Multi-Robot Path-Planning Problem for a Heavy Traffic Control Application: A Survey

Ebtehal Turki Saho Alotaibi, Hisham Al-Rawi
Computer Science Department, Al Imam Muhammad Ibn Saud Islamic University
Riyadh, SA

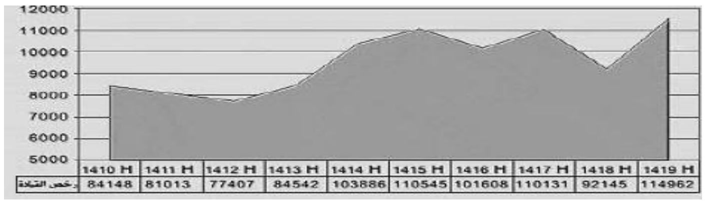
***Abstract*—This survey looked at the methods used to solve multi-autonomous vehicle path-planning for an application of heavy traffic control in cities. Formally, the problem consisted of a graph and a set of robots. Each robot has to reach its destination in the minimum time and number of movements, considering the obstacles and other robots' paths, hence, the problem is NP-hard. The study found that decoupled centralised approaches are the most relevant approaches for an autonomous vehicle path-planning problem for three reasons: (1) a city is a large environment and coupled centralised approaches scale weakly, (2) the overhead of a coupled decentralised approach to achieve the global optimal will affect the time and memory of the other robots, which is not important in a city configuration and (3) the coupled approaches suppose that the number of robots is defined before they start to find the paths and resolve collisions, while in a city, any car can start at any time and hence, each car should work individually and resolve collisions as they arise. In addition, the study reviewed four decoupled centralised techniques to solve the problem: multi-robot path-planning rapidly exploring random tree (MRRRT), push and swap (PAS), push and rotate (PAR) and the Bibox algorithm. The experiments showed that MRRRT is the best for exploring any search space and optimizing the solution. On the other hand, PAS, PAR and Bibox are better in terms of providing a complete solution for the problem and resolving collisions in significantly much less time, the analysis, however, shows that a wider class of solvable instances are excluded from PAS and PAR domain. In addition, Bibox solves a smaller class than the class solved by PAS and PAR in less time, in the worst case, and with a shorter path than PAS and PAR.**

Keywords—component; Heavy traffic control; Multi robots; Coupled Path Planning; Decoupled Path Planning; Collision Avoidance; Heuristics; RRT; Push and Swap; Push and Rotate; Bibox

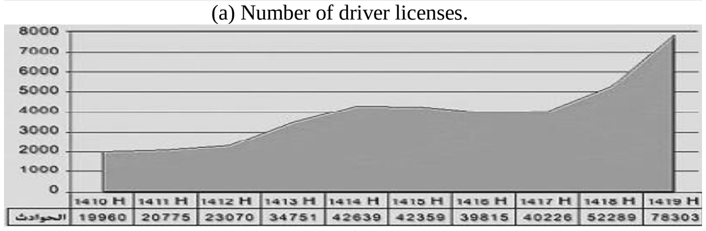
I. INTRODUCTION

As a city's population grows, the number of vehicle accidents increases (Fig.1), which may be caused by the personality of the driver, ignoring a disliked traffic regime, traffic congestion or variations in the speed of the vehicle. Therefore, moving vehicle control to a unified controlling system that optimises all vehicle preferences with respect to the surrounding traffic rules is one way to address part of the problem defined. Multi-robot path planning (MPP) is an abstraction of the problem of finding the complete optimal path from the start point to the target point for each robot with the minimum time and path length while considering the robotic constraints (obstacle avoidance) and inter-robotic constraints (collision avoidance). This survey studies the use of path-planning approaches for MPP for a heavy traffic control problem through a brief review of existing approaches. In addition, it explores four of the main decoupling path-planning approaches. The first is a well-known optimisation technique for single-robot path planning and three are exact methods proposed in the literature. Rapidly exploring random tree (RRT)[1] is an optimisation technique that uses an exploring tree to find the goal and enhance the path for a single robot. An extension to the technique has been implemented in this survey, to fit the MPP definition. Push and rotate (PAR)[2] is an extension of push and swap (PAS)[3]. Both are used to solve any problem with two unoccupied vertices. They are built on two primitives: (1) push on meeting a robot if it has lower priority and (2) swap with it if it has higher priority. Bibox[4] is used to solve any bi-connected graph with two unoccupied vertices. It decomposes the problem into two smaller handles and original cycle and solves them by iteratively moving and locking finished robots and shrinking the problem size. MPP is a relevant problem in a wide range of domains including; automatic packages inside a warehouse [5], automated guided vehicles [6], planetary exploration [7], robotics mining [8], and video games [9].

This paper is organized as follows; after this brief introduction, the multi robot path planning problem is defined from literature in section II. In Section III, the four selected methods is described briefly, followed by Section IV with a discussion of their theoretical analysis. In Section V, the experiments are introduced with a comparative analysis between the selected methods on different types of instances. Finally, conclusions and future work is presented in Section VI.



(a) Number of driver licenses.



(b)Number of accidents

Fig. 1. The latest statistics for Riyadh city between 1410 and 1419 A.H from https://www.moi.gov.sa/wps/portal/Home/sectors/publicsecurity/traffic/trafficriyadh

