article in press as: M. Davoodi, et al., An optimal algorithm for two robots path planning problem on the grid, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.07.012

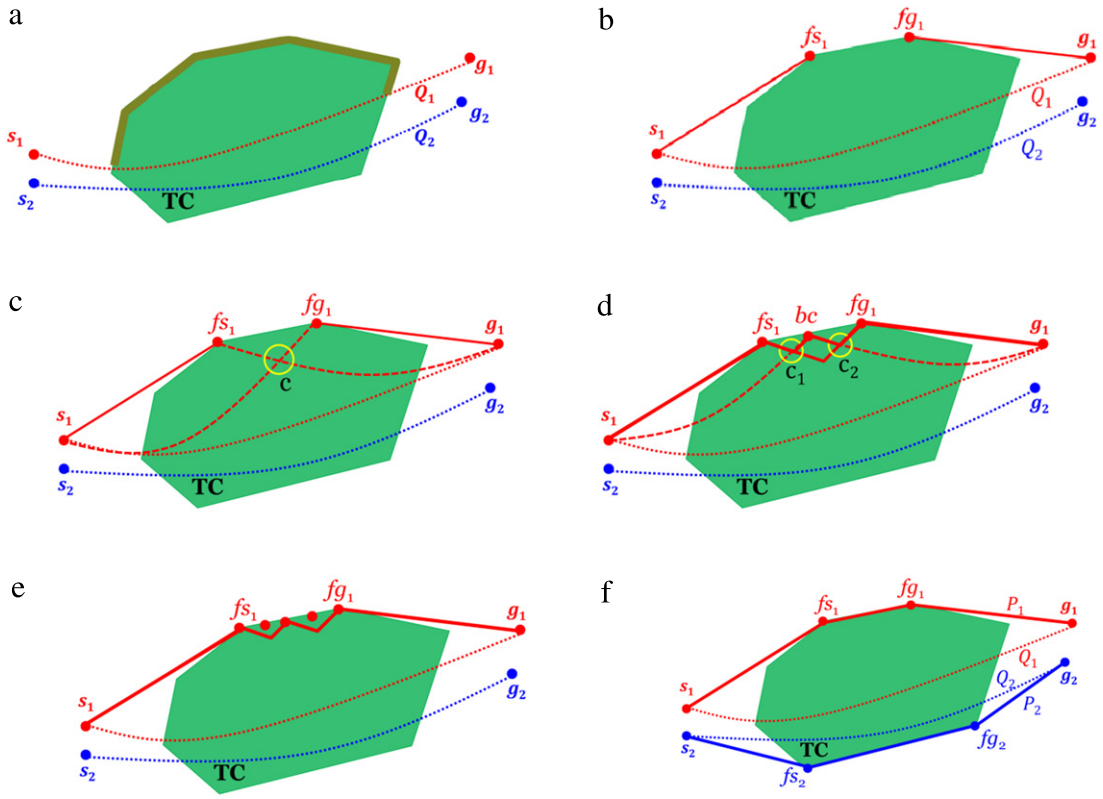

Fig. 11. Concepts used in proving Lemma 5.

Based on the definition of the TC set, the lengths of both paths s_1–fg_1–g_1 and s_1–fs_1–g_1 are equal to L and are not in collision with Q_2 (remember the sequence of nearest to furthest cells of TC with respect to s_1 (or g_1) is either CW or CCW). Let c be the intersection cell between these two paths. See Fig. 11(c). Note that based on the definition of fs_1 and fg_1 these two paths must intersect interior of TC. If it is not the case, so $fs_1 = fg_1$ and the proof is complete. Since $c \in TC$ it is the intersection of two shortest paths s_1–fg_1–g_1-path and s_1–fs_1–g_1-path both with length L, so the path s_1–fs_1–c–fg_1–g_1 has length L, too. We denote this path by P_1'. If subpath fs_1–c–fg_1 of P_1' lies on the boundary of TC, so P_1' is P_1 that has the maximum intersections with $\overline{TC}$ and uses only the boundary cells of TC, thus the proof is complete. Otherwise, it is possible to select a boundary cell between fs_1 and fg_1 like bc. See Fig. 11(d). Similarly path s_1–bc–g_1 intersects with P_1' in c_1 and c_2 and the length of path s_1–$fs_1 - c_1$–bc–c_2–fg_1–g_1 is L. By continuing in this manner (see Fig. 11(e)) we find the path P_1 with length L such that all cells between fs_1 and fg_1 lie on the boundary cells of TC. Similarly, by using fg_2 and fs_2 we find P_2 as the s_2–g_2-*path* which lies on the boundary cells of TC (see Fig. 11(f)). ■

