Coordination in Navigation of Multiple Mobile Robots

Rahul Kala
School of Cybernetics, School of Systems Engineering,
University of Reading, Whiteknights, Reading, RG66AY, United Kingdom
rkala001@gmail.com, http://rkala.99k.org/

Citation: R. Kala (2014) Coordination in Navigation of Multiple Mobile Robots. Cybernetics and Systems, 45(1): 1-24.

Final Version Available At: http://www.tandfonline.com/doi/abs/10.1080/01969722.2014.862085

Abstract: Coordination in the problem of multi-robot path planning enables the individual robots to escape collision when planned in a decentralized manner. A good coordination strategy should not be time consuming; should be probabilistically complete; and should not over-prefer a robot. In this paper an altering local search based strategy is studied. Experiments reveal the ability of the algorithm to place the different robots judiciously for near-optimal collision avoidance. The algorithm takes lesser time as compared to the centralized approaches, is probabilistically complete as opposed to the reactive approaches, and treats the robots alike as opposed to the prioritized approaches.

Keywords: *Multi-robot path planning, multi-robot coordination, evolutionary algorithms, local search, collision avoidance.*

1. Introduction

Path planning of a mobile robot deals with the problem of devising a collision-free trajectory for the motion of the robot (Latombe, 1991; Tiwari et al., 2013). Increasing robotic applications have given rise to the notion of multi-robotics (Arai et al., 2002; Parker et al., 2005), where a number of robots are employed for any task. In the planning domain this corresponds to the problem of multi-robot motion planning, where a number of robots need to reach their pre-specified goal starting from their pre-specified source without collision with any other robot or static obstacle. The problem of planning of multiple robots may be studied under the head of centralized methods and decentralized methods (Lumelsky and Hariharan, 1997; Sánchez-Ante and Latombe, 2002). The centralized methods construct a joint configuration space accounting for all the robots in the static environment. These methods are optimal and complete but highly computationally expensive. The decentralized approaches plan each robot separately. The possible collisions are rectified using a coordination strategy. The approach is computationally inexpensive but not complete or optimal.

The task of planning of a single robot attempts to generate a trajectory such that the time to reach the goal is shortest, while the robot maintains a large curvature and a large clearance from the obstacles nearby. Some of these objectives may be contradictory and the planned path attempts to tradeoff between these objectives. Genetic Algorithms are widely used choice of algorithm due to the advantages of being probabilistically complete, near optimal, iterative in nature, and guarantee of validity of nonholonomic constraints. However the limitations are large computational time (lesser than that of graph search based methods) especially in cases when optimal paths of robots have a large number of turns. The optimality is larger for the probabilistic roadmap and rapidly exploring random trees counterparts, while the computational time is also larger in general.

An ideal coordination scheme would be the one which produces an optimal travel plan of the multiple robots within small execution times. An optimal travel plan for any scenario may be visualized by the manner in which humans walk and drive with each individual reaching its destination keeping all the objectives met as far as possible. In some scenarios however this may make a robot travel by a different route altogether, like humans sensing a prospective congestion sometimes take an alternative way to the destination. The ideal scheme may be produced by a centralized planning technique, which may however be time consuming.

1.1 Coordination Schemes

Some broad approaches of coordination are discussed. One of the easy ways to solve the problem is to embed all the travel possibilities of different robots into a centralized problem which may be optimized by an evolutionary technique. Robots whose optimal paths are far from collision do not necessarily affect each other. Hence from an optimization perspective this problem is towards the class of separable problems which are usually easier to solve. Co-evolution (Potter and de Jong, 1998, 2000) may be a good choice of algorithm where the different robots may be made different modules which cooperate during the evolution. The approach is probabilistically complete and near-optimal. Co-evolution does a good job by working over the same complex joint coordination space of the multiple robots, but using a pre-known heuristic of individual robots being prone to act as separable optimization variables. However it is still difficult to simulate and search the optimal travel plan as the optimal combination of a large number of competing travel plans needs to be searched, which is computationally expensive. Similar problems exist with the Probabilistic Roadmap (Kavraki et al., 1998; Sánchez-Ante and Latombe, 2002) method where a high dimensional space of single robot is sampled to a low dimensional space, but the joint configuration space may yet be unworkable within small execution times. Especially for multiple robots, these techniques are not complete in cases where sufficient open spaces are not available in the map and the obstacles may make the way rather congested.

The other option is to use reactive planners which treat the different robots as dynamic obstacles. These include artificial potential field (Baxter et al., 2009; Khatib, 1985; Pradhan et al., 2006) and fuzzy approaches (Kala et al., 2010a; Selekwa et al., 2008). These planners operate in a real time mode. They usually employ a deliberative planning technique at a higher level to induce completeness. However even for a single robot case, characteristic obstacle placement can make the robot get trapped at places from where it is impossible to come out, thus they are still not really complete. Robots can play a much mysterious role as obstacles, highly increasing the possibility of one robot getting another one trapped. Besides due to the dynamic nature, different robots may not place themselves in the best of ways to avoid each other, thus forcing others to slow down largely which is a loss of optimality.

In prioritized planners (Bennewitz et al., 2001, 2002) each robot is assigned a priority and the lower priority robots only consider the higher priority robots and attempt to avoid collisions. The priorities may be predefined, or they may be optimized using some scheme. These approaches ensure that results are generated in small execution times as compared to the centralized counterparts. The problem with the approach is however that whichever robot has a higher priority; it would always travel by its optimal plan. The other robots may have to deviate by a large amount to avoid collision and maintain sufficient clearance. If there is some obstacle on both sides, lower priority robot may fail to find a way. Ideal travel plan would however be the robots mutually aligning themselves to leave enough space for the other robot to fit in.

1.2 Related Works

A lot of work is done to solve the problem of planning of single and multiple robots using Genetic Algorithms (GA). Kala et al. (Kala et al., 2011) proposed a coarser to finer strategy to formulate the path of a single robot. The initial little generations coarsely computed the fitness of the robot path, while the higher generations employed a finer fitness computation. All clearances were initially computed. This paper proposes a better mechanism of generating initial individuals, computing clearances, besides devising other operators for path generation. In a similar work Kala et al. (2010b) used GA for navigating a single robot in a dynamic environment. One instance of GA optimized the coarser path, while another was used for the optimization of the finer path which worked in a sub-map. The motion of the dynamic obstacles was extrapolated to be moving with same speed. The notion of optimization by extrapolation clearly results in sub-optimal paths. The authors did not talk about scenarios where a robot gets trapped. Further the GA has to be working constantly as the robot moves due to the changing environment. In an environment where the robots are the only dynamic elements, a stronger coordination strategy (as is the aim of this work) is desirable. Chakraborty et al. (2008) used Differential Evolution to solve the problem. The authors made both centralized and decentralized versions. The modelling used evolution to optimize the next step or the next move, which iterated through all moves, produced the travel plan. The approach cannot be called complete as the modelling scenario assumed it was always possible to reach the goal directly from the taken position, which is an assumption for simpler maps only. In this paper the entire travel plan is optimized instead.

Kala (2012) solved the problem of planning with multiple robots using cooperative co-evolution. A memory based data structure was proposed using which the robots could share the shortest path information between the landmarks for additional coordination. The map however consisted only of narrow corridors from which only one robot could move, unlike this work which has open spaces. In another work Wang and Wu (2005) make use of cooperative co-evolution. The robots were however point sized. Both these approaches have restrictions of cooperative co-evolution as indicated in section 1.1.

Completeness is a major problem associated with reactive approaches. Sgorbissa and Zaccaria (2012) used a Voronoi graph based deliberative planning for guiding a reactive planner. Further completeness was added by identifying scenarios called roaming trails. This disallows any robot to move to a position which traps it, or from which motion as per the deliberative plan is not possible. Xidias and Azariadis (2011) solve a similar problem using centralized GA. The objective is to visit through a set of predefined landmarks without collision, which means the search space is restricted. The authors assume that the robots always move by the maximum allowable speed, which is derived by the trajectory curvature. This is unlike the proposed approach where the speed is itself optimized. Kapanoglu et al. (2012) use GA to solve the problem of area coverage. Their model only allows the robot to have rectilinear moves. A centralized path planning is adopted. The approach presented in this paper is decentralized with flexible moves. In a similar problem Szlapczynski and Szlapczynska (2012) solved the problem for ship planning, where the objective function was specifically designed for the problem domain. The authors used centralized GA for optimization with some customized operators. However the problem had much less robots as would be expected in the domain of mobile robotics.