II. PROBLEM STATMENT

Wilson [10] proposed an efficient decision procedure for the solvability of 15-puzzles, 14-pebble with one unoccupied vertex without considering the number of moves. Therefore his solution required exponentially many moves. Kornhauser et al. [11] generalized Wilson solution to solve Pebble Motion on Graph (PMG) problem with n-1 pebble in $O(n^3)$ for the number of moves. In that work, they defined PMG for number of pebble < vertices. Resultantly a move consists of transferring a pebble to an adjacent unoccupied vertex. In this scenario, the problem is to decide whether one arrangement of the pebbles is reachable within shortest sequence of moves from another or not. The problem defined by Kornhauser et al. can easily mapped to Multi-robot Path Planning (MPP) problem. Mächler [12] found out the differences between PMG and MPP showing PMG models usually assume a central planner aiming to minimize a sequential execution of moves on the graph, there are mainly two distinctions in MPP models; (1) whether the robots execute in parallel or sequential, (2) whether there is a centralized planner or the planning is distributed. Therefore it can be easily concluded that the basic MPP is PMG. Several techniques have emerged in literature to solve sequential MPP. MPP with parallel moves (MPPp) can be defined as a chain of robots that can be moved simultaneously as long as there is an unoccupied vertex at the head on the chain. This particular problem was studied by Ryan in [13] where he separated subgraphs on the base of MPPp. Yu and LaValle [14] added the simultaneous rotation in fully occupied cycle (MPPpr) as natural result for parallel movement. One variant of PMG defined by Yu and Rus [15] allowed simultaneous rotation for sequential problem. They defined a problem of finding a sequence of simple moves and rotations that take initial configuration to the goal configuration thereby transforming into Pebble Motion with Rotation (PMR). PMR model varies from MPPpr in term that the robots are able to move in parallel in case of rotation in fully occupied cycle. The only precondition being that robots can move in sequence. To illustrate the problem, visualize a roadmap $G = (V,E)$, which is a connected graph. Some important variables to be used are; R a set of robots, Γ an initial assignment of robots to vertices $\Gamma: R\rightarrow V$, and a target assignment of robots to vertices $\Phi: R\rightarrow V$. The functions Γ and Φ are total, injective, and non surjective. The path is therefore defined as a map p_i: $Z+\rightarrow V$. A path p_i is feasible for a robot $r_i \in R$ if it satisfies the following properties: (1) $p_i(0) = \Gamma(r_i)$, (2) for each i, there exists a smallest $k_i^{min} \in Z+$ such that for all $k >= k_i^{min}$, $p_i(k) = \Phi(r_i)$, (3) and for any $0 \leq k < k_i^{min}$, $(p_i(k), p_i(k+1)) \in E$ or $p_i(k) = p_i(k+1)$. If $p_i(k) = p_i(k+1)$, then the robot r_i stays at vertex $p_i(k)$ between the time steps k and $k + 1$. We say that two paths p_i, p_j are in collision if there exists $k \in Z+$ such that $p_i(k)=p_j(k)$ (collision on a vertex, or meet) or $(p_i(k), p_i(k+1)) = (p_j(k+1),p_j(k))$ (collision on an edge, or head-on). The problem is to find set of paths P = $\{p_1, \ldots , p_n\}$ such that p_i's are feasible paths for respective robots r_i's and no two paths p_i, p_j are in collision.

The MPP literature uses centralised and decentralised approaches [16]. Each approach can consist of coupled or decoupled robots. Furthermore, due to the basic definition of the problem as NP-hard [17], solution approaches can be categorised based on their completeness for exact and optimisation methods.

In the coupled centralised path-planning approaches, robots act as single robots with multi-bodies and apply a classical single-robot path-planning algorithm. By integrating with complete search methods, such as A*, a coupled algorithm achieves complete and optimal solutions theoretically and probabilistically. In practice, however, the computational time is exponential with the dimension of the configuration space, thus, they are applicable only to small problems. Sharon et al. [18] introduced a new MRPP computing technique by designing the increasing cost tree (ICT) and increasing cost tree search (ICTS). The key idea is to create a snapshot of all robot information (initially the start point of every robot and the cost), represented as states in the ICT attached to the total cost. The tree is spanned for every possible action of every robot. The results show better running time and success rate in large grids.

In the coupled decentralised path-planning approaches, robots apply a distributed MRPP algorithm, decomposing the problem into a set of single-robot problems, which greatly reduces the complexity of each problem and enables the use of single-robot path planners to solve these smaller problems. However, decomposing the problem in a decentralised fashion requires an information sharing and coordination mechanism. The deconfliction technique proposed by Scerri et al. [19] aims to resolve conflicts before they happen. Although this technique produces conflict-free paths, the required synchronisation between robots produces an additional delay. Trodden and Richards [20] propose an improvement in the updating method without synchronisation. In each iteration, one robot re-plans its path and sends it to others, which allows one robot to re-plan in each iteration in order, while all others continue executing their plans to reduce the waiting time. However, a robot with high priority for re-planning has to wait for its turn, which increases the waiting time, and thus, the time cost.

In the decoupled centralised path-planning approaches, paths are planned in two steps: in the first step, each robot's path is calculated individually, and in the second step, the space-time position is calculated to avoid collisions. The first step can be fully distributed while in the second step, there should be communication between robots and priorities may apply in this step. Although these algorithms are distant-optimal, they lose completeness since the paths are calculated completely in the first step, which may contain some conflicts. David [21] proposes a global centralised reservation table where the row for robot i is <location, time>, to avoid conflicts. The table should not contain duplicated values. Although this approach reduces the calculation time, it does not provide a completeness guarantee. A similar reservation table has been used by Stone [6] to solve the car junction problem. Khorshid et al. [22] devised the graph-to-tree decomposition algorithm (GTD). Based on a tree-like graph representation of the problem, they introduced swapping to solve a conflict whenever it occurs.

In conclusion, decoupled approaches run relatively fast, scale well for larger problems but their optimality and even completeness are not always guaranteed [23]. In a coupled approach, the global optimal can be achieved with an overhead of time and computing. Hence, the most relevant problem structure for a heavy traffic control application is one that can be solved by decoupled centralised approaches for three reasons:

- A city is a large environment and coupled centralised approaches scale weakly.
- The overhead of a coupled decentralised approach to achieve the global optimum will affect the time and memory of other robot, which is not important in city configuration.
- A coupled approach supposes that the number of robots is defined before it starts to find the paths and resolve the collisions. However, in a city, any car can start at any time and hence, the car should work individually and resolve collisions as they arise.

III. Methods

When it comes to decoupled centralised approaches, the methods can be classified by their completeness. The exact methods can solve a subclass of MPP completely with a guarantee of the solution. However, despite all of this, their limitation is that they work on only one subclass of the problem. Optimisation methods work on a wider class of the problem than the exact methods and can yield advantage in several aspects such as the path length, execution time or any other metric. Admittedly, optimization methods are not guaranteed to find the solution if there is one.