Therefore, if P_1 and P_2 are two s_1–g_1-path and s_2–g_2-path with the same length L such that they are in some next-to-next collision(s), the MM2rPP problem has a feasible solution with length L or $L + 1$. The explained procedure for handling next-to-next collisions finds the optimal solution with length L if it exists. Otherwise, the optimal solution has length $L + 1$, which is to force one of the robots to stop one time step before the first collision.

As the final algorithmic result and for a comprehensive overview we present the algorithm of MM2rPP step-by-step in the following.

Algorithm *Min–Max two robots path planning* (MM2rPP)

Input: A regular grid G as the workspace, and two pairs (s_1, g_1) and (s_2, g_2) as the start and goal cells of robots.

Output: Two collision-free paths s_1–g_1-path and s_2–g_2-path such that minimize the maximum path's length.

Step 0: Determine if G contains some parking nodes or not. If G is a simple chain or a simple cycle solve the problem trivially by finding all (constant number of) possible solutions. Otherwise go to the next step.

Step 1: Ignore temporarily the second robot and find P_1 as an s_1–g_1-path using BFS.

Step 2: Ignore temporarily the first robot and find P_2 as an s_2–g_2-path using BFS.

Step 3: Determine collision cell(s) between P_1 and P_2 using a parallel linear walking on them. If P_1 and P_2 are not in (time) collision, report them as the output and finish the algorithm. Otherwise go to the next step.

Step 4: If P_1 and P_2 are in a head-to-head collision, handle it by using the *Head-to-Head collision handling* procedure and finish the algorithm. Otherwise, if P_1 and P_2 contain some next-to-next collisions, go to the next step.

Step 5: Let L_1 and L_2 be the length of P_1 and P_2, respectively. If $L_1 \neq L_2$, solve the problem by accepting P_1 and P_2 and scheduling the robots by stopping the robot with smallest path length for one time step. Otherwise, if $L_1 = L_2$, go to the next step.

Step 6: Compute TC cells. If TC is more than one connected component, solve the problem by finding two s_1–g_1-path and s_2–g_2-path using distinct components for the first and second robots. Otherwise, if TC is one component, go to the next step.

Step 7: By using fs_1 and fg_1 find P_1 as the s_1–g_1-path which lies on the boundary cells of TC. Similarly, by using fs_2 and fg_2 find P_2 as the s_2–g_2-path which lies on the boundary cells of TC.

Step 8: If P_1 and P_2 are not in (time) collision, without changing their scheduling report them as the output. Otherwise, schedule P_1 and P_2 by stopping the first or the second robot for one time step (there is no priority in this case), and finally report the scheduled paths.

Please cite this article in press as: M. Davoodi, et al., An optimal algorithm for two robots path planning problem on the grid, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.07.012

Let n be the number of free cells in G. Determining whether G contains some parking nodes or not can be done in $O(n)$ time by checking the number of neighbors for each cell. If G is a simple chain or a simple cycle there are at most constant possible solutions (four solutions) for the problem which can be computed in $O(n)$ time. So, step 0 of the above algorithm runs in $O(n)$ time. Also, based on Theorem 1, breadth first searches in step 1 and step 2, and determining collisions in step 3 take $O(n)$ time. The *Head-to-Head collision handling* procedure which was explained before runs in $O(n)$, as well. The remaining steps of the algorithm which are computing TC and finding the boundary paths can be done by using a similar approach like the breadth first search technique in $O(n)$ time. So, the total running time of the algorithm is $O(n)$. Thus, we conclude this section with the final result as the following theorem.