Lin and Hsu (1997) studied the need of coordination between the different robots for the problem of object sorting. Here they made two protocols namely help based protocol where the robots used their local knowledge and cooperation based protocol where broadcast of information was used. Protocols may however be sub-optimal and are usually hard to specify for every problem. Larionova et al. (2006) considered coordination in the problem of olfactory area coverage. The authors coined the concept of bounding lines which denoted lines that just escape some point of obstacle. Each robot left chemical substances over the path as it moved, which could be sensed by the other robots in their motion, and enabled them to stay away from the already visited places. This is a decentralized coordination mechanism. However it does not ensure completeness or optimality.

1.3 Basic Idea and Main Results

Based on the facts the requirement of a coordination mechanism which balances the various advantages is clear. The mechanism should not take too long to execute, should be just with the various robots, and should be near-optimal and near-complete. The developed solution is based on iterative adjustments. Imagine a room is fitted with a gang of people, each standing at his/her place of interest which may be overlapping. However people do not generally prefer to be at congested places. Hence given such a profile, one would generally expect a person to shift somewhere within a step or two, wherever it is not too congested. It may not always be possible to move, in which case the maximum attempt is made. The landscape of room occupancy changes as the people move. One would normally expect the congested areas spreading out and the unoccupied regions of the room close to the congested areas being slowly occupied. People at the boundaries of the relatively congested areas may push outside towards the non-occupied or non-congested areas, thereby giving more space to people inside who might as well follow them. In this way the people may be required to further step out, expanding the occupancy of the particular congested region. People in very congested region, blocked in with obstacles on all sides naturally have no other option than to have someone go out completely and spot a new place to be in. The motion converges to a stage when all the people have sufficient un-congested space to be in, or enjoy the maximum that they actually can. The developed solution uses a similar analogy with the algorithm working in a trajectory space of individual robots.

The key contributions of the paper are: (i) A new probabilistically-complete and near-optimal coordination scheme is proposed which takes a reasonably small computational time. Computational time is larger than the prioritized genetic algorithm approach and lesser than the centralized approaches. The additional computational time is used to solve the greatest problem of prioritized genetic algorithm , which is its over-preference to a single robot. (ii) The paper deals with establishing a tradeoff between the individual optimal robot path and the overall optimal travel plan of multiple robots. (iii) The paper presents coordination via simultaneous alterations in the position and velocity profiles while many erstwhile techniques attempt to do so by alterations in any one. (iv) The paper, via simulation studies, presents interesting examples of multiple robots avoiding each other in the best possible mechanism.

2. Problem Statement

Consider a configuration space ζ in which a number of robots need to be traversed. Each robot R_i has a pre-known source S_i and a pre-known destination G_i. Let the trajectory of the robot be given by $\tau_i(t)$ and its velocity profile by $\tau_i^{'}(t)$ where t denotes time. For a collision free travel any robot must not collide with any static obstacle and the robots should not collide with each other, that is (equation (1)).

$$\tau_i(t) \otimes R_i \in \zeta^{free} \wedge \tau_i(t) \otimes R_i \cap \tau_j(t) \otimes R_j = \phi, \forall R_i \neq R_j \quad (1)$$

Here ζ^{free} denotes the free configuration space. The maximum speed of the robot is restricted, or $|\tau_i^{'}(t)| \leq v^{max}_i$. The robots may be subjected to nonholonomic constraints which places restrictions on their individual trajectories τ_i. The algorithm assumes that the configuration space ζ, the sources S_i and the goals G_i are known in advance.

Let τ^*_i denote the optimal trajectory of the robot R_i in the absence of any other robot. It is natural that τ^*_i may be un-admissible in the overall optimal plan due to mutual collisions between the robots. Let the overall optimal travel plan of the motion be τ^* in which robot R_i has a trajectory τ_i. The degree of cooperation of R_i is given by the magnitude by which its trajectory is deviated from the optional trajectory or $Coop_i = \| \tau_i - \tau^*_i \|$. The factor may usually be broken down into cases of high cooperation ($Coop_i > \varepsilon$) and low cooperation ($Coop_i \leq \varepsilon$) for some moderate ε. A robot steering or slowing to some degree to let some other robot pass by denotes a low cooperation, while changing the path largely denotes a high cooperation. Let $C(\tau_i)$ denote the cost of the path τ_i. The penalty of having multiple robots for a robot R_i is the rise in its path cost, or $Pen_i = C(\tau_i) - C(\tau^*_i)$. Penalty is an alternative and more convenient form to measure cooperation.

An effective solution to the problem would be to have the lowest average path cost. Hence objective is to minimize (2), or $\tau^*=\min(C(\tau))$

$$C(\tau) = \frac{\sum_i C(\tau_i)}{N} \quad (2)$$

Here N is the number of robots. A small deviation in the path of a robot rarely increases its path cost significantly, unless a collision-free path becomes collision prone. Hence the cost objective closely resembles the objective to minimize the overall deviation, to minimize the maximum deviation or to minimize the overall penalty. These are alternative objectives of the algorithm.

Consider a trajectory segment S of trajectory τ^*_i. The segment may be deviated by small degrees to produce different new segments. $Space(S)$ is defined as all collision-free segments produced by small deviations of segment S and recursively deviation of trajectories generated henceforth given by equation (3).

$$\text{Space}(S) = \text{Space}(S, \infty)$$
$$\text{Space}(S,0) = S$$
$$\text{Space}(S, k+1) = \text{Space}(S, k) \bigcup_{S_a \in Space(S,k), small\, d} (S_a + d),\ (S_a + d) \otimes R_i \in \zeta^{free} \quad (3)$$

Practically $Space(S)$ denotes the entire space around a trajectory segment bounded by obstacles. Let $|Space(S)|$ denote the maximum deviation between any two trajectories in $Space(S)$ given by equation (4). This denotes the width of space available between the obstacles.

$$|Space(S)| = \max(\|S_a - S_b\|),\ S_a, S_b \in \text{Space}(S) \quad (4)$$

The planning algorithm developed here first generates τ^*_i for all robots, and then deviates them to make them collision free as well as near-optimal as per the cost metric as far as possible. Consider two robots R_i and R_j that collide as per their individual plans τ^*_i and τ^*_j. Let the trajectory around the colliding point be S. If $|Space(S)|$ is wide enough, it would be simple to deviate either of the robots or both the robots in opposite directions to avoid the collision with some extra separation. In case of more number of robots which happen to potentially collide or lie near at the same time $|Space(S)|$ should be wide enough to accommodate all of them. In case the factor is not high, a robot may be required to alter the speed or trajectory profile so as to alter the region of contact S, if there needs to be any. Following a robot or waiting for it are some possibilities. Alternatively it may need to take a completely different route. In general $|Space(S)|$ denotes the ease of optimization in case of multiple robots, where S is the region where the inter-robot distances are small.

3. Algorithm

The task is to construct collision free trajectories of a given number of robots. The task is divided into two separate heads. The first head deals with the computation of optimal trajectory for a single robot or τ^*_i. This may ideally be done by any algorithm as per scenario. In this paper however a particular implementation would be presented which is used for the experimentation. Once the individual optimal paths are obtained, the second head is to formulate the overall optimal travel plan τ*. This would be done by iteratively modifying the paths of the individual robots, till the stopping criterion is met. The concept is shown in figure 1.