A. Rapidly-exploring Random Tree (RRT)

A rapidly exploring random tree (RRT) algorithm is designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem. RRTs were developed by LaValle, et al [1]. They easily handle problems with obstacles and differential constraints (nonholonomic and kinodynamic). RRT iteratively expanded by randomly selecting a point q_{rand} in the search space, finding the nearest vertex $q_{nearest}$ to that point implies to the tree, and adding new point q_{new} toward the random point with the edge length equals to Δq (Fig.2).

1) *iRRT for multi-robots*

iRRT is a simple Java program developed by Karaman and Frazzoli [24]. It is dedicated for a single-robot configuration, based on the functionality provided by the iRRT planners. The following extensions are implemented incrementally to fit multi-robot RRT (MRRRT) requirements.

Threads are used to simulate the robots in iRRT to extend the iRRT path planner to support a multi-robot configuration. In addition, robots search based on the A* heuristic where f(n) = min (d + h), such that d is the distance from the start point to n and h is the estimated distance from n to the goal point. A collision is detected by evaluating the slope of a selected random point q_{rand}. The technique used in MRRRT with collision avoidance is asynchronous communication between robots through accessing a synchronized shared data centre (server). Each robot maintains the paths of the other robots – they are uploaded onto the server – while planning. Every robot migrates its final path in the server at the end; hence, there is no waiting and no master robot is required. On this basis, collisions can be avoided by checking iteratively all pairs of vertices in every uploaded path. If the randomly selected point q_{rand} is in the x-axis range and y-axis range between two vertices, then all vertices with the same slope as the edge linking those two vertices with q_{rand} are invalid (Fig. 3). In the experiments, both MRRRT – without heuristics – and MARRT* have been evaluated. Fig. 4 shows part of the results.

B. *Kornhauser's work*

Kornhauser, et al. [11] in their work have provided complete centralized planned procedure to solve all biconnected pebble motion problems with at least one unoccupied vertex. The solutions he has created with upper and lower bounds of O(n^3) moves required on graphs with n vertices and O(n^3) time. However, there is no a single algorithmic description of Kornhauser procedure in the literature. Instead, some works are focused on implementing and improving Kornhauser idea. Moreover, Kornhauser proves a feasibility test of the problem instance in the form of a tree of biconnected (non-separable) components that are linked by chains of vertices with degree two (called isthmuses) as; "*It is impossible for robots to swap if they are separated by an isthmus longer than the number of unoccupied vertices minus two*".

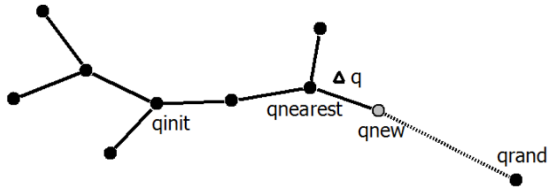


Fig. 2. RRT expansion method

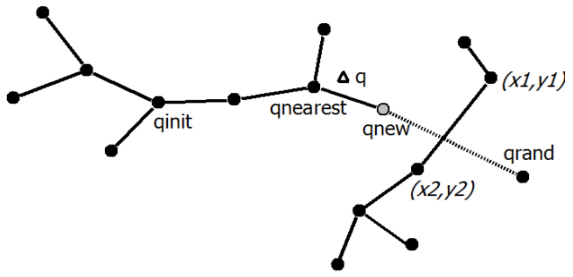


Fig. 3. Validate RRT expansion to avoid collision, if (y_i-y_1/x_i-x_1)= (y_i-y_2/x_i-x_2) AND *xi* ∈{*x1*,*x2*} AND *xi* ∈{*y1*,*y2*} then invalid expansion

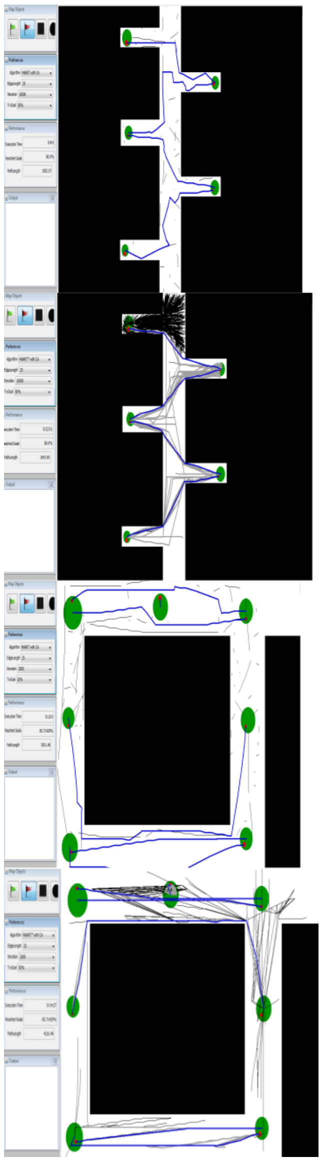


Fig. 4. MARRT paths (left) versus MARRT* paths (right)

To clarify Kornhauser result, consider sequential moves only, Fig.5 shows an instance containing both solvable and unsolvable instances based on Kornhauser condition. Note that it is possible for any of the robots in A1 = {a1…,a6} to swap their positions. The same goes true for the robots in A2 = {a7,a8} as they can exchange their positions too. For example, robot 5 and robot 6 can swap their positions while robot 7 and robot 8 can swap their position. The condition to be met is that no robot from A1 can exchange position with any robot in A2. To understand this, again referring to Fig. 5, if robot 7 wants to swap its position with robot 6, it can be possible only if they move to either the component at the left side of the vertex occupied by robot 6 or to the component at the right side in order to execute swap operation. This swapping would invariably fail due to the reason that during the swapping robots passing the bridge to the left component would stack all robots on it. Hence essentially after this, there will be no possibility to swap unless there are at least enough free vertices to fit all opponent robots on the bridge as well as the two swapping robots. The same thing would happen if the swapping robots pass the bridge to the right component.

