A Survey on Path Planning Algorithms for Mobile Robots

Márcia M. Costa
Department of Electrical Engineering
School of Engineering of the Polytechnic Institute of Porto
Porto, Portugal
marcia_costa23@hotmail.com

Manuel F. Silva
INESC TEC and ISEP-IPP
Porto, Portugal
mss@isep.ipp.pt

***Abstract*—The use of mobile robots is growing every day. Path planning algorithms are needed to allow the coordination of several robots, and make them travel with the least cost and without collisions. With this emerged the interest in studying some path planning algorithms, in order to better understand the operation of each one when applied in this type of robots. The objective of this paper is to present a state of the art survey of some algorithms of path planning for mobile robots. A brief introduction on mobile robots and trajectory planning algorithms is made. After, the basis of each algorithm is explained, their relative advantages and disadvantages are presented and are mentioned areas of application for each of them. This study was developed in order to implement some of these algorithms in the near future, with the objective to find out their relative advantages and disadvantages, and in which situations their implementation is more adequate.**

***Index Terms*—Algorithms, mobile robots, path planning.**

I. Introduction

Mobile robots can be found in industry, military installations, security environments, and as consumer products, whether for entertainment or to perform some work, such as vacuum cleaning or cutting grass. They have the ability to move around their environments and are not attached to a physical location. According to the environment in which they move, they can be classified into three types: (*i*) terrestrial robots, usually have wheels, but some have legs, like humans, quadruped animals or arthropods; (*ii*) aerial, usually referred to as Unmanned Aerial Vehicles (UAV); and (*iii*) underwater, often referred as Autonomous Underwater Vehicles (AUV) or Remote Operated Vehicles (ROV) [1].

The path planning algorithms for these autonomous vehicles can include aspects such as the planning of movements between obstacles and the coordination of movement with other mobile robots. Thus, these algorithms aim at choosing the route that usually takes less time and that presents less costs for the mobile robots to accomplish the intended tasks [2].

This work is financed by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the FCT Fundao para a Ciłncia e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

978-1-7281-3558-8/19/$31.00 ©2019 IEEE

This paper is organized as follows. Section II presents some path planning algorithms for different types of mobile robots. Section III discusses the studied algorithms, their advantages and disadvantages, and their application areas. The conclusions and future work are drawn in Section IV.

II. Path Planning Algorithms

In this section, are described path planning algorithms that can be applied to mobile robots, the relative advantages and disadvantages of the analysed algorithms, as well as examples of some areas in which they have been applied.

A. A star

The A star (A*) is a search method that uses a heuristic function, $h(n)$, where n represents a node n. To each node n is associated an approximation $h(n)$ of the cost of a path from n to a goal node, while $h^*(n)$ corresponds to the real distance (cost) from n to a goal node. A heuristic h is consistent if and only if: (*i*) $h(n) = 0$ (if n is the goal node); and (*ii*) for all nodes and their successors n', the estimated cost of moving from node n to the goal node is not greater than the cost of moving from node n to node n' plus the estimated cost of moving from node n' to the goal node, as can be seen in Equation 1 [3].

$$h(n) \leqslant c(n, n') + h(n') \tag{1}$$

A heuristic h is admissible when $h(n)$ underestimates $h^*(n)$, that is, it respects Equation 2 [3]. The heuristic to be used may be the straight line distance, or the euclidean distance [4].

$$h(n) \leqslant h^*(n) \tag{2}$$

The other functions of this method are $g(n)$, that denotes the cost of the path from the start node to node n, and $f(n)$, which represents the estimated cost of the path passing through n to reach the goal node. $f(n)$ is defined as the sum of $g(n)$ with $h(n)$, as in Equation 3 [3].

$$f(n) = g(n) + h(n) \tag{3}$$

The main advantage of this algorithm results from the fact that using the heuristic, the algorithm can quickly converge [5].

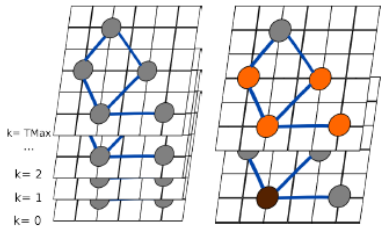


Fig. 1. Temporal graphs (left) and AGV positions (orange circles) in each temporal layer of the TEA* algorithm (right) [9].

The disadvantages include (*i*) not consider obstacles for preventing collisions [6]; and (*ii*) be slow in searching speed and be of poor applicability in the large scale paths search [7].

The A* algorithm has been applied in several areas, among which can be mentioned the areas of automation / robotics (for trajectory planning of AGV in Smart Park [5]), medicine (for needle penetration during surgery procedures [6]) and games (for determining the paths in games [7], [8]).

B. Time Enhanced A*

The Time Enhanced A* (TEA*) is an extension of the A*, used when there are multiple vehicles. It contains an additional component – time. This component allows a better prediction of the vehicles' movements during the run time [9].