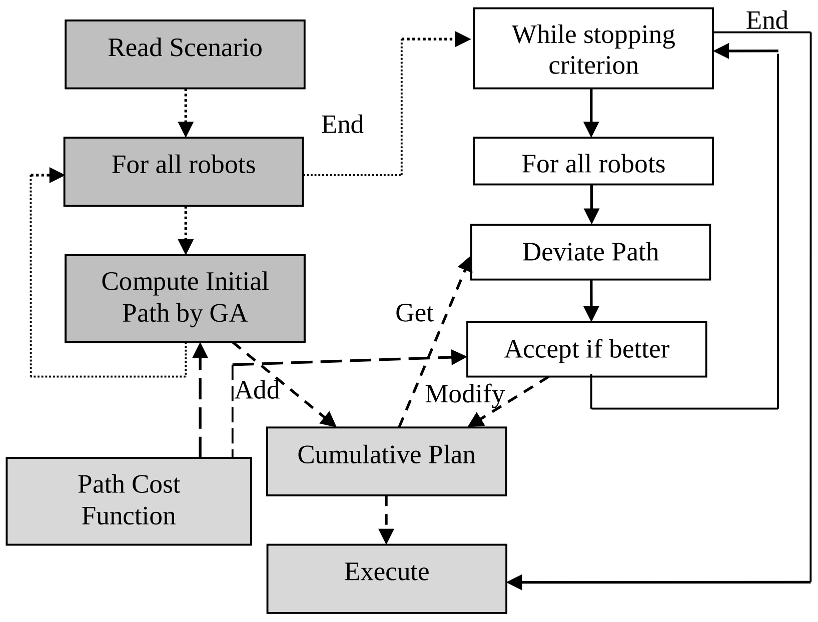


Figure 1: The general concept of the algorithm. The initial path is generated by genetic algorithm based on the scenario specified. The path hence generated is iteratively modified by a local search like algorithm for all the robots. Cumulative Plan stores the current plan of action of all robots. Every robot modifies its part by the search operation.

This breaking up of the problem into two sub-problems is motivated by the general mechanism in which humans walk where the route is largely decided independent of the other people, and once route is decided efforts are made to escape people on the go. Gestures act as communication indicating other's people plan. Using such decomposition the search space of the overall problem is highly restricted, with a little loss of optimality over specific scenarios. The hypothesis especially holds well due to the fact even though the scenario may have a large number of robots, few of them potentially collide with each other and most others can be corrected by small deviations. The coordination step has to hence work over the limited search space around the collision areas in the space time continuum of the configuration space, which is much limited as compared to the overall search space.

3.1. Initial Path Generation

The task is to construct the trajectory $τ^*_i$ from the source S_i to the destination G_i. The algorithm uses GA for the purpose. The algorithm is summarized in figure 2. The genotype is a collection of variable number of points in the configuration space ζ such that the first point is always the source S_i and the last point is always the goal G_i. Two restrictions are however placed on the genotype. First, no two consecutive points on the genotype can be at a distance closer than *d*. The GA may sometimes be prone to add a lot of points to fine tune the trajectory. However during coordination, too many points are likely to make the GA sub-optimal and computationally expensive. Second, no three consecutive points in individual can make an angle of less than 90 degrees. A lesser angle denotes a very sharp turn, which the robots may not be able to practically make.

The initial population is generated randomly. A problem with variable length individual representation is the need to guess the length of each generated individual. This number largely depends upon the map or the scenario used and hence cannot be specified at the design time. Hence the algorithm starts with the shortest individual, which consists of only the source and the goal. An operator 'add point' is applied to increase the size of this individual. The operator is allowed to increase the size of the individual till the operator results in a better individual at every step. A few attempts may be made by 'add point' in pursuit of a better individual before the process terminates and the resultant individual gets added to the population pool.

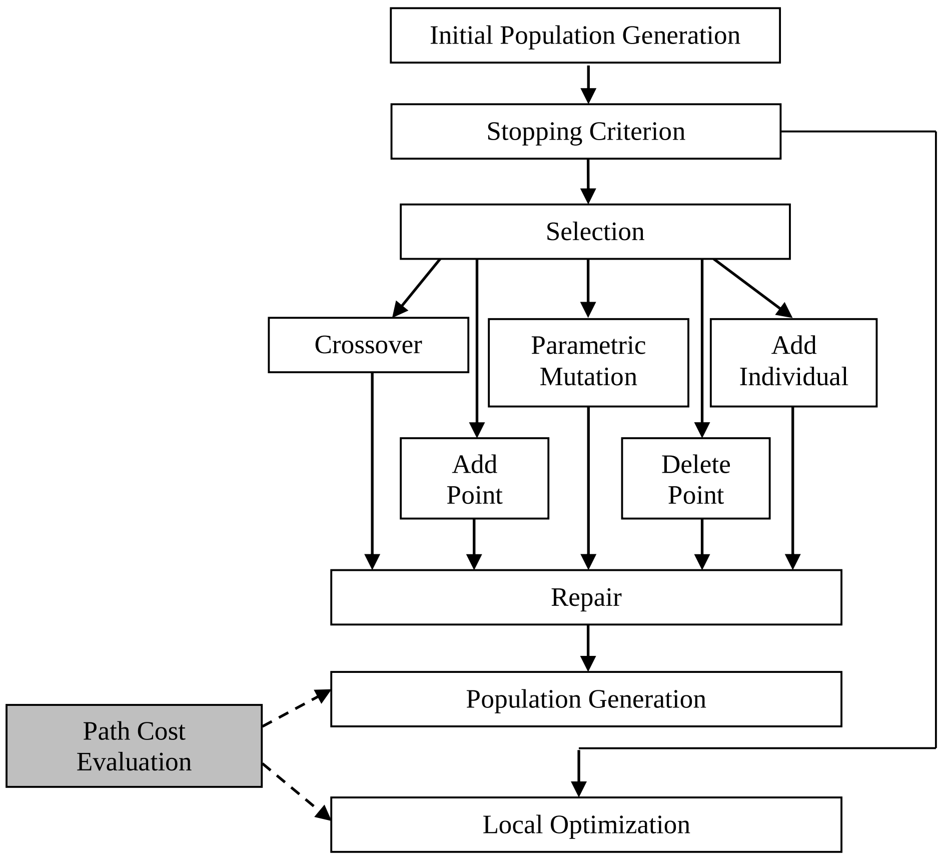


Figure 2: Generation of initial path by genetic algorithm. The genetic individual may be of variable size which is handled by all operators separately. Repair operator checks the validity of constraints and alters the individual accordingly. The final path generated is subjected to a few iterations of local optimization.

A number of genetic operators are constructed. Crossover operator is designed in which the first step is to replicate random points (excluding source and goal) in the shorter parent to make the parents of same length. Scattered crossover produces the children. The children are then processed to delete all redundant points. Parametric mutation operator shifts all points (excluding source and goal) by mutation rate which is taken from a Gaussian distribution. Two structural mutation operators are used. 'Add point' operator randomly selects two consecutive points in individual and adds a newly generated point between them. The newly generated point is biased to lie in between the two points. 'Delete point' (shorten) operator deletes a random point from the individual (excluding the source and goal). 'Add individual' adds a random individual to the population pool. This operator is specially designed for the fact that since the solution returned by the GA is to be further processed, convergence to a local optima is more dangerous than to lie anywhere near the global optima. 'Repair' modifies points starting from the source (which is untouched) so that the individual representation constraints are met. The penultimate point may require moving goal, which is not allowed, in which case the point is deleted.

In the initial few generations a coarser fitness evaluation is performed while in the latter generations a time consuming finer fitness evaluation is performed. The granularity of fitness evaluation decreases with the increase in generation (Bohlin and Kavraki, 2000; Kala et al., 2011). The points represented in individual representation are used as control points of spline curve (Bartels et al., 1987; de Boor, 1978) which is the trajectory. The path cost function is discussed in depth in section 3.3. For the evaluation for this section no other robot is considered.

The stopping criterion is specified in terms of both maximum number of generations and stall generations. Due to the granularity based collision checking, it is evident that the fitness of an individual would change even though no change may be made to it. Hence the term stall is defined in terms of no change in the fittest individual. The next generation is chosen by creating a mix of population obtained from the genetic operators and the previous generation, from which the fittest half individuals are chosen as the next generation. The path hence computed by the GA is subjected to a few iterations of local search (Russell and Norvig, 2003) using operators similar to mutation in the GA. This was done on the basis of experimental observations, which indicated local search could save the need to run GA using computationally expensive settings. The robot path hence computed is assumed to be traversed by the highest allowed robot speed v^i_{max}. This forms the optimal robot trajectory τ^*_i.