C. Push and Swap

Push And Swap (PAS) algorithm provides complete MPP algorithm for problem instances with at least two unoccupied vertices. However, it considers the priority between robots which can cause a limitation and has been solved later in another publication [25]. For each robot *a*, the PAS algorithm finds the shortest path *p** linking the start location of *a* to its goal location, advances the robot through *p** by push and swap algorithms. PAS creates the solution of the problem of n vertices and k robots in O(n^4) time.

1) Push

While the next vertex in the shortest path *p** of robot *a* is unoccupied, push algorithm will advance robot *a*, when another robot *b* is detected on vertex *v* in *a*'s path, the robot *a* has the ability to push it away if it has a lower priority (Fig. 6). If it has a higher priority, the push operator fails, hence, PAS will switch to the swap operator.

2) Swap

When another robot *b* is detected on vertex *v* in *a*'s path and b has higher priority than a. Both robots will advance to the nearest vertex *u* such that the degree of *u* is more than or equal to 3. If the vertex *u* has less than two unoccupied neighbor vertices, then, two occupied vertices should be cleared regardless of the occupying robots' priority (Fig.7). Considering robots *a* and *b* as obstacles, then, robot *a* will swap with robot *b* (Fig. 8). Finally, each robot is affected by the clear operation, which will be resolved by moving it back to its previous location.

D. Push and Rotate

Push and Rotate (PAR) algorithm[2] is a complete version of the PAS algorithm. It creates the solution of the problem of *n* vertices and *k* robots with O(*k*.n^3) moves and in O(*k*.n^5) time. It solves any solvable instance recognized by Kornahuser.

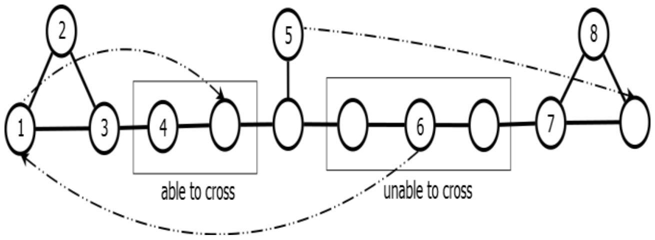


Fig. 5. Graph with solvable and unsolvable instance based on Kornhauser result

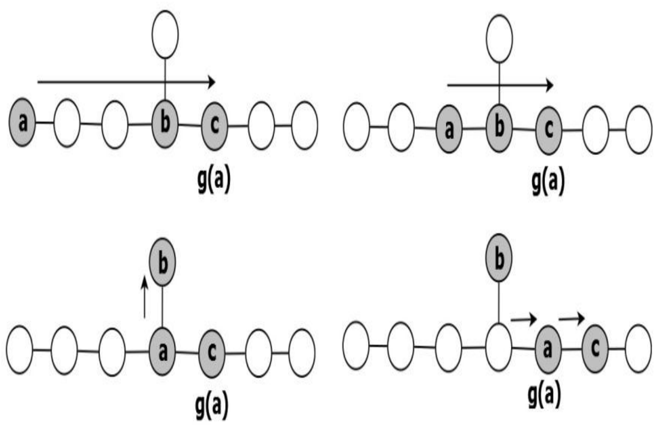


Fig. 6. Push operator: Robot a faces b through its p^* , and b has lower priority than a, robot a pushes b away from its p^*

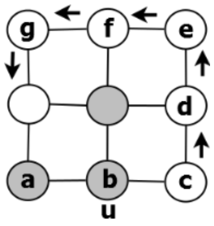


Fig. 7. Robots *a* and *b* are advanced to vertex *u* to achieve swap operation, vertex *u* contains only one unoccupied neighbor, therefore, it should clear the other neighbor. To do it, it should not move *a*, *b* or the unoccupied neighbor, hence, all those vertices will be considered as obstacles for the cleared robots

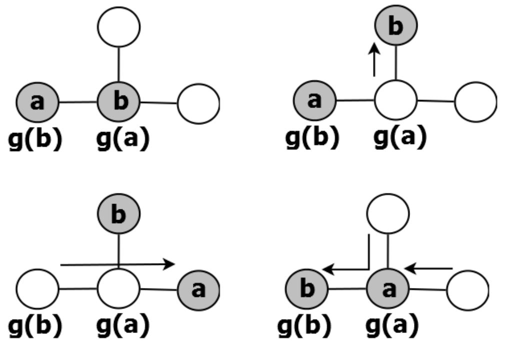


Fig. 8. Swap operator: Robot *a* faces *b* through its SP and *b* has higher priority than *a*, robot *a* and *b* are moved to vertex *u* with degree more than two and swap

result as the instances with number of unoccupied vertices more than to the bridge length minus two. PAR reported four issues contradicting PAS claims. These contradictions will be described in this section exactly as it was presented in [2].

1) Push and rotate proved that multi robot path planning problem can be solvable when Swap fails. The Polygon graph with two unoccupied vertices is a graph conform to Push and Swap requirements. However, MPP problem is solvable in polygon even though Swap fails due to the absence of vertex with degree more than two (Fig.9).

Another example to prove that the problem can be solved when Swap fails is the isthmus graph. When the graph satisfies Push and Swap requirements, under some configuration of robot ordering, the isthmus contains vertex with degree more than or equal three and it is solvable but Swap fails (Fig.10).

2) Push and rotate proved that Clear operation doesn't consider some possibilities when evacuating two vertices in neighborhood of a vertex v.

3) Resolve algorithm in PAS executes recursively to return each robot effected by Swap operator back to its previous location which may lead to swap with other robot. In this case, Resolve algorithm will turn to the swapped robot to resolve it, This will also change the last robots locations, resulting in moving the robots two steps away from their location. This is a possibility which is not considered by Resolve algorithm.

4) Push and Rotate proved that there are new redundant moves resulting after executing Smooth operator in PAS, which may be required to execute Smooth again.

In addition, PAR introduces the rotate operator to resolve cascade moves. The rotate operator is called to move robots forward in a cycle. To do so, one vertex should be cleared by swapping it with a neighboring vertex. Then, all the robots are rotated in the clockwise direction so that any robot affected by the clear operator will be resolved. Fig. 11 describes rotation procedure.