TEA* consists of an incremental algorithm that builds the path of each vehicle considering the movements of other mobile robots. This feature allows the algorithm to produce conflict free routes and, at the same time, deal with deadlock situations, since the paths are constantly recalculated and the map information is updated at each iteration. This way, the unpredictable events are considered in the input map, allowing to avoid the main challenges of any multi-robot approach, such as collisions and deadlocks [9]. Each node on the map has three dimensions: the Cartesian coordinates (x, y) and a representation of the discrete time. The time is represented through temporal layers, $k = [0; T_{Max}]$, on which T_{Max} represents the maximum number of layers. Each temporal graph is composed of a set of free and occupied/obstacles nodes, as can be can see in Fig. 1 [9].

The path for each robot is calculated during the temporal layers. In each temporal layer, the position of each vehicle is known and shared with the other vehicles. This way, it is possible to detect possible future collisions and avoid them at the beginning of the paths' calculation. Each robot can only start and stop in nodes and a node can only be occupied by one vehicle on each temporal layer [9].

The operation of the TEA* algorithm is similar to A*, since for each AGV, during the path calculation, the next neighbor node analyzed depends on a cost function, $f(n)$, given by the sum of two terms: the distance to the initial vertex, $g(n)$, and the distance to the end point, $h(n)$. The main difference between the two algorithms is that time is considered in TEA*, resulting in two definitions, according to [9]:

- "Definition 1: The neighbor vertices of a vertex j in the temporal layer k belong to the next temporal layer given by $k + 1$" (Fig. 1), that is, the total number of temporal layers depends of the number of iterations required to reach the intended destination. The more complex the map, the more iterations are needed.
- "Definition 2: The neighbor vertices of vertex j (v^j_{adj}) include the vertex containing the AGV current position, and all adjacent vertices in the next time component.", that is, the set of neighboring nodes includes not only the adjacent nodes, but also the node corresponding to the position in analysis. This property allows a vehicle to maintain its position between consecutive time instants if no neighbour node is free. In this case, it is not considered a zero value for the euclidean distance; instead a constant heuristic value corresponding to the stopped movement is assigned.

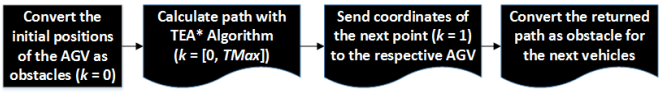


Fig. 2. Control diagram for each algorithm iteration in a multi-AGV situation [9].

Fig. 2 depicts the control diagram of the TEA* operation, for each iteration of the algorithm, in a multi-AGV context. The initial positions of the AGV are placed as obstacles, in the first time layer ($k = 0$), allowing a vehicle to consider the positions of the other vehicles as nodes occupied. In order to avoid deadlocks, those nodes are placed as obstacles only in $k = 0, 1$, that is, in the first two time layers. Next, is analyzed what the AGV has to do (missions) and the path for each of the vehicles is calculated using the TEA*. The coordinates of the next node, in the second time layer ($k = 1$), are transmitted to the respective AGV. Before moving to the next mission, the full path is converted as obstacle to the following missions and respective AGV. With this in mind, it is possible to coordinate the vehicles, while avoiding collisions [9].

This algorithm is yet little explored. The only advantage reported is that it considers obstacles in order to avoid collisions, unlike A*. An example of its application is in AGV fleet management [9].

C. Rapidly exploring Random Tree

The Rapidly exploring Random Tree (RRT) algorithm incrementally constructs a search tree in the configuration space until the goal configuration can be connected to one of its nodes. The operation of the RRT, exemplified in Fig. 3, involves the iterative execution of the following steps [10]:

1) A random configuration, q_{rand}, is sampled in the configuration space.
2) The tree is searched for a configuration q_{near}, which is the nearest node in the tree to q_{rand}.
3) A new configuration q_{new} is created by moving a predefined distance d from q_{near} in the direction of q_{rand}.
4) If q_{new} is a valid configuration that falls in C_{free} (unobstructed space), and if the local path between it

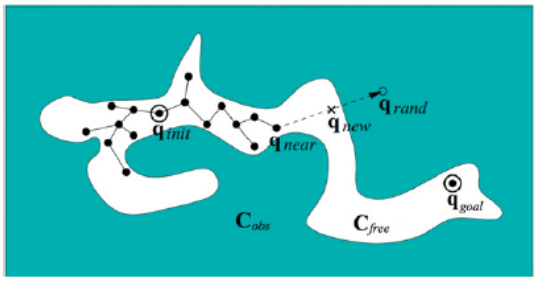


Fig. 3. Example of the RRT operation at an intermediate stage during its construction [10].

and q_{near} is collision-free, then q_{new} is added to the tree as a new node and an edge is created between q_{new} and q_{near}. However, if q_{new} falls in C_{obs} (obstacle space), and if the local path between it and q_{near} has collisions, then is not created an edge between q_{new} and q_{near}.

These steps are repeated until the goal configuration can be connected to the tree or is reached a maximum number of iterations, or a given number of nodes in the tree, or a given running time [10], [11]. That is, the goal is to execute the whole process from q_{init} (starting point) to q_{goal} (end point).