3.2. Multi-Robot Coordination

The task associated with the coordination module is to use the individual optimal paths τ^*_i to produce the overall optimal plan τ^*. Let τ represent any travel plan as a collection of travel plan of individual robots. The basic hypothesis is to adjust the position of a robot in the space time configuration space, so as to best fit into the travel plan so far. The adjustment of every robot modifies the travel plan, which the other robots need to adjust to. In this manner iterative working over all the robots produces the optimal travel plan. The hypothesis of iteratively mutually adjusting the paths of the robots is inspired from the fact how one would manually attempt to solve the entire problem. Given a number of robots and knowing they might interact in multiple ways, it would be fair to first compute the optimal individual trajectories. Once done, it should be possible to locate the collision points and pull them apart from each other as far as possible, attempting not to cause further collisions. It should further be possible to pull trajectories which lie too close when simulated. Any extra space created can be used to increase the gap between trajectories. An important observation of the process is that two colliding robots being tightly packed by robots on all sides cannot be avoided for collision. However since separation between colliding robots and the surrounding robots is low, the surrounding robots would further attempt to lie far apart. In this manner iteratively the robots attempt to push each other outer and outer, and in case the space is bounded by obstacles, they attempt to fit in with the maximum separation possible.

From the perspective of any single robot, the algorithm acts as a local search on a constantly changing environment. At each iteration the robot has a copy of the best plan so far τ, in which it attempts to modify its trajectory so as to improve its path cost. The robot attempts to modify the trajectory using a set of operators, and if a better trajectory is found, it replaces the old trajectory in the plan τ. If no better trajectory can be found for a few iterations, the algorithm moves to the next robot or the next iteration of the iterative coordination cycle. Only a single replacement is allowed for every robot per iteration of the coordination cycle, which prohibits a robot dominating the scenario.

The operators are the same as the mutation parameters of the GA. The operator of 'add individual' is also added. Many times it may be beneficial to take a completely different route due to congestion at particular areas, which is what the operator models. The additional difference between the GA and the optimization at this stage is that this stage has to additionally optimize the speed profile. Many schemes disregard the speed aspect. An optimal plan is however a mix of both speed and trajectory alterations (Kant and Zucker, 1986). As an example consider a high speed robot generated behind a low speed robot in a narrow corridor. The fast speed robot has to slow down to avoid collision, and hence generate a feasible plan. Hence the individual representation is altered to account for the new genes.

The additional genes are in form of a collection of triplets $<s_a, d_a, v_a>$ representing that robot moves with speed v_a for the time period of s_a to s_a+d_a in its journey. In case of conflicts, the speed corresponding to the last triplet in the collection is chosen. For all times in the robot's journey not covered in any triplet, it is assumed that the robot moves with maximum allowed speed v^i_{max}. An additional operator is constructed to alter the triplets of speed using a Gaussian mutation. In the current implementation only a single triplet is used. The initial value of this triplet, which is appended to all robots, is $<0, 0, v^i_{max}>$ which produces the same effect as the robot asked to drive through with the highest speed. As optimization goes on, this triplet enables a robot to drive slowly for some initial part of its journey, or to wait for some part of its initial journey. This delays its move and eliminates some unavoidable collision, at the same time making separations large. Additional triplets may be employed to model behaviours where robots need to wait to avoid an initial potential collision, move ahead, and later again slow down to avoid another potential collision. The coordination space of the robot is thus capable of representing all possible plans. The notion of probabilistic complete nature of the genetic algorithm can hence be extended to the coordination algorithm as well.

The algorithm is clearly not cooperative between robots, as a robot has no direct incentive for assuming a trajectory which results in better trajectory of the other robot. By implementation it appears that the algorithm is rather competitive wherein different robots compete in pursuit of their optimal trajectories. However as a characteristic of the problem competence leads to cooperation. If a robot R_i has a poor path cost due to a robot R_j, then the robot R_j also has a poor path cost due to the robot R_i. Hence if R_i gets overall incentive for assuming a trajectory by which R_j is penalized, R_i also gets a penalty due to the trajectory of R_j which acts as the penalty for non-cooperation. Usually this would be due to less separation being maintained between R_i and R_j, or a collision between them. Coordination talks about the direction (side) in which two robots avoid each other, along with the magnitude by which avoidance is done. The algorithm cannot over-prefer a robot hence a leader approach cannot be taken. This puts forth the additional issue of the robots coming in consensus with each other regarding the direction of avoidance. The competitive pressures created as a result of the competitive nature of the algorithm are the agents of both these tasks. Practically this is seen as the different robots 'pushing' each other in space time configuration space, till each of them occupies a trajectory which puts the robot comfortably away from all other robots at all times. In other words a robot may be forced to adjust its trajectory to imply with the trajectory of some other robot.

3.3. Path Cost Function

A number of objectives are considered which make the total path cost of a trajectory τ_i. The trajectory is in form of consecutive points in the configuration space. The first objective is the total time of travel of the robot given by equation (5).

$$T = \sum_a \frac{\left\|\tau_{i,a+1} - \tau_{i,a}\right\|}{V_{i,a}} \tag{5}$$

Here $\tau_{i,a}$ is the ath point in the trajectory. $V_{i,a}$ is the specified speed at point $\tau_{i,a}$.

The second objective is static clearance which is the distance of the trajectory from static obstacles. The objective is given by equation (6).

$$C = \sum_a \exp\left(-Clear(\tau_{i,a})^2 / 2\sigma_c^2\right)\left\|\tau_{i,a+1} - \tau_{i,a}\right\| \tag{6}$$

Here $Clear(\tau_a)$ is the closest distance of robot placed at $\tau_{i,a}$ with any static obstacle. This is approximated by measuring the clearance at the corner points of a rectangular robot, at each corner measured along 8 sampled directions. The minimum distance from 4 corners is taken. The distance measurement is limited to *maxClear*, which is the maximum preferred clearance. The factor σ_c controls the change of clearance by increasing distances to obstacle.

Measurement of $Clear(\tau_{i,a})$ is a time consuming process. Two possible ways are to have this factor pre-computed for all points in the configuration space and use it directly at GA computation time (Kala et al., 2011). This is time consuming at pre-processing stage but quick at optimization stage as well as gives precise values. The other option is to compute the value in the fitness evaluation which makes the optimization slow. In this algorithm the values are computed in the fitness evaluation, but once computed these values are stored for direct use at later stages. This saves a lot of time since mostly on convergence all individuals have similar regions being queried for clearance again and again. Hence as per approximations, the algorithm traverses from all corners of robot along the 8 sampled directions to compute clearance. Whenever a point is found for which clearance has already been computed, the same value is used for computation for the particular corner. Equation (6) converts the clearance into an exponential scale, which more practically expresses the cost metric. Multiplication of factor $\| \tau_{i,a+1} - \tau_{i,a} \|$ means that a similar clearance value for the continuum from $\tau_{i,a}$ to $\tau_{i,a+1}$ is assumed.

The next objective is the total length of the trajectory within static obstacles. The next couple of objectives are clearance from any other robot and total length of trajectory within which the robot is colliding with another robot. The clearance in this case is approximated as minimum distances between the robot corners. The next objective is genetic individual size. This factor is put due to the fact that lesser sized individuals imply lesser steering and hence smoother paths. A less turn prone drive is always considered better. Further penalizes the individual size thus prohibiting the convergence to local minima, or a computationally intensive coordination stage. Last objective is distance to goal from first collision. Suppose a robot lowers its speed in modified plan. It may yet collide with the robot and have a similar trajectory. However this may be a step towards lowering the speed enough to completely avoid the collision. This objective incentivizes such steps. The same reasoning would hold true for the robot with which the collision was happening, which is incentivized to increase speed.

Each objective has an associated parameter weight associated with it. The setting of different parameters is itself a large task. However the different objectives serve different purposes and may be easily set. Time is taken as a base objective whose weight is set to 1. The collision and clearance objectives corresponding to the robots are given same weight as the static counterparts. The weight of collision is set alarmingly high as compared to any other weight. The clearance weight is set much less, but somewhat larger than 1. These objectives would be multiplied by a factor proportional to the path length; however the factor of size is a direct add-on, which may hence be given a weight larger than clearance but smaller than collision. The last objective plays a minor role and hence a small weight.