E. Bibox

The Bibox algorithm [4] solves biconnected graphs completely. Initially Bibox decomposes the problem to many handles and one original cycle (Fig. 12). Then, robots with goal locations at the outer handles are solved earlier, and they will be locked and excluded from the search space. This results in a smaller problem, and the original cycle will be solved in different ways. Bibox creates the solution of the problem of n vertices in O(n^3) time and O(n^3) moves. However, Bibox always solves the graph considering the worst scenario. If there are more than two free vertices, it fills those vertices with dummy robots, solves them, and removes their solutions from the final solution. This permits the algorithm of utilise additional free vertices, hence, permits any improvement.

1) Solve handle

Solving a handle means bringing the robots whose goal vertex is the handle, starting from the vertex at the beginning of the handle one by one until all vertices in the handle are finished. If a robot is located outside the handle, bring it to the

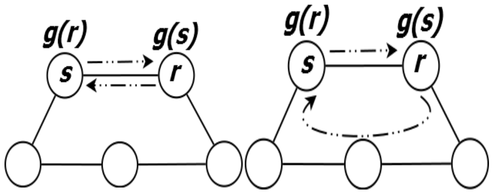


Fig. 9. Polygon graph satisfies Push and Swap requirements (two unoccupied vertices), robot *s* pushes robot *r* to reach its goal position through its p^*,then, *s* is finished robot, *r* tries to push *s* to reach its goal through its p^* and it fails, swap operator will fail also since there is no vertex with degree more than two in Polygon (right), hence, this graph will be unsolvable in Push

and Swap context while it is solvable if there is an alternative path for the robot to follow it (left)

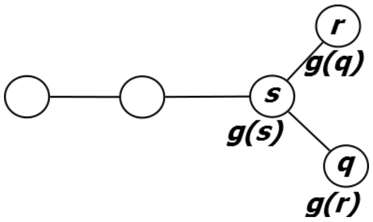


Fig. 10. Graph with isthmus is solvable in Push and Swap context if robot *s* has lower priority among others only

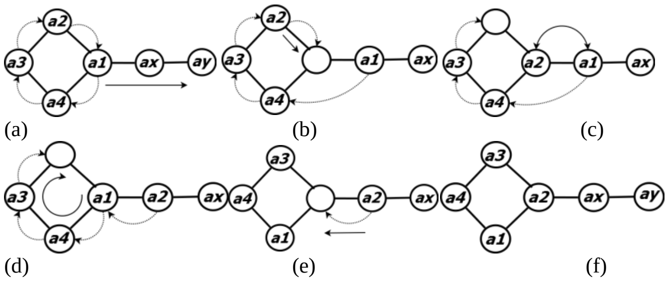


Fig. 11. Rotate operator execution, in the worst case, all vertices in the cycle are occupied

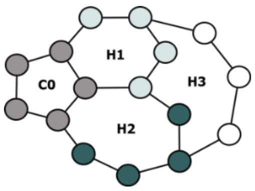


Fig. 12. Decomposing biconnected graph to *i* handles (H_i) and original cycle (C_0)

entering vertex, u, lock the handle to protect the finished vertices from any arbitrary moves, move it to its goal with the rotate operator, and unlock the handle. The idea is that the handle keeps the finished vertices at the beginning, which means they will be rotated and reverse rotated to ensure their robots are back in their locations (Fig. 13). If a robot is located in the handle, rotate the completed vertices until the robot reaches the entering vertex. Advance the robot to any free outer vertex, rotate in the reverse direction by the same path traversed before to protect finished vertices, and continue considering the robot as in the outer robot case (Fig. 14).

2) Solve original cycle

The original cycle will be solved at the end, when the problem consists of a cycle with two unoccupied entering vertices, and all robots in it have their goal inside it also. For each two robots that want to swap, rotate them to the entering vertex, exchange their locations, and reverse rotate to restore the previous locations (Fig. 15).

IV. Analysis

Even though Kornhauser procedure provides complete solution for wide class of problems, it unrecognized wide set of solvable instances by simultaneous rotation. This is because it considers sequential moves only. We can see it vividly, when the instance containing number of unoccupied vertices equal to the maximum bridge length and the problem is defined as MPPp, MPPpr or PMR. In that case the instance is solvable. However, while the bridge length increases, the number of unrecognized solvable instances increases. For instance, the instance described in Fig.16. is a solvable instance in MPPp, MPPpr, PMR boundaries even though the number of unoccupied vertices equal the maximum bridge.

To prove the solvability of such instances, there are two main cases for robot *r* current position and goal position with all other situations being the subcases of these two. The cases are: (1) Current position of robot *r* and its goal position are at a cycle. According to its definition, simultaneous rotation would always be able to pass a robot occupying a vertex in the cycle to its goal on the same cycle even if the cycle is fully occupied, (2) Current position of robot *r* is at a cycle and its goal position is at other cycle. Let *G* be an instance containing two cycles C_1 and C_2, a bridge with length *bl*, and unoccupied vertices *m* equal *bl*. In the worst case, the bridge is fully occupied, the robot *r* at the entrance vertex of C_2 and its goal position inside C_1. Consider two main configuration in this case; (i) All unoccupied vertices are inside C_2 while C_1 is fully occupied (Fig.16). In this configuration, robot *r* would be occupying vertex *v*, which is the entrance to C_2. However, the feasibility would still be guaranteed by moving robot *r* away one step to -*v*, followed by shifting all robots in the bridge to C_2, and passing robot *r* toward C_1 through unoccupied bridge. Once r reaches the head of the bridge, which is the entrance to C_1, the robot will be treated as Case.1. (ii) The unoccupied vertices are divided between C_1 and C_2. In this configuration, robot *r* would be occupying vertex *v*, which is the entrance to C_2. However, the feasibility of the problem would still be guaranteed by moving robot *r* away one step to –*v* followed by shifting half of robots in the bridge to C_2. Once this achieved, robot *r* would be passed towards C_1 through half unoccupied bridge and shifting all opponent robots in the second half toward C_1. Once *r* reaches the head of the bridge, which is the entrance to C_1, the robot will be treated as Case.1. This results can be generalized to grid and bi-connected graphs, when they are contain set of cycles connected by bridge of length one (an edge). Hence, one unoccupied vertex is enough to make the graph solvable.