The most common metric for the nearest-neighbor selection is the Euclidean distance between points. In this case, the expansion pattern of the tree is modeled by the Voronoi diagram over the nodes within the tree. The probability of a node being expanded is directly proportional to the volume of its corresponding Voronoi region. Nodes that have a larger Voronoi region (*i.e.* the portion of the space that is closer to the node than to other nodes of the tree) are more likely to be chosen for expansion and are referred to as major nodes. This way, the tree is pulled towards unexplored areas, spreading rapidly in the configuration space (as the Voronoi regions of samples become approximately equal in size, the exploratory behavior gradually shifts from expansion of the tree to refinement). In the case of the Euclidean metric, these nodes tend to lie on the outside of the tree during the initial exploration. Conversely, inner or minor nodes have smaller Voronoi regions and often lie on the inside of the tree. Once the tree has explored the state space, these become major nodes as the algorithm begins to fill in the space. This phenomenon of favoring some nodes over others is referred to as the Voronoi bias, and yields an initial preference towards the exploration of the state space [12], [13].

Summing up, the efficiency of RRT stems from the Voronoi bias property which promotes tree growth towards unexplored regions. Therefore, the key is the determination of the distance metric which computes the nearest-neighbour in the RRT algorithm [14]. In holonomic planning, the Euclidean distance is an ideal metric to generate a Voronoi bias because any node which is the closest from the sampled points can be expanded. If there exists differential constraints, however, which limit the evolution of the system states, the Euclidean distance measure fails to capture the true distance. Fig. 4 exhibits this problem. A state Xr is drawn randomly in Fig. 4a. Here, Xs is an initial state of the system and $X1...X8$ are existing nodes in the current tree. If Euclidean distance is used for the distance metric, $X2$ is chosen as the closest node from Xr as shown in Fig. 4b. However, this is not true for the system which has differential constraints. Instead, $X4$ is the closest node from Xr if the nonholonomic constraints are considered as in Fig. 4c. From this example, it can be seen that the true distance metric is extremely important to the RRT planner under differential constraints [14].

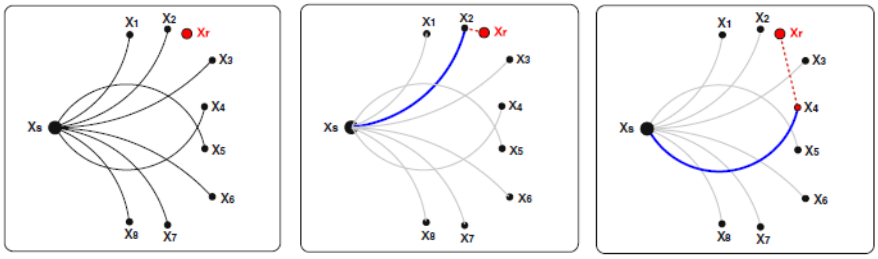
The relative advantages of this algorithm are that it: (*i*) is successful at solving path-planning problems in high-dimensional spaces [15]; (*ii*) can be implemented for real-time, online planning [16]; and (*iii*) avoids wandering around in explored regions [17]. The disadvantages are that: (*i*) it is not appropriate when road planning involves narrow passages [18]; and (*ii*) the solution obtained is sub-optimal, since the planning process is merely a random exploration of the space [19].

The RRT algorithm has already been applied in various areas, such as, molecular biology ([10], [11], [17], [20]), automation / robotics ([12], [21]) including path planning for mobile robots ([14], [22]–[42], medicine (for steerable needles in 3D environments with obstacles [43]) and human-system interaction [44]. The practical applications just referred are based on the RRT algorithm; however, they do not use their standard/basic version, but variants of this or the conjugation with other algorithms, in order to overcome their disadvantages and/or to be able to acquire characteristics that fit to each problem under consideration.

D. Time Windows

In dynamic routing a calculated path depends on the number of currently active AGV missions and their priorities. The Time Windows (TW) method allows to determine the shortest path using time windows (Fig. 5). This method checks the mission candidate paths by using the time windows to verify if certain paths are feasible. Viability of a particular path is evaluated by a time windows insertion followed by a time windows overlap (conflict) test. In the case of conflict, the algorithm iteratively reinserts time windows until conflicts disappear or an overlap is present only on the paths origin arc, indicating that the candidate path is not feasible. The procedure is repeated for all candidate paths and the result is a set of executable paths. The final task of the algorithm is to choose the shortest one among executable paths in terms of a time required for a vehicle in mission to get from the origin to the destination arc. When a new mission is requested at a given time, is searched a idle vehicle to assign it to that mission (with an initial mission priority). As a goal of dynamic routing is to determine the shortest path for certain mission under the current state of the system, all candidate paths should be checked for feasibility [45].

The applications areas for Time Windows encompass AGV (for dynamic routing in multi-AGV systems [45]), logistics (in vehicle routing problem [46]–[58], pickup and delivery problems [59], [60], Home Health Care (HHC) [61], [62] and petrol station replenishment problems [63]).



(a) (b) (c)

Fig. 4. A state Xr is drawn randomly where Xs is an initial state of the system and $X1...X8$ are existing nodes in the current tree (a), the Euclidean distance is a ideal metric to generate Voronoi bias in holonomic planning but it does not incorporate differential constraints of the system (b) and in nonholonomic planning, a new distance metric which incorporates limitations of the system is needed to compute the real distance (c) [14].