4. Simulation Results

The approach was tested through simulations. The simulation agent and the algorithm were developed in MATLAB. From a simulation perspective, the first part of the problem was to read a pre-specified scenario and generate a travel plan using the algorithm. The algorithm itself had different stages. The second part was to move the individual robots as per plan, which was graphically displayed at the user screen.

Table 1: Typical Parameter Values

S. No.	Parameter	Value
GA Parameters		
1.	Maximum Generations	20
2.	Maximum Stall Generations	5
3.	Population Size	20
4.	Minimum distance between points in individual representation	4
5.	Granularity change in GA	10 to 1
GA Operator Parameters(Percent of individuals generated by particular operator)		
6.	Crossover	10%
7.	Parametric Mutation	50%
8.	Structural Mutation – Add Points	20%
9.	Structural Mutation – Delete Points	10%
10.	Add individual	10%
Local Optimization Parameters		
11.	Maximum Generations	100
Coordination Parameters-Probabilities of invocation of operators		
12.	Parametric Mutation	0.4
13.	Structural Mutation – Add Points	0.1
14.	Structural Mutation – Delete Points	0.2
15.	Speed Alteration	0.2
16.	Add individual	0.1
Path Cost Function Weights		
17.	Time	1
18.	Clearance	10
19.	Collision with static obstacle	1000000
20.	Collision with robot	100000
21.	Individual Size	50
22.	Distance to goal on first collision	1

Table 2: Experimental Results

Robot No.	Robot Specifics (source, goal, maximum speed)	Time	Static Clearance, Collision	Robot Clearance, Collision, distance from collision	Individual Size	Path cost
Complex Obstacle Scenario						
1.	(40, 40), (460, 460), 10	67.1091	55.7654, 0	24.642, 0, 0	4	1071.2
2.	(460, 460), (40, 40), 10	80.1067	89.8587, 0	25.057, 0, 0	3	1379.3
3.	(460, 40), (40, 460), 10	102.1337	53.7016, 0	17.9983, 0, 0	3	969.1324
4.	(40, 460), (460, 40), 10	65.3843	106.1674, 0	37.3946, 0, 0	4	1701
Average Path Cost						**1280.16**
Single Obstacle Scenario						
1.	(40, 40), (460, 460), 5	133.0317	20.9829, 0	19.0006, 0, 0	3	682.8665
2.	(460, 460), (40, 40), 10	72.9751	20.885, 0	18.987, 0, 0	3	621.6949
3.	(460, 40), (40, 460), 10	63.879	19.1408, 0	18.8541, 0, 0	3	593.8278
4.	(40, 460), (460, 40), 10	68.289	22.7803, 0	20.0928, 0, 0	3	647.0196
Average Path Cost						**636.3522**
Robot Coordination Scenario						
1.	(40, 40), (460, 460), 10	60.4306	17.748, 0	17.2969, 0	3	560.8799
2.	(460, 460), (40, 40), 10	61.7661	17.2608, 0	18.175, 0	2	516.1239
3.	(460, 40), (40, 460), 10	60.7323	17.9749, 0	17.3485, 0	3	563.9661
4.	(40, 460), (460, 40), 10	65.2688	19.65, 0	18.2927, 0	3	594.6961
Average Path Cost						**558.9165**
Robot wait Scenario						
1.	(250, 460), (460, 460), 5	139.512	32.0207, 0	17.1398, 0	4	831.1173
2.	(460, 460), (40, 40), 10	73.6623	43.3519, 0	24.198, 0	6	1049.20
Average Path Cost						**940.15865**

4.1 Results on Various Scenarios

The purpose behind testing was to construct diverse scenarios which more or less cover every scenario that the robots may practically face in a realistic world. The algorithm has a number of parameters. Most of the parameters are of the GA or similar, whose effect is well studied in literature and hence can be easily set. The manner of setting the objective weights has already been discussed. The major factor in deciding the parameters is based on the general scenarios that the robot may face. A brief study was done on general scenarios and the kind of paths generated and the optimization done on them was the basis by which the parameters were set. The parameters used in the study for the scenarios presented are summarized in table 1.

The first requirement of any algorithm is the ability for robots to go through complex obstacle frameworks and in this case with multiple mobile robots. This ability is tested with results shown in figure 3(a) and video 1. The large number of obstacles took much of the space of the map. The robots had to fit in the rest of the space, and avoid each other. Ideally the robots would have collided at a place where obstacle density was high, and it would have been difficult to avoid each other. Two robots hence preferred waiting for another robot to pass by. In this manner all the robots could enjoy a high clearance, which would not have been possible in any other case. The various metrics of the scenario are summarized in table 2. The presence of large number of obstacles make planning of a single robot a difficult issue, but largely simply the manner in which different robots interact or avoid each other. Hence in the second scenario a single obstacle is taken, while the maximum speed of one robot is reduced to give diversity. Robots need to avoid the obstacle and any other robot nearby. It may be seen that the robots can compute good trajectories which are smooth and not very close to each other, while still avoiding the obstacle. The scenario is given in figure 3(b) and video 1.