The MRRRT algorithm appear as favorable optimization algorithms due to the reason that they don't impose any restriction on the roadmap. In addition, the PAS and PAR algorithms appear as preferable algorithms due to its completeness guarantee. Furthermore, the technique of Bibox algorithm is also suited since it is powerful for smaller class of instances, with higher performance. Despite the fact that these algorithm seem most appropriate among the decoupled centralised approaches, these have their own certain theoretical issues;

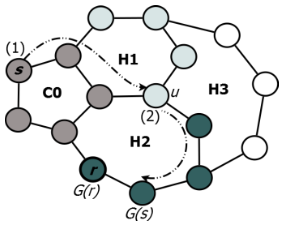


Fig. 13. SolveHandle Case.1, if the robot outside the handle

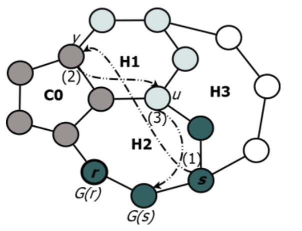


Fig. 14. SolveHandle Case.2, if the robot inside the handle

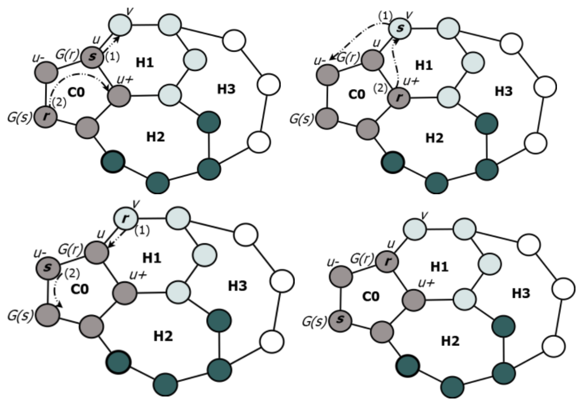


Fig. 15. Solve original cycle

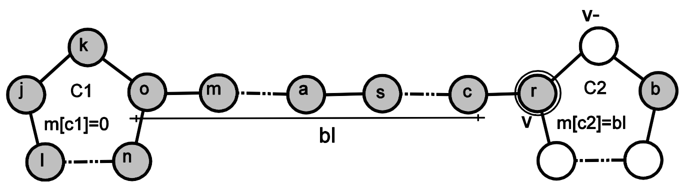


Fig. 16. Solvable instance unrecognized by Kornhauser

1) Kornhauser procedure unrecognized wide set of solvable instances by simultaneous rotation.

2) Since PAR is limited to the solvable instances recognized by Kornhauser procedure, it failed to solve a solvable instance with number of unoccupied vertices equal to the bridge length.

3) Bibox is limited in application since it provides complete solution for only bi-connected graph with two unoccupied vertices.

4) Bibox always solves the graph considering the worst scenario. If there are more than two unoccupied vertices, it fills those vertices with dummy robots, solves them, and removes their solutions from the final solution. This permits the algorithm to utilize additional unoccupied vertices, hence, permits any optimization.

5) There are wider class of solvable instances with only one unoccupied vertices, which is excluded from PAS, PAR and Bibox assumption.

V. Experimental Results and Discussions

The end objective was to answer the question: what is the benefit of these algorithms? To answer this question, three main factors of the problem are evaluated; (1) the algorithms' execution time to find the whole path, (2) the rate of success results in a fixed time period, (3) and the total path length calculated mathematically as the sum the number of vertices composing the path. All experiments are employed in Ubuntu Dell M5110 laptop, Intel(R) Core(TM) i7 CPU@ 2.20GHz. A series of experiments were conducted to compare between MRRRT, MRRRT*, PAS, PAR and Bibox algorithms. The factors are examined ten times on biconnected graph, tree-with-cycle-leaves (TWCL) and six benchmark problems proposed in [3] that describe different scenarios of intersecting paths (Fig.17).