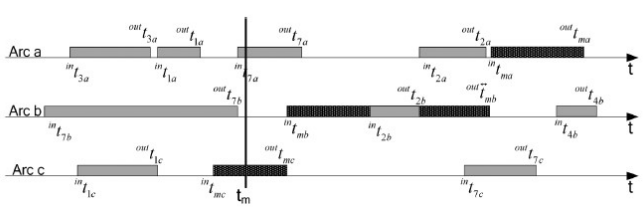


Fig. 5. Example of time windows in a vector form [45].

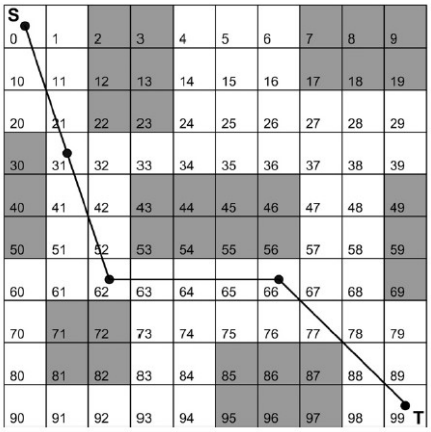


Fig. 6. Example of the orderly numbered grid environment representation [64].

According to what was described above, it is possible to verify that the Time Windows algorithm is applied in various vehicle routing problems in order to be able to optimize the solution to those problems.

E. Genetic Algorithms

Genetic Algorithms (GA) are a parallel and global search technique that emulates natural genetic operators. As it simultaneously evaluates many points in the parameter space, it is more likely to converge to the global optimal [64].

Many path planning methods use a grid-based model to represent the environment space, leading to two possible representations: (*i*) through an orderly numbered grid, as shown in Figure 6, or (*ii*) through the (x,y) coordinates plane [64].

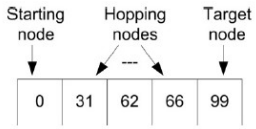


Fig. 7. Decimal coded genes of a chromosome [64].

A chromosome represents a candidate solution for the path planning problem. A chromosome representing a path encodes a starting node, a target node and the nodes through which the mobile robot travels. These nodes, or steps, in the path are called genes of the chromosome. A valid path consists of a sequence of grid labels which begins at the starting node and ends at the target node, as shown in Figure 7 [64].

The initial population is generally generated randomly. Thus, some of the generated chromosomes may include infeasible paths intersecting an obstacle. An optimal, or near optimal, solution can be found by genetic operators, even though the initial population includes infeasible paths. However, this reduces the search capability of the algorithm and increases the time to find the solution. Additionally, crossover of two infeasible chromosomes may generate new infeasible paths. To solve this problem, each chromosome must be checked whether it intersects an obstacle, when generating the initial population. If it does, the intersected gene on the chromosome is changed randomly, until a feasible one is found [64].

The optimal path may be the shortest one, or the one requiring the least time or less energy to be traversed. Generally, in path planning problems, the objective function is considered as the shortest path. In [64], the objective function value of a chromosome used in the GA is given by Equations 4 and 5:

$$f = \begin{cases} \sum_{i=1}^{n-1} d(p_i, p_{i+1}), & \text{for feasible paths} \\ \sum_{i=1}^{n-1} d(p_i, p_{i+1}) + penalty, & \text{for infeasible paths} \end{cases} \quad (4)$$

$$d(p_i, p_{i+1}) = \sqrt{(x_{(i+1)} - x_i)^2 + (y_{(i+1)} - y_i)^2} \quad (5)$$

being, f the fitness function, p_i the ith gene of the chromosome, n the length of the chromosome, d the distance between

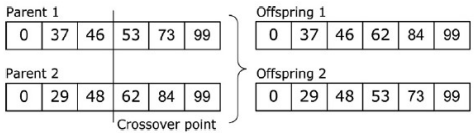


Fig. 8. Single-point crossover [64].

two nodes, x_i and y_i the robot current position, and $x_{(i+1)}$ and $y_{(i+1)}$ the robot next position. The direction of the robot path is given by equation 6 [64]:

$$\alpha = \tan^{-1} \frac{(y_{(i+1)} - y_i)}{(x_{(i+1)} - x_i)} \tag{6}$$

The objective function value is defined as the sum of distances between each node in a path. If there is an obstacle in the robot path, a penalty is added to the objective function value. The penalty value should be greater than the maximum path length on the environment. In order to find an optimal path, the algorithm searches for the chromosome with the least value for the objective function [64].

The main principle of the GA is that the best genes on the chromosomes should survive and be transferred to new generations. A selection procedure needs to be done to determine the best chromosomes. This process consists in the following three steps [64]:

1) Objective function values of all chromosomes are computed.
2) Fitness values are assigned to chromosomes according to their objective function values. In [64], the rank based fitness assignment is used instead of the proportional assignment method. This prevents a few better chromosomes to be dominant in the population.
3) Chromosomes are selected according to their fitness values and then placed into a mating pool to produce new chromosomes.

Normally, crossover combines the features of two parent chromosomes to form two offsprings. In Fig. 8, single-point crossover operator is illustrated, and the genes of the two "parent" chromosomes after the crossover point are changed [64].