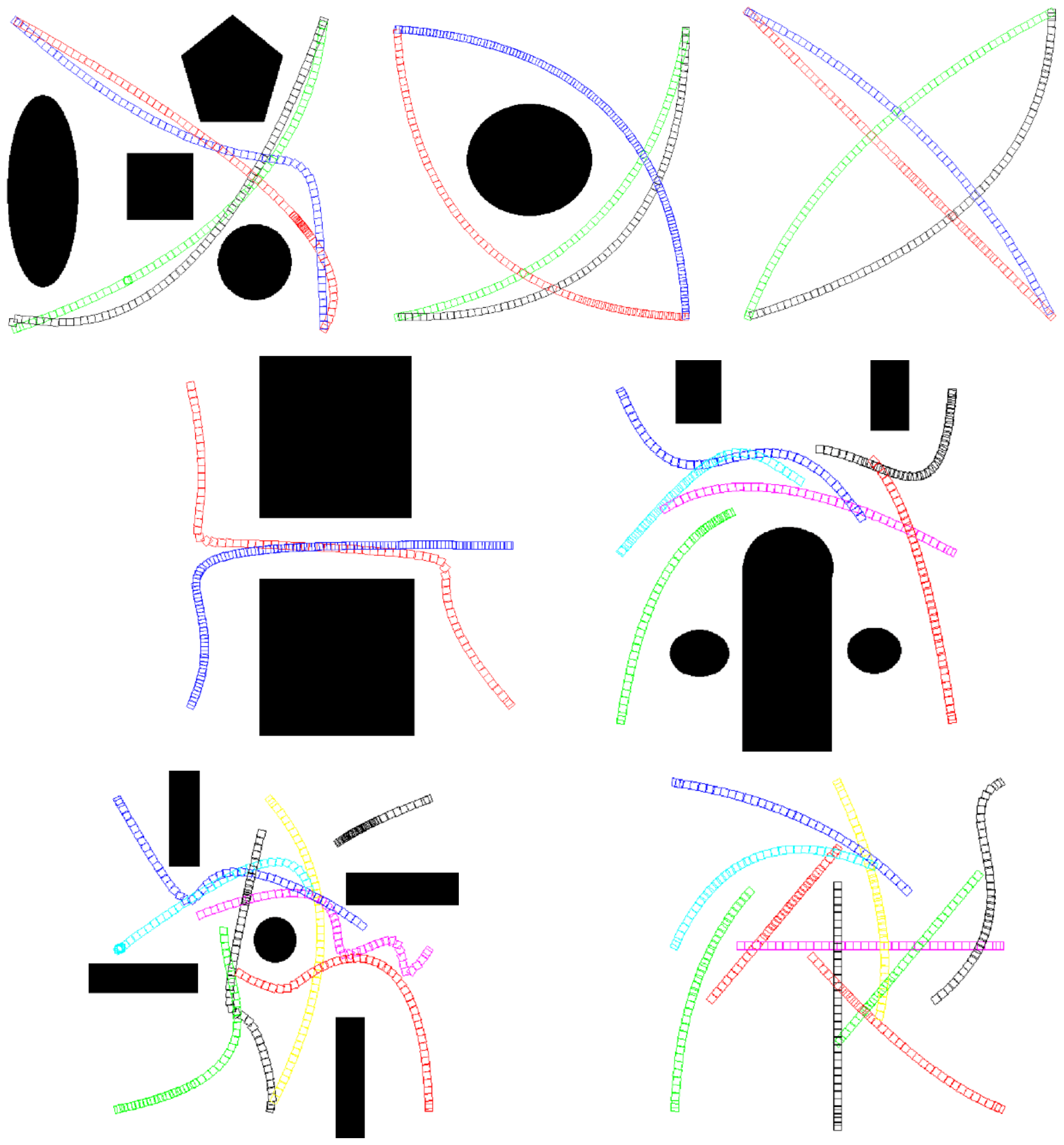


Figure 3: Experimental results (a) checks the ability of the robots to first generate plans in a complex environment, and then to avoid each other. (b) is a simpler scenario where the robots get adequate space to fit themselves in, and they must hence ensure wide spaces between them. (c) ensures that all robots collide at same point as per their ideal plans, and hence they need to mutually avoid each other . (d) forces a robot to slow down or wait, as the robots would collide if they move by any plan with their maximum speeds. (e) has a passage within which all robots need to schedule and place themselves. (f) has a central obstacle along with obstructions on the sides which makes planning for each robot difficult. (g) tests the ability to plan a very large number of robots. To see the entire motion of all robots please refer video 1.

In the next scenario no obstacle is taken in order to completely understand the manner in which robots avoid each other. Obstacles may make some robot go from a way which is completely absent in the local search domain of other robot. Absence of obstacles makes all sorts of collisions possible between robots. Robots are generated at one corner and they attempt to reach the other corner. Hence ideal trajectories of all robots collide at the centre of the map. The results are given in figure 3(c) and video 1. In all the scenarios presented so far, it was possible for the robots to go by their maximum speeds and yet get to their goal. The last scenario is constructed so that robots cannot reach their goals by their maximum speeds and one of the robots has to slow down. The scenario is presented in figure 3(d) and video 1. The robot generated in the middle had half the maximum speed as compared to the one generated at corner. Due to narrow corridor, they could only move one at a time. Hence the plan generated faster robot followed by the slower one.

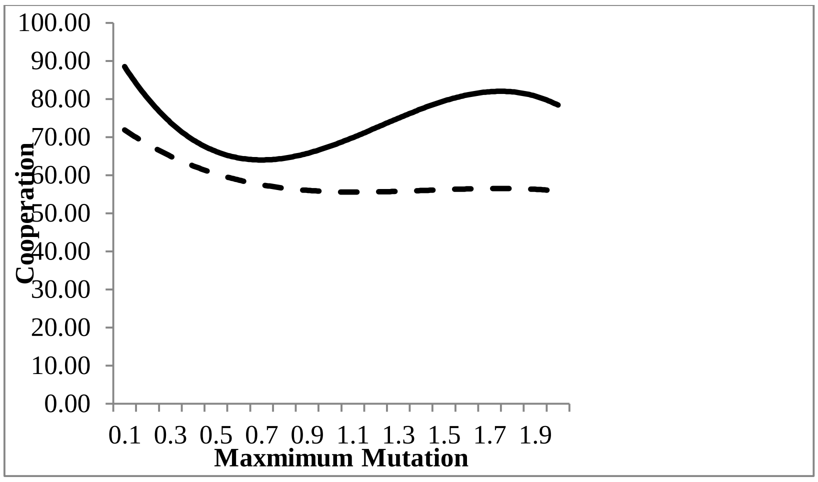
Experimentation is further carried for more complex scenarios for a higher number of robots. In the next scenario 6 robots are generated which need to pass through a kind of passage formed by obstacles in the upper half (figure 3(e)). The other scenario has a centrally located obstacle with blockages all around, in which 8 robots generated at the corners and corner centres have to go to the opposite end (figure 3(f)). The last experiment is an obstacle-free map with 10 robots, 4 at the corners, 4 at corner centres, and 2 at the centre. All (non-centre) robots go to the opposite end (figure 3(g)). These scenarios can be found in video 1.

4.2 Algorithm Analysis

The algorithm has two parts which are initial path generation and coordination. The algorithm is largely invariant of the mechanism by which the initial path is generated, although a mechanism was proposed using GA. For other algorithms it should be possible to sample out points from the plan and use them in coordination phase. The second part or the coordination part is essentially the algorithm for which the parameters need to be studied. The algorithm simply deviates trajectories by some degree, in which the major share is of the mutation operator. The other operators create the needed number of points, and possibly close to positions where collisions are prone to happen. In the next experiment the maximum mutation was restricted. Since coordination is being studied, the best scenario is the third scenario which was used. The optimal (non-coordinated) plan is a straight line from source to goal. An extra point was added as mid-point of path for all robots. In coordination part now the robots have the needed number of points needed to produce a collision-free path, and hence only mutation operator was used for deviation. The experiments were done for a number of maximum mutation rates.

Three metrics were studied, the average cooperation (measured as a difference between the path cost returned by the algorithm and the path cost of single-robot optimal path) for all robots, maximum cooperation shown by a robot, and the minimum number of fitness evaluations needed for generation of a path which is 20% as good as the optimal coordinated travel path. Average cooperation gives an indication of the average path cost. Since different robots have different optimal path costs, it is be a better metric of study. The last metrics gives an indication of the computation time. It is with respect to the fact that if one can generate a travel plan 20% as good as the optimal plan cost, the resultant plan may still be acceptable for navigation, or optimal path may be produced by small mutations henceforth, which is an easy endeavour. Results are shown in figure 4. All values are first averaged for 5 independent runs and plotted as a trendline.

In the region of small mutation values, the average and maximum cooperation are high showing sub-optimality. However the average cooperation stays rather constant henceforth for different maximum mutations. If one the robot cooperates by a large amount, its path quality is poor, but it enables some other robot to have a small path cost and show less cooperation. Hence the cooperation averages. In other words even though a robot dominates, the average cooperation shows no change. The maximum cooperation however exposes the sub-optimality in such scenarios. Ideally all robots should have same cooperation and should deviate by equal amounts. At higher mutation values however a robot moves by a large magnitude as the initial step, while other is yet at its single-robot optimal path. Now robots have large distances and do not collide. There is less incentive for the optimal path robot to change its path. The maximum cooperation is however large for the robot which initially moved. Still further mutations do not result in collision-free paths, as the robot often goes outside map by mutation. Hence the study is restricted to the mutation values shown in the graph. Similar trend is shown for the minimum number of fitness evaluations required. For small values robot deviates less and hence more iterations are needed. For larger values the deviation is large and the robot may lie outside the map or too far apart from the optimal move region.



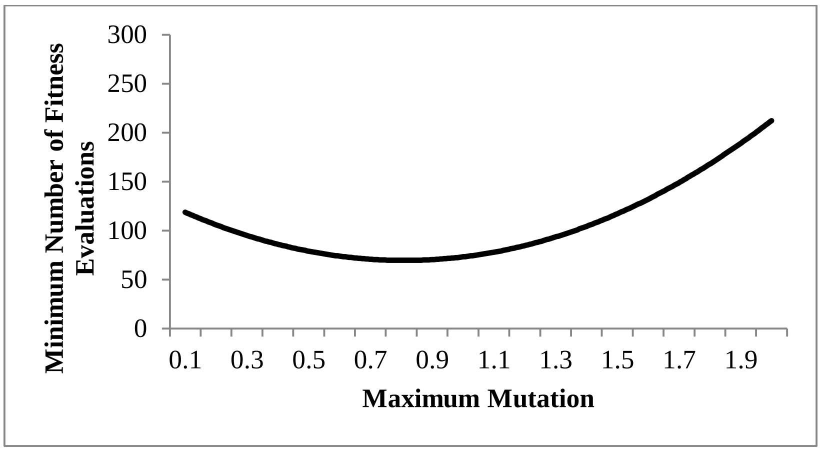


Figure 4: Effect of varying maximum mutation rate. (a) shows the average amount by which a robot has to deviate from its optimal path in order to avoid collision as well the maximum amount by which any robot deviates. High difference between the two lines shows that at least one robot is having a large deviation from its ideal path, while the other robots may not be deviating by large amounts. (b) shows that too low and too high mutation makes it more difficult to generate good plans quickly.