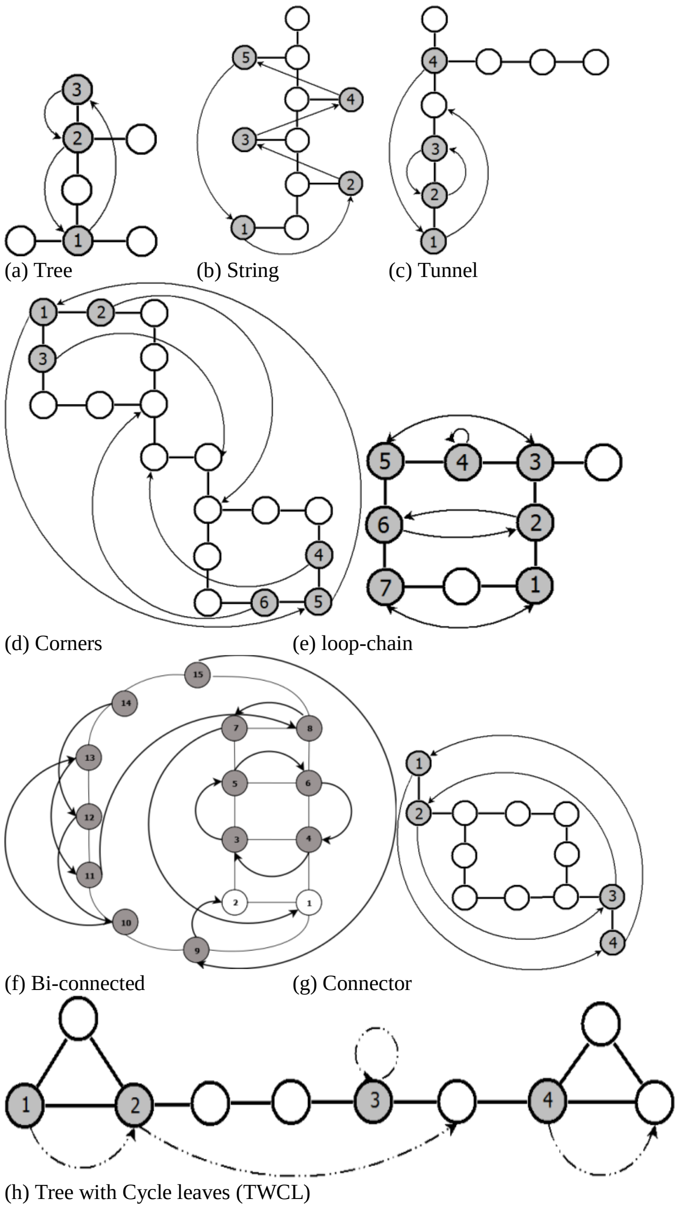


Fig. 17. Benchmark problems

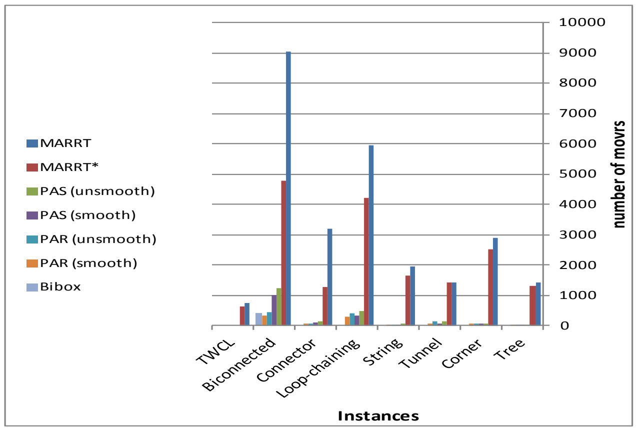


Fig. 18. Path length defined by number of moves

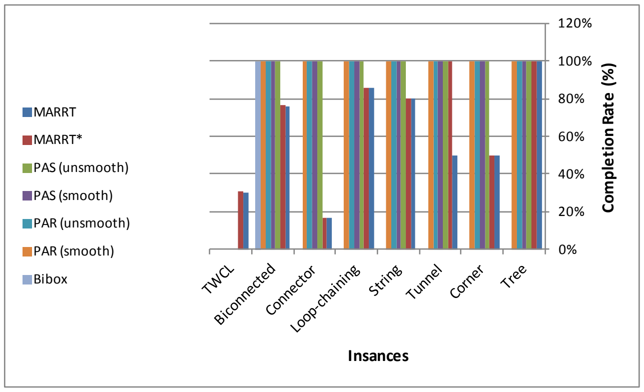


Fig. 19. Completion rate defined as the ratio of reached robots to all robots

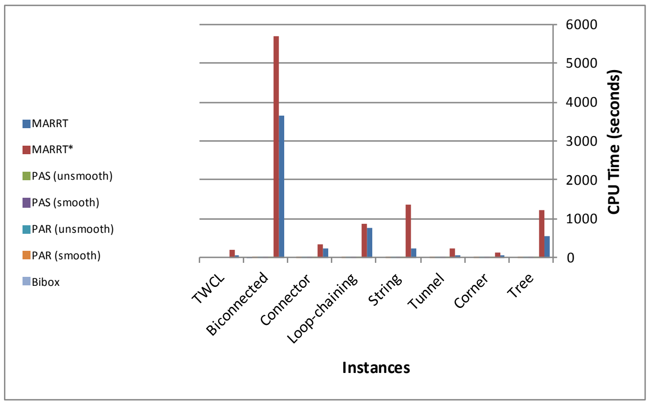


Fig. 20. CPU time for all algorithms

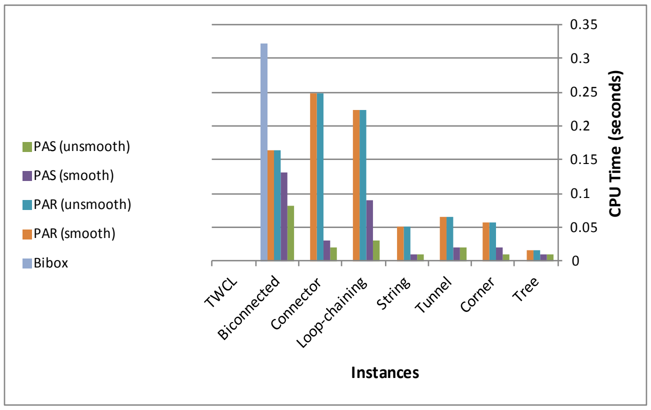


Fig. 21. CPU time for all algorithms except MRRRT and MRRRT* algorithms (To see the differences between others)

The results in Figs. 18–21 show that, for all problems, MRRRT* outperforms MRRRT in finding shorter paths for the robots since it uses the A* heuristic. However, MRRRT* consumes more computation time than MRRRT mainly for tree and biconnected problems, since it explores all valid paths then selects the shortest one. On the other hand, all PAS and PAR versions outperform MRRRT versions in minimising the path length due to the basic difference between the planners, which is that MRRRT works in an unknown environment while PAS works in a fully known environment, which reduces the overhead of path exploration. For the same reason, the execution time for PAR is much less than for MRRRT. Furthermore, all PAS and PAR versions outperform MRRRT versions in the total number of achieved goals due to the completeness guarantee provided by PAS and PAR and since MRRRT reserves the full path of a robot, which reduces the space for other robots. However, the higher-priority robots may permit a lower-priority robot to achieve the shortest path. In addition, Bibox has better calculated path lengths than other algorithms in the class it solves. Finally, PAS and PAR fail to solve the TWCL class, which is a subclass of the solvable problems since it contains bridge length equals to the number of unoccupied vertices minus one while MRRRT and MRRRT* algorithms solve that instances since they don't impose any restriction on the graph topology.

VI. Conclusion

In this survey, the MPP literature for heavy traffic control was briefly reviewed. The closely related structure was discussed and a practical comparison was done between an optimisation technique, RRT, and three of the exact techniques, PAS, PAR and Bibox. Their specifications, strengths and weaknesses were compared, as summarised in Table 1. The experiments showed that MRRRT is the best for exploring any search space and optimizing the solution. On the other hand, PAS, PAR and Bibox are better in terms of providing a complete solution for the problem and resolving collisions in significantly much less time, the analysis, however, shows that a wider class of solvable instances are excluded from PAS and PAR domain. In addition, Bibox solves a smaller class than the class solved by PAS and PAR in less time, in the worst case, and with a shorter path than PAS and PAR. Based on the results, future work may look at these weaknesses in Table.1 as holes for contributions.