All candidate chromosomes in the population are subjected to the random mutation after the crossover operation. This is a random bit-wise binary complement operation or a random small change in a gene, depending on the coding of chromosomes, applied uniformly to all genes of all individuals in the population, with a probability equal to the mutation rate. The mutation operation increases the diversity of the population and avoids the premature convergence. It expands the search space to regions that may not be close to the current population, thus ensuring a global search [64].

In conventional GA, random mutation is the most commonly used operator. However, random mutation can cause generation of infeasible paths. Even though a chromosome is feasible before the mutation operation, the new node changed by the mutation may have an obstacle and therefore it constitutes an infeasible path (see Fig. 9). This makes the optimization slower and increases the number of generations [64].

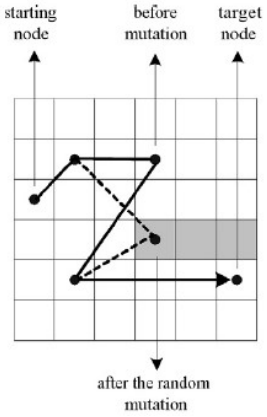


Fig. 9. Infeasible path [64].

To overcome this problem, several studies concerned with the improvement of mutation operation have been done in the literature. The authors of those studies, as well as the method proposed by each author, are described in [64].

GA have been applied in mobile robots ([64]–[70]), timetabling problems ([71]–[73]), sensor networks ([74]), building trade systems ([75]), logistics ([76], [77]), automobile industry ([78]) and cloud computing ([79]–[83]). Among the mentioned references, what is often referred to is that to increase the speed of convergence of GA and to apply them with modifications that help the case study in question.

III. Discussion

Table I summarizes the application areas of each algorithm considered in this contribution. All studied algorithms are applied to mobile robots path planning, and there are other areas of application common to them. It is also seen that GA are used in more areas, followed by A* and RRT, and finally the TW and TEA*.

According to this survey, it was noticed that the most used algorithms for mobile robots path planning are A*, RRT, TW and AG. On the other hand, the least studied one is the TEA*, since only one case study was found in the literature ([9]).

Regarding their relative advantages and disadvantages, the TEA* presents advantages in relation to the A*, since it allows the planning of routes in multiple mobile robots systems. In the remaining algorithms it is difficult to ascertain their relative advantages and disadvantages, since each case study presents specific and different characteristics; however, all are adequate to solve problems of path planning in mobile robots.

According to what has been presented, it is possible to realize that the standard algorithms are often used. Typically, they are used after some improvement or in conjunction with another algorithm, according to the needs of each case study. For instance, throughout the research on GA were found cases that reconciled this type of algorithm with Time Windows ([49], [52], [54], [84]–[89]) and A* ([90]).

TABLE I
APPLICATION AREAS OF THE ALGORITHMS

Algorithms	Mobile Robots	Medicine	Games	Molecular biology	Timetabling problem	Logistics	Cloud Computing
A*	X	X	X				
TEA*	X						
RRT	X	X		X			
TW	X					X	
GA	X				X	X	X

IV. CONCLUSIONS AND FUTURE WORK

This paper presented some algorithms used in the path planning of mobile robots. For each one, its operation was briefly described, some of its advantages and disadvantages were analyzed, and were presented some of its possible application areas. It was possible to conclude that there are several modifications that can be made to make each of these algorithms more efficient, and they can be associated with other algorithms to solve particular problems, as exemplified in several of the referred case studies. Next is planned to implement some of these algorithms in AGV fleets, to find out their relative advantages and disadvantages and in which situations their implementation is more adequate.

REFERENCES

[1] D. Floreano, P. Husbands, and S. Nolfi, *Springer Handbook of Robotics*, 01 2008.

[2] S. M LaValle, *Planning Algorithms*, 01 2006.

[3] N. Rivera, J. A. Baier, and C. Hernandez, "Weighted real-time heuristic search," in *Proc. of the 2013 Int. Conf. on Autonomous agents and multi-agent systems (AAMAS '13)*, Richland, SC, 2013, pp. 579–586.

[4] F. Ducho, A. Babinec, M. Kajan, P. Beo, M. Florek, T. Fico, and L. Juriica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, 12 2014.

[5] Z. Zhang and Z. Zhao, "A multiple mobile robots path planning algorithm based on a-star and dijkstra algorithm," *Int. Journal of Smart Home*, vol. 8, pp. 75–86, 05 2014.

[6] A. Mohammadi, M. Rahimi, and A. Suratgar, "A new path planning and obstacle avoidance algorithm in dynamic environment," 05 2014, pp. 1301–1306.

[7] G. Elizebeth Mathew, "Direction based heuristic for pathfinding in video games," *Procedia Computer Science*, vol. 47, pp. 262–271, 12 2015.

[8] E. Risky Firmansyah, S. Ummi Masruroh, and F. Fahrianto, "Comparative analysis of a* and basic theta* algorithm in android-based pathfinding games," 11 2016, pp. 275–280.

[9] J. Santos, P. Costa, L. Rocha, K. Vivaldini, A. Moreira, and G. Veiga, "Validation of a time based routing algorithm using a realistic automatic warehouse scenario," vol. 418, 11 2016.