The other task is to test the effect of different number of robots. The additional robots into the scenario may come into the optimal paths of existent robots and may hence demand extra cooperation. The study of this factor for different scenarios is shown in figure 5. In all these experiments the optimal path costs were computed by long GA runs, while the obtained path costs were computed by small runs of the initial path computation and coordination modules. Hence the cooperation may be positive for a single robot case as well. The same metrics are studied. It can be observed that if the different robots do not necessarily come in each other's path as per optimal plan, the extra cooperation needed is negligible. However the additional robots can many times badly affect the other robots, making their path costs high. The minimum fitness evaluations here are addition of the fitness evaluations for initial path generation as well as coordination. Since scenario 1 is complex for initial path generation, the number of evaluations is large. One would normally expect the computational cost of 2 robots be double the cost of a single robot. This is not the case since different robots have different ease by which their initial paths are generated, and coordination is a common step for all the robots. Motion of one robot may make it unnecessary for another robot to move.

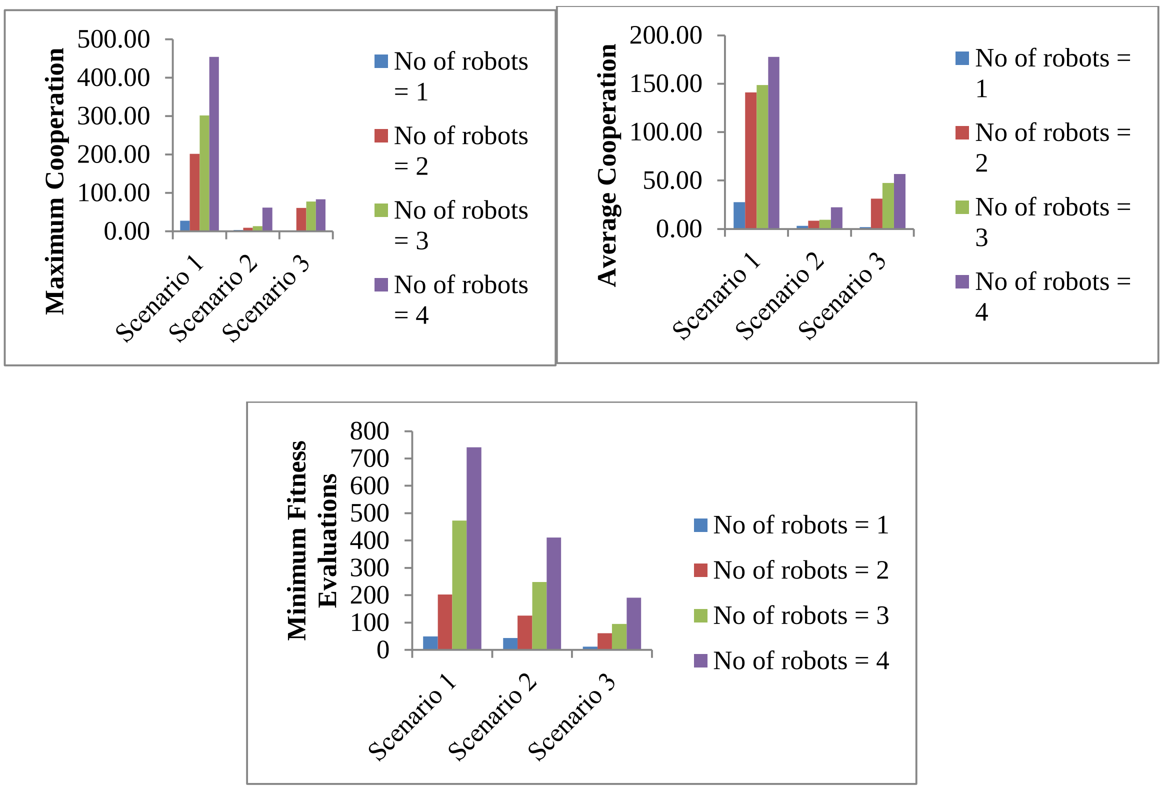


Figure 5: Effect of increasing the number of robots. Increasing the number of robots may place the new robots near the optimal paths of old robots, hence forcing them to move, which increases (a) the maximum cooperation, (b) the average cooperation, and (c) the fitness evaluations for generation of a solution.

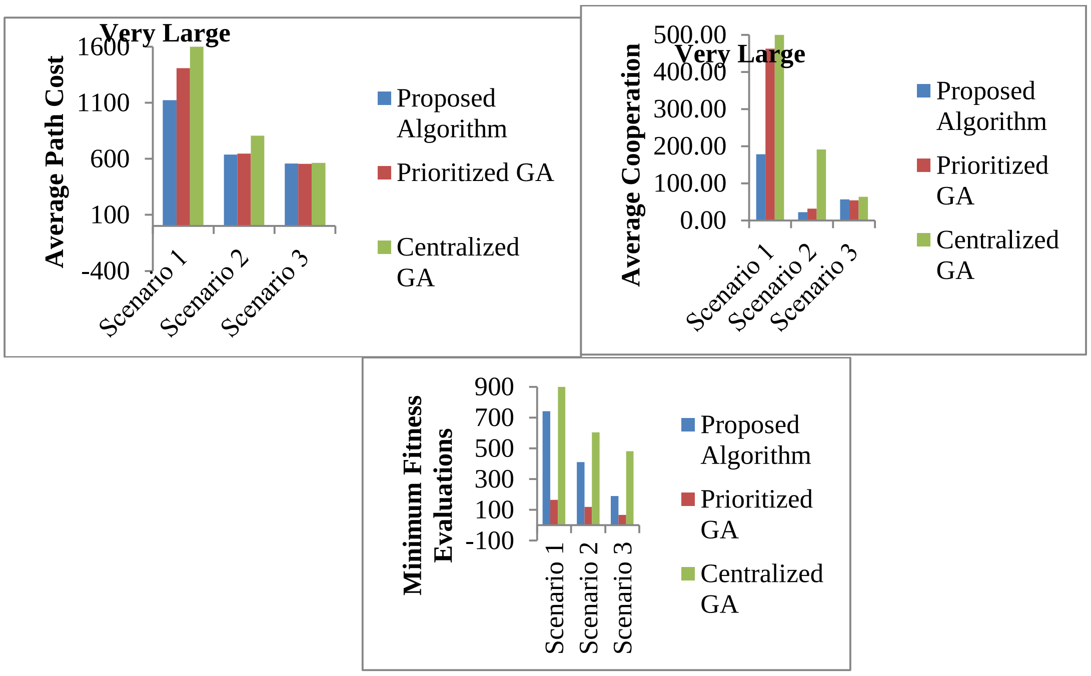


Figure 6: Comparative analysis with different algorithms (a) Average Path Cost (b) Maximum Cooperation (c) Minimum fitness evaluations for generation of a solution. Centralized GA for scenario 1 did not generate any feasible solution and hence all the metrics are significantly large (trimmed in figure). Proposed algorithm is intermediate in terms of computation cost and best in terms of the path cost and cooperation metrics.

4.3 Comparisons with other approaches

In order to assess the performance of the proposed algorithm, the experiments are repeated over two more algorithms. The first algorithm is the prioritized GA. Here every robot is given a priority and robots are planned in the order of their priorities. The other algorithm is the centralized GA. Here the trajectories of all the robots are fused into a single genetic individual, which is optimized in the evolutionary process. The studied metrics are average path cost, average cooperation, maximum cooperation and least number of fitness evaluations. Only first three scenarios are used for comparison which marks a trend from very simple to complex scenarios through an intermediate scenario. The results are given in figure 6. Centralized GA could not generate a feasible plan for first scenario within a large span of time and its values were much larger than the ones shown in graph.

Based on the figures it is clear that the proposed algorithm takes more computation time as compared to the prioritized GA. This is because the proposed algorithm needs to first generate optimal paths of all robots and then carry coordination, while the prioritized GA plans each robot once. However the approach is a lot quicker than the centralized GA, which could not generate a feasible travel plan for the first scenario. However the proposed algorithm is much better in terms of cost and cooperation metrics as compared to other algorithms (with the exception of scenario 3 in which case it performs as good as the prioritized GA). The prioritized GA may place a robot roughly towards the centre of obstacles, leaving a wide gap. The other robot may need to fit itself in one such gap, leaving less space available. The centralized GA has a highly dimensional space, which is hence not possible to optimize in small computation times.