Theorem 3. *The problem of Min–Max two robots path planning with four-points connectivity on the grid can be solved in $O(n)$ time, where n is the size of the grid.*

4. Conclusion

The paper proposes an optimal complete algorithm for solving the Min–Max two robots path planning (MM2rPP) problem on a grid of size n. In this problem the goal is to find two collision-free paths for given pairs of source and goal cells such that the length of the longest path is minimized. Our algorithm finds such a solution in $O(n)$ time if there exists one, otherwise it reports that there is no feasible solution to the problem. Because of the trade-off between time complexity and completeness of the algorithms discussed in multi-robots path planning problems, there are many challenging issues in this context. So, two natural future works of this paper are as follows:

- *The problem of Min–Max three robots path planning (MM3rPP).* Our algorithm works based on the BFS algorithm and efficient collision handling procedures based on collision types. The definition of the parking and TC set in this paper can also be extended for three robots. So, we conjecture that the problem of MM3rPP can be solved in linear time, as well.
- *The problem of Min–Sum two robots path planning (MS2rPP).* In this problem the goal is to find two paths such that the sum of their lengths is minimized. Most of the concepts proposed in this paper are also true in the MS2rPP problem. So, finding a linear time solution in this case is also another interesting problem.

Acknowledgments

We would like to thank A. Frank van der Stappen for his comments in improving this paper, and to him and Steven M. LaValle for helpful discussions on this issue in the 4th winter school on computational geometry in Amirkabir University of Technology.