[10] I. Al-Bluwi, T. Simon, and J. Corts, "Motion planning algorithms for molecular simulations: A survey," *Computer Science Review*, vol. 6, p. 125143, 07 2012.

[11] D. Devaurs, M. Vaisset, T. Simon, and J. Corts, "A multi-tree approach to compute transition paths on energy landscapes," 07 2013.

[12] A. Shkolnik, M. R. Walter, and R. Louis Tedrake, "Reachability-guided sampling for planning under differential constraints," 06 2009, pp. 2859 – 2865.

[13] O. B. M. Rickert, A. Sieverling, "Balancing exploration and exploitation in sampling-based motion planning," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1305–1317, Dec 2014.

[14] K. Yang, S. Moon, S. Yoo, J. Kang, N. Lett Doh, H. Kim, and S. Joo, "Spline-based rrt path planner for non-holonomic robots," *Journal of Intelligent & Robotic Systems*, vol. 73, 01 2014.

[15] D. Devaurs, T. Simon, and J. Corts, *Efficient sampling-based approaches to optimal path planning in complex cost spaces*, 04 2015, pp. 143–159.

[16] G. Kewlani, G. Ishigami, and K. Iagnemma, "Stochastic mobility-based path planning in uncertain environments," 11 2009, pp. 1183 – 1189.

[17] L. Jaillet, F. Corcho, J. Perez, and J. Corts, "Randomized tree construction algorithm to explore energy landscapes," *Journal of computational chemistry*, vol. 32, pp. 3464–74, 12 2011.

[18] J. Denny, E. Greco, S. Thomas, and N. Amato, "Marrt: Medial axis biased rapidly-exploring random trees," 09 2014, pp. 90–97.

[19] S. S. Seng Keat Gan, Kwang Jin Yang, "3d online path planning in a continuous gaussian process occupancy map."

[20] J. Corts, D. Thanh Le, R. Iehl, and T. Simon, "Simulating ligand-induced conformational changes in proteins using a mechanical disassembly method," *Physical chemistry chemical physics : PCCP*, vol. 12, pp. 8268–76, 08 2010.

[21] G. Kewlani, G. Ishigami, and K. Iagnemma, "Stochastic mobility-based path planning in uncertain environments," 11 2009, pp. 1183 – 1189.

[22] G. Aoude, B. Douglas Luders, J. How, and T. Pilutti, "Sampling-based threat assessment algorithms for intersection collisions involving errant drivers," vol. 7, 11 2010.

[23] L. Han, H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on rrt," 05 2011, pp. 5622–5627.

[24] L. Jaillet, J. Hoffman, J. van den Berg, P. Abbeel, J. Porta, and K. Goldberg, "Eg-rrt: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles," 09 2011, pp. 2646–2652.

[25] K. Yang, "An efficient spline-based rrt path planner for non-holonomic robots in cluttered environments," 05 2013, pp. 288–297.

[26] R. Kala and K. Warwick, "Planning of multiple autonomous vehicles using rrt," *Proc. of the 10th IEEE Int. Conf. on Cybernetic Intelligent Systems, CIS 2011*, 09 2011.

[27] M. Kothari, I. Postlethwaite, and D.-W. Gu, "Multi-uav path planning in obstacle rich environments using RRT," 12 2009.

[28] A. A. Neto, D. G. Macharet, and M. F. M. Campos, "Feasible rrt-based path planning using seventh order bzier curves," in *2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct 2010, pp. 1445–1450.

[29] E. K. Yiqun Dong, Changhong Fu, "Rrt-based 3d path planning for formation landing of quadrotor uavs," 11 2016.

[30] A. Neto, D. Macharet, and M. Campos, "On the generation of trajectories for multiple uavs in environments with obstacles," *Journal of Intelligent and Robotic Systems*, vol. 57, pp. 123–141, 04 2011.

[31] K. Yang, S. K. Gan, and S. Sukkarieh, "An efficient path planning and control algorithm for ruavs in unknown and cluttered environments," *J. of Intelligent and Robotic Systems*, vol. 57, pp. 101–122, 01 2010.

[32] E. Koyuncu, N. Ure, and G. Inalhan, "Integration of path/maneuver planning in complex environments for agile maneuvering ucavs," *Journal of Intelligent and Robotic Systems*, vol. 57, pp. 143–170, 09 2010.

[33] C. D. C. J. Yoon, "Path planning for unmanned ground vehicle in urban parking area," in *2011 11th Int. Conf. on Control, Automation and Systems*, Oct 2011, pp. 887–892.

[34] Y. Dong, E. Camci, and E. Kayacan, "Faster rrt-based nonholonomic path planning in 2d building environments using skeleton-constrained path biasing," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 89, pp. 1–15, 05 2017.

[35] Y. Dong, Y. Zhang, and J. Ai, "Experimental test of unmanned ground vehicle delivering goods using rrt path planning algorithm," *Unmanned Systems*, vol. 05, pp. 45–57, 01 2017.

[36] D. Vaz, R. Inoue, and V. G. Jr, "Kinodynamic motion planning of a skid-steering mobile robot using rrts," 11 2010, pp. 73 – 78.