TABLE I. SUMMARY

Method / Criterion	MRRRT* [1]	PAS [3]	PAR [2]	Bibox [4]
Environment information	Unknown/ dynamic	Unknown/ dynamic	Known/ dynamic	Known/static
Assumption	Any problem instance	Graph with at least two unoccupied vertices.	Graph with at least two unoccupied vertices.	Biconnected graph with two unoccupied vertices.
Completeness	Incomplete	Complete (except the instances reported in [2]).	Complete	Complete
Path Complexity	-	-	O(n^3 . k)[2]	O(n^3)
Time Complexity	Since it works in unknown environments and selects the shortest path among different paths, the time is the longest among other planners	-	O(n^5 . k)[2]	O(n^3)
Strengthens	- Capability to explore unknown environment. - The use of online path planner (CL-RRT) makes the proposed planner applicable to the dynamic environment. - Finds the shortest path among different paths by A* heuristic.	- Simple. - The experiments show high efficient results in term of the computation time, solution length and the success rate with high scalability on most of the cases. - Merges the advantages of decoupling approaches in terms of fast calculations, and the advantages of coupling approaches in term of local robots negotiation	- Simple -The experiments show high efficient results in term of The computation time, solution length and the success rate with high scalability in all cases. -Merges the advantages of decoupling approaches in terms of fast calculations, and the advantages of coupling approaches in term of local robots negotiation	- Achieves the least time and path length proved by Kornhauser.
Weaknesses	- The priority of the higher robots may permit a lower robot to finds its paths - Cooperation process may add additional computing overhead. -Optimizing the path may add additional computing overhead	- The priority of the higher robots may permit a lower robot to achieve shortest paths. -There are wider class of solvable instances which is excluded from PAS assumption.	- The priority of the higher robots may permit a lower robot to achieve shortest paths. -There are wider class of solvable instances which is excluded from PAR assumption.	Solves the instance in its worst case, which permit any improvement.

REFERENCES

[1] S. M. LaValle, "Rapidly-Exploring Random Trees A New Tool for Path Planning," 1998.
[2] B. d. Wilde, A. W. ter Mors, and C. Witteveen, "Push and Rotate: a Complete Multi-agent Pathfinding Algorithm," Journal of Artificial Intelligence Research, pp. 443-492, 2014.
[3] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," in IJCAI, 2011, pp. 294-300.
[4] P. Surynek, "A novel approach to path planning for multiple robots in bi-connected graphs," in Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, 2009, pp. 3613-3619.
[5] E. Guizzo, "Three Engineers, Hundreds of Robots, One Warehouse," IEEE Spectr., vol. 45, pp. 26-34, 2008.
[6] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," Journal of Artificial Intelligence Research, pp. 591-656, 2008.
[7] J. Leitner, "Multi-robot cooperation in space: A survey," in Advanced Technologies for Enhanced Quality of Life, 2009. AT-EQUAL'09., 2009, pp. 144-151.
[8] J. M. Roberts, E. S. Duff, and P. I. Corke, "Reactive navigation and opportunistic localization for autonomous underground mining vehicles," Information Sciences, vol. 145, pp. 127-146, 2002.
[9] D. Nieuwenhuisen, A. Kamphuis, and M. H. Overmars, "High quality navigation in computer games," Science of Computer Programming, vol. 67, pp. 91-104, 2007.
[10] R. M. Wilson, "Graph puzzles, homotopy, and the alternating group," Journal of Combinatorial Theory, Series B, vol. 16, pp. 86-96, 1974.
[11] D. Kornhauser, G. Miller, and P. Spirakis, Coordinating pebble motion on graphs, the diameter of permutation groups, and applications: IEEE, 1984.
[12] P. Mächler, "Pebbles in Motion Polynomial Algorithms for Multi-Agent Path Planning Problems," Master of Science in Computer Science, Department of Mathematics and Computer Science, University of Basel, Basel, 2012.
[13] M. R. K. Ryan, "Exploiting subgraph structure in multi-robot path planning," Journal of Artificial Intelligence Research, pp. 497-542, 2008.
[14] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in Algorithmic Foundations of Robotics X, ed: Springer, 2013, pp. 157-173.
[15] J. Yu and D. Rus, "Pebble motion on graphs with rotations: efficient feasibility tests and planning algorithms," in Algorithmic Foundations of Robotics XI, ed: Springer, 2015, pp. 729-746.
[16] J.-C. Latombe, "Robot Motion Planning, Chapter," 1996 1991.
[17] C. H. Papadimitriou, P. Raghavan, M. Sudan, and H. Tamaki, "Motion planning on a graph," in Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on, 1994, pp. 511-520.
[18] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," Artificial Intelligence, vol. 195, pp. 470-495, 2013.
[19] P. Scerri, S. Owens, B. Yu, and K. Sycara, "A decentralized approach to space deconfliction," in Information Fusion, 2007 10th International Conference on, 2007, pp. 1-8.
[20] P. Trodden and A. Richards, "Robust distributed model predictive control using tubes," in American Control Conference, 2006, 2006, p. 6 pp.
[21] D. Silver, "Cooperative Pathfinding," in AIIDE, 2005, pp. 117-122.

[22] M. M. Khorshid, R. C. Holte, and N. R. Sturtevant, "A polynomial-time algorithm for non-optimal multi-agent pathfinding," in Fourth Annual Symposium on Combinatorial Search, 2011.
[23] Q. Zhu, J. Hu, W. Cai, and L. Henschen, "A new robot navigation algorithm for dynamic unknown environments based on dynamic path re-computation and an improved scout ant algorithm," Applied Soft Computing, vol. 11, pp. 4667-4676, 2011.
[24] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The International Journal of Robotics Research, vol. 30, pp. 846-894, 2011.
[25] Q. Sajid, R. Luna, and K. E. Bekris, "Multi-Agent Pathfinding with Simultaneous Execution of Single-Agent Primitives," in SOCS, 2012.