References

[1] J.E. Hopcroft, J.T. Schwartz, M. Sharir, On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "Warehouseman's problem", International Journal of Robotics Research 3 (4) (1984) 76–88.
[2] D. Ratner, M. Warmuth, Finding a shortest solution for the NN extension of the 15-puzzle is intractable, in: Proceedings of AAAI National Conference on Artificial Intelligence, AAAI-86, 1986, pp. 168–172.
[3] P. Surynek, An optimization variant of multi-robot path planning is intractable, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI-10, 2010, pp. 1261–1263.
[4] M.D. Ardema, J.M. Skowronski, Dynamic game applied to coordination control of two arm robotic system, in: Differential Games—Developments in Modeling and Computation, Springer-Verlag, 1991, pp. 118–130.
[5] D. Parsonsand, J. Canny, A motion planner for multiple mobile robots, in: IEEE Int. Conf. on Robotics and Automation, 1990, pp. 8–13.
[6] J.T. Schwartz, M. Sharir, On the piano movers: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal obstacles, International Journal of Robotics Research 2 (3) (1983) 46–75.
[7] L. Kavraki, P. Svestka, J.-C. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (4) (1996) 566–580.
[8] J.-C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.
[9] P. Svestka, M. Overmars, Coordinated motion planning for multiple car like robots using probabilistic roadmaps, in: IEEE Conf. on Robotics and Automation, Nagoya, Japan, 1995.
[10] M.R.K. Ryan, Multi-robot path planning with subgraphs, in: The 19th Australasian Conf. Robotics and Automation, 2006.
[11] M.R.K. Ryan, Graph decomposition for efficient multi-robot path planning, in: The 20th Int. Joint Conf. on Artificial Intelligence, 2007.
[12] P. Fiorini, Z. Shiller, Motion planning in dynamic environments using the relative velocity paradigm, in: IEEE International Conference of Automation and Robotics, (1), 1993, pp. 560–566.
[13] T.-Y. Li, H.-C. Chou, Motion planning for a crowd of robots, in: IEEE Int. Conf. on Robotics and Automation, 2003, pp. 4215–4221.
[14] J.P. van den Berg, M. Overmars, Prioritized motion planning for multiple robots, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2005, pp. 2217–2222.
[15] P. O'Donnell, T. Lozano-Perez, Deadlock-free and collision-free coordination of two robot manipulators, in: IEEE Int. Conf. on Robotics and Automation, Scottsdale, 1989, pp. 484–489.
[16] S. Leroy, J.P. Laumond, T. Siméon, Multiple path coordination for mobile robots: a geometric algorithm, in: IJCAI, 1999, pp. 1118–1123.
[17] Y. Li, K. Gupta, S. Payandeh, Motion planning of multiple agents in virtual environments using coordination graph, in: IEEE Int. Conf. on Robotics and Automation, 2005, pp. 378–383.
[18] M. Saha, P. Isto, Multi-robot motion planning by incremental coordination, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2006, pp. 5960–5963.
[19] R. Alami, F. Robert, F. Ingrand, S. Suzuki, Multi-robot cooperation through incremental plan-merging, in: IEEE Int. Conf. on Robotics and Automation, 1995, pp. 2573–2579.
[20] T. Standley, Finding optimal solutions for cooperative path finding problems, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, 2010.
[21] E. Gawrilow, M. Klimm, R.H. Mohring, B. Stenzel, Conflict-free vehicle routing, EURO Journal on Transportation and Logistics 1 (1–2) (2012) 87–111.
[22] Z. Shiller, H.H. Lu, Robust computation of path constrained time optimal motions, in: IEEE Int. Conf. on Robotics and Automation, Cincinnati, OH, pp. 144–149, 1990.
[23] Z. Bien, J. Lee, A minimum-time trajectory planning method for two robots, IEEE Transactions on Robotics and Automation 8 (1992) 414–418.
[24] J.E. Bobrow, Optimal robot path planning using the minimum-time criterion, IEEE Transactions on Robotics and Automation 4 (4) (1988) 443–450.
[25] B. Peis, M. Skutella, A. Wiese, Packet routing: complexity and algorithms, in: Proc. of the 7th Workshop on Approximation and Online Algorithms, in: LNCS, Springer, 2009, pp. 217–228.
[26] C.L. Li, S.T. McCormick, D. Simchi-Levi, The complexity of finding two disjoint paths with min–max objective function, Discrete Applied Mathematics 26 (1990) 105–115.

Mansoor Davoodi received his B.S. degree in computer science from Vali-Asr University, Rafsanjan, Iran, in 2005, and the M.S. and 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 areas include: computational geometry, imprecise data modeling and multi-objective optimization.

Marjan Abedin earned her M.S. degree ('2010) in computer science at Amirkabir University of Technology. She is interested in different fields of computational geometry and now is working on arrangements of lines.

Bahareh Banyassady received the B.S. degree in computer science in 2011 from Amirkabir University of Technology (AUT), Tehran, Iran, where she is currently working toward the master degree in computer science. She is a member of the Algorithms and Computational Geometry Laboratory in the Mathematics and Computer Science Department of AUT.

Please cite this article in press as: M. Davoodi, et al., An optimal algorithm for two robots path planning problem on the grid, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.07.012

Payam Khanteimouri received the M.S. degree in computer science from Sharif University of Technology, Tehran, Iran, in 2010. He is a student of the Ph.D. degree in computer science in Amirkabir University of Technology, Tehran, Iran. His current research interests include computational geometry, robotics and categorical data.

Ali Mohades is manager of the computer science group in the Faculty of Mathematics and Computer Science in Amirkabir University of Technology in Tehran, Iran. He is the dean of the Laboratory of Algorithms and Computational Geometry in the Faculty of Mathematics and Computer Science. His main fields of interest are computational geometry, motion planning and facility location.

Please cite this article in press as: M. Davoodi, et al., An optimal algorithm for two robots path planning problem on the grid, Robotics and Autonomous Systems (2013), http://dx.doi.org/10.1016/j.robot.2013.07.012