[37] V. Vonasek, M. Saska, K. Konar, and L. Preucil, "Global motion planning for modular robots with local motion primitives," 05 2013.

[38] N. Vahrenkamp, A. Barski, T. Asfour, and R. Dillmann, "Planning and execution of grasping motions on a humanoid robot," *9th IEEE-RAS Int. Conf. on Humanoid Robots, HUMANOIDS09*, 12 2009.

[39] S. Dalibard, A. El Khoury, F. Lamiraux, A. Nakhaei, M. Tax, and J.-P. Laumond, "Dynamic walking and whole-body motion planning for

humanoid robots: an integrated approach," *The Int. Journal of Robotics Research*, vol. 32, pp. 1089–1103, 08 2013.
[40] A. Tomas Perez, S. Karaman, A. Shkolnik, E. Frazzoli, S. J. Teller, and M. R. Walter, "Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms," 09 2011, pp. 4307–4313.
[41] A. Yershova and S. M. LaValle, "Motion planning for highly constrained spaces," *Lecture Notes in Control and Information Sciences*, vol. 396, 08 2009.
[42] J. John Bialkowski, S. Karaman, and E. Frazzoli, "Massively parallelizing the rrt and the rrt*," 09 2011, pp. 3513–3518.
[43] S. Patil and R. Alterovitz, "Interactive motion planning for steerable needles in 3d environments with obstacles," 10 2010, pp. 893 – 899.
[44] M. Taix, D. Flavigne, and E. Ferre, "Human interaction with motion planning algorithm," *J. of Int. & Robotic Systems*, vol. 67, 09 2012.
[45] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and T. Petrovic, "Time windows based dynamic routing in multi-agv systems," *IEEE Trans. on Automation Science and Eng.*, vol. 7, pp. 151 – 155, 02 2010.
[46] R. Skinderowicz, "Co-operative, parallel simulated annealing for the vrptw," 09 2011, pp. 495–504.
[47] R. Tavakkoli-Moghaddam, M. Gazanfari, M. Alinaghian, A. Salamat-bakhsh, and N. Norouzi, "A new mathematical model for a competitive vehicle routing problem with time windows solved by simulated annealing," *Journal of Manufacturing Systems*, vol. 30, pp. 83–92, 04 2011.
[48] Y. Nagata, O. Brysy, and W. Dullaert, "A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows," *Computers & OR*, vol. 37, pp. 724–737, 04 2010.
[49] C. Santilln, L. Cruz Reyes, J. J. Gonzlez Barbosa, H. Fraire-Huacuja, R. A. Pazos R., and J. J. Martnez P., "Ant colony system with characterization-based heuristics for a bottled-products distribution logistics system," *Journal of Computational and Applied Mathematics*, vol. 259, pp. 965–977, 03 2014.
[50] A. Kok, E. Hans, M. Schutten, and W. Zijm, "Vehicle routing with traffic congestion and drivers' driving and working rules," 01 2010.
[51] E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau, "A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows," *Networks*, vol. 54, pp. 190–204, 12 2009.
[52] I. Moon, J.-h. Lee, and J. Seong, "Vehicle routing problem with time windows considering overtime and outsourcing vehicles," *Expert Systems with Applications*, vol. 39, p. 1320213213, 12 2012.
[53] H.-K. Chen, C.-F. Hsueh, and M.-S. Chang, "Production scheduling and vehicle routing with time windows for perishable food products," *Computers & Operations Research*, vol. 36, pp. 2311–2319, 07 2009.
[54] S. R. Balseiro, I. Loiseau, and J. Ramonet, "An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows," *Computers & OR*, vol. 38, pp. 954–966, 06 2011.
[55] L. Hong, "An improved lns algorithm for real-time vehicle routing problem with time windows," *Computers & OR*, vol. 39, pp. 151–163, 02 2012.
[56] J.-Q. Li, P. Mirchandani, and D. Borenstein, "Real-time vehicle rerouting problems with time windows," *European Journal of Operational Research*, vol. 194, pp. 711–727, 05 2009.
[57] S. Pirkwieser and G. Raidl, "Multiple variable neighborhood search enriched with ilp techniques for the periodic vehicle routing problem with time windows," 10 2009, pp. 45–59.
[58] ——, "A column generation approach for the periodic vehicle routing problem with time windows1," 05 2012.
[59] Y. Nagata and S. Kobayashi, "A memetic algorithm for the pickup and delivery problem with time windows using selective route exchange crossover," vol. 6238, 09 2010, pp. 536–545.
[60] ——, "Guided ejection search for the pickup and delivery problem with time windows," 04 2010, pp. 202–213.
[61] R. liu, X. Xie, and T. Garaix, "Hybridization of tabu search with feasible and infeasible local searches for periodic home health care logistics," *Omega*, vol. 47, 09 2014.
[62] Y. Kergosien, C. Lent, and J.-C. Billaut, "Home health care problem an extended multiple traveling salesman problem," 08 2009.
[63] F. Cornillier, G. Laporte, F. Boctor, and J. Renaud, "The petrol station replenishment problem with time windows," *Computers & Operations Research*, vol. 36, pp. 919–935, 03 2009.
[64] L. Changan, Y. Xiaohu, L. Chunyang, and L. Guodong, "Dynamic path planning for mobile robot based on improved genetic algorithm," *Chinese Journal of Electronics*, vol. 19, 05 2010.
[65] J. C. Mohanta, D. R. Parhi, and S. K. Patel, "Path planning strategy for autonomous mobile robot navigation using petri-ga optimisation," *Computers & Electrical Engineering*, vol. 37, no. 6, pp. 1058 – 1070, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790611001091
[66] M. Tahan and M. Manzuri, "Efficient and safe path planning for a mobile robot using genetic algorithm," 05 2009, pp. 2091–2097.
[67] Z. Yao and L. Ma, "A static environment-based path planning method by using genetic algorithm," *Computing, Control and Industrial Engineering, International Conf. on*, vol. 2, pp. 405–407, 06 2010.
[68] S.-Y. Fu, L. Han, Y. Tian, and G.-S. Yang, "Path planning for unmanned aerial vehicle based on genetic algorithm," 08 2012, pp. 140–144.
[69] T. Shima and C. Schumacher, "Assigning cooperating uavs to simultaneous tasks on consecutive targets using genetic algorithms," *JORS*, vol. 60, pp. 973–982, 07 2009.
[70] Y. Liang and L. Xu, "Global path planning for mobile robot based genetic algorithm and modified simulated annealing algorithm," 01 2009, pp. 303–308.
[71] N. Pillay and W. Banzhaf, "An informed genetic algorithm for the examination timetabling problem," *Appl. Soft Comput.*, vol. 10, pp. 457–467, 03 2010.
[72] R. Raghavjee and N. Pillay, "Evolving solutions to the school timetabling problem," 01 2010, pp. 1524 – 1527.
[73] ——, "An informed genetic algorithm for the high school timetabling problem," 01 2010, pp. 408–412.
[74] A. Bari, S. Wazed, J. Arunita, and S. Bandyopadhyay, "A genetic algorithm based approach for energy efficient routing in two-tiered sensor networks," *Ad Hoc Networks*, vol. 7, pp. 665–676, 06 2009.
[75] H. Jiang and L. Kang, "Building trade system by genetic algorithm," vol. 5821, 10 2009, pp. 18–23.
[76] L. Shi, H. Fan, P. Gao, and H. Zhang, "Network model and optimization of medical waste reverse logistics by improved genetic algorithm," 09 2009, pp. 40–52.
[77] T. Rakkiannan and P. P Balasubramanie, "A genetic algorithm with a tabu search (gta) for traveling salesman problem," *SHORT PAPER International Journal of Recent Trends in Engineering*, vol. 1, 04 2009.
[78] B. Ju, L. Xi, and X. Zhou, "New product design based target cost control with bp neural network and genetic algorithm - a case study in chinese automobile industry," vol. 5821, 10 2009, pp. 111–122.
[79] S. Ho Jang, T. Kim, J.-K. Kim, and J. Sik Lee, "The study of genetic algorithm-based task scheduling for cloud computing," *International Journal of Control and Automation*, vol. 5, pp. 157–162, 12 2012.
[80] Y. Kessaci, N. Melab, and E.-G. Talbi, "A pareto-based ga for scheduling hpc applications on distributed cloud infrastructures," 08 2011, pp. 456 – 462.
[81] K. Zhu, H. Song, L. Liu, J. Gao, and G. Cheng, "Hybrid genetic algorithm for cloud computing applications," 12 2011, pp. 182–187.
[82] E. Mocanu, M. Florea, M. Ionut Andreica, and N. Tapus, "Cloud computingtask scheduling based on genetic algorithms," 03 2012, pp. 1–6.
[83] A. Verma and P. Kumar, "Scheduling using improved genetic algorithm in cloud computing for independent tasks," 08 2012.
[84] K. Ghoseiri and S. Ghannadpour, "A hybrid genetic algorithm for multi-depot homogenous locomotive assignment with time windows," *Applied Soft Computing*, vol. 10, pp. 53–65, 01 2010.
[85] ——, "Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm," *Applied Soft Computing*, vol. 10, pp. 1096–1107, 09 2010.
[86] Z. Ursani, D. Essam, D. Cornforth, and R. Stocker, "Localized genetic algorithm for vehicle routing problem with time windows," *Appl. Soft Comput.*, vol. 11, pp. 5375–5390, 12 2011.
[87] S. Ghannadpour, S. Noori, R. Tavakkoli-Moghaddam, and K. Ghoseiri, "A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application," *Applied Soft Computing*, vol. 14, pp. 504–527, 01 2014.
[88] L. Cruz Reyes, J. J. Gonzlez Barbosa, J. F. Orta, B. A. Arraaga C, and H. Fraire-Huacuja, "A new approach to improve the ant colony system performance: Learning levels," 06 2009, pp. 670–677.
[89] Y. Zhu and T. Zhen, "Hybrid ant colony algorithm based on vehicle routing problem with time windows," in *2009 WASE Int. Conf. on Information Engineering*, vol. 2, July 2009, pp. 50–53.
[90] C. Liu and A. Kroll, "A centralized multi-robot task allocation for industrial plant inspection by using a* and genetic algorithms," 04 2012, pp. 466–474.