The reason for the proposed algorithm not performing better for scenario 3 as compared to prioritized GA is twofold. First, the optimal path in the third scenario as returned by the initial GA contains no turning points (genotype only) and addition of point (which is a must for coordination) cannot be done without deviation path by large amounts. In other cases the addition of point would have been bounded by the neighbouring points in the optimal path (genotype only). Once any of the robots is deviated by any large amount, there is no incentive for the other robot to move. Hence the algorithm behaves like prioritized GA, which is verified by experimental observation. This may be taken as the worst scenario and the worst performance of the proposed algorithm, which behaves like prioritized GA. The second reason is that the proposed algorithm attempts not to over-prefer a single robot, but can only do so within a performance window. As per the algorithm, the last robot which deviates its path has a slight preference bounded by the maximum deviation that may have been produced by any robot. The difference observed is within the performance window which is not significant.

The different scenarios differ in terms of free spaces available for the robots or $|Space(S)|$. This factor is least for the first scenario and largest for the last. Based on experimental results it is seen that the time required for computation alarmingly increases with decrease in $|Space(S)|$ for centralized GA, while the least increase is recorded in prioritized GA. However the path costs of solution largely increases for centralized GA with decrease in $|Space(S)|$, while decrease in the proposed algorithm is the least. Based on these results performance for other scenarios may be extrapolated. Hence it can be seen that the proposed algorithm performs better as compared to other approaches, especially when the scenarios get highly complex. Hence the betterment and near-optimal nature of the algorithm can be ascertained. Further the aim of designing an algorithm which takes extra computation as compared to prioritized GA, but in return gives better results has been met.

5. Conclusions

Planning the collision-free trajectories of multiple robots is a much difficult task as compared to planning the trajectory of a single robot. The different robots may come into each other's way in a variety of ways, which raises issues for the planning and coordination algorithm to solve. The basic intent behind the design of algorithms is to generate the best plan possible within the limited computing infrastructure. The algorithm should be able to perform in a variety of scenarios, from simple to more complex ones. In this paper the problem was separated into two parts, generation of optimal single robot trajectories and coordinating various robots by modifying the optimal trajectories. Experimental results showed the algorithm to be better than both centralized GA and prioritized GA. The centralized GA was too computationally expensive for complex tasks, while sub-optimal in simpler tasks. As compared to prioritized GA it was observed how a little additional computational time can lead to better travel plan, as observed by a number of metrics. This is practical from the point of view that deliberative planning techniques usually have some time, which can be used in entirety for the generation of plans, as long as additional time gives benefits in terms of plan quality in return. Currently a major thrust is on local searching the region around the optimal trajectory of the robot, which means the algorithm may not perform well in case too many robots get scheduled in same regions at same times. In future attempt would be to consider all combinations of globally competing paths of all robots, which is a computationally expensive task. Similarly making the operators and parameters adaptive may significantly boost the computational performance. Other grounds for future work include better addressing the issues of completeness, use of algorithm for scenarios involving high congestion amongst the robots and implementation of the algorithm in real-time.

References

T. Arai, E. Pagello, L. E. Parker, “Editorial: Advances in Multi-Robot Systems,” *IEEE Transactions on Robotics and Automation* 18(5)(2002): 655-661.

R. H. Bartels, J. C. Beatty, B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling* (San Francisco, CA: Morgan Kaufmann, 1987).

J. L. Baxter, E. K. Burke, J. M. Garibald, M. Normanb, “Shared Potential Fields and their place in a multi-robot co-ordination taxonomy,” *Robotics and Autonomous Systems* 57(10)(2009): 1048-1055.

M. Bennewitz, W. Burgard, S. Thrun, “Optimizing schedules for prioritized path planning of multi-robot systems,” *Proceedings of the 2001 IEEE international conference on robotics and automation*, pp. 271–276, 2001.

M. Bennewitz, W. Burgard, S. Thrun , “Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots,” *Robotics and Autonomous Systems* 41(2-3)(2002): 89–99.

R. Bohlin, L. E. Kavraki, “Path Planning Using Lazy PRM,” *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pp. 521-528, 2000.

J. Chakraborty, A. Konar, U. K. Chakraborty, L. C. Jain, “Distributed cooperative multi-robot path planning using differential evolution,” *Proceedings of the IEEE World Congress on Evolutionary Computation*, pp. 718-725, 2008.

C. de Boor, *A Practical Guide to Splines* (Heidelberg: Springer, 1978).

R. Kala, “Multi-Robot Path Planning using Co-Evolutionary Genetic Programming,” *Expert Systems with Applications* 39(3)(2012): 3817-3831.

R. Kala, A. Shukla, R. Tiwari, “Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning,” *Artificial Intelligence Review* 33(4)(2010a): 275-306.

R. Kala, A. Shukla, R. Tiwari, “Dynamic Environment Robot Path Planning using Hierarchical Evolutionary Algorithms,” *Cybernetics and Systems* 41(6)(2010b): 435-454.

R. Kala, A. Shukla, R. Tiwari, “Robotic Path Planning using Evolutionary Momentum based Exploration,” *Journal of Experimental and Theoretical Artificial Intelligence* 23(4)(2011): 469-495.

K. Kant, S. W. Zucker, “Toward Efficient Trajectory Planning: The Path-Velocity Decomposition,” *The International Journal of Robotics Research* 5(3)(1986): 72-89.

M. Kapanoglu, M. Alikalfa, M. Ozkan, A. Yazıcı, O. Parlaktuna, “A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time,” *Journal of Intelligent Manufacturing*, 23(4)(2012): 1035-1045.

L. E. Kavraki, M. N. Kolountzakis, J. C. Latombe, “Analysis of probabilistic roadmaps for path planning,” *IEEE Transactions on Robotics and Automation* 14(1)(1998): 166-171.

O. Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,” *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, pp. 500-505, 1985.

S. Larionova, N. Almeida, L. Marques, A. T. de Almeida, “Olfactory coordinated area coverage,” *Autonomous Robots* 20(2006):251–260.

J. C. Latombe, *Robot Motion Planning*, (MA: Kluwer Academic Press, 1991).

F. C. Lin, J. Y. Hsu, “Cooperation Protocols in Multi-Agent Robotic Systems,” *Autonomous Robots* 4(1997): 175–198.

V. J. Lumelsky, K. R. Harinarayan, “Decentralized Motion Planning for Multiple Mobile Robots: The Cocktail Party Model,” *Autonomous Robots* 4(1997): 121–135.

L. E. Parker, F. E. Schneider, A. C. Schultz, *Multi-robot systems: From swarms to intelligent automata vol. 3*, (NY: Springer-Verlag, 2005).

M. A. Potter, K. A. de Jong, “A cooperative coevolutionary approach to function optimization,” *Proceedings of the third conference on parallel problem solving from nature*, Springer-Verlag, Berlin, Germany, pp. 249–257, 1994.

M. A. Potter, K. A. de Jong, “Cooperative coevolution: An architecture for evolving coadapted subcomponents,” *Evolutionary Computing* 8(1)(2000): 1–29.

S. K. Pradhan, D. R. Parhi, A. K. Panda, R. K. Behera, “Potential field method to navigate several mobile robots,” *Applied Intelligence* 25(3)(2006): 321-333.

S. J. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach 2nd ed.* (Upper Saddle River, NY: Prentice Hall, 2003), pp. 111–114.

G. Sánchez-Ante, J. C. Latombe, “Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems,” *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington DC, pp. 2112 – 211, 2002.

M. F. Selekwa, D. D. Dunlap, D. Shi, E. G. Collins Jr, “Robot navigation in very cluttered environments by preference-based fuzzy behaviours,” *Robotics and Autonomous Systems* 56(3)(2008): 231-246.

A. Sgorbissa, R. Zaccaria, “Planning and obstacle avoidance in mobile robotics”, *Robotics and Autonomous Systems* 60(4)(2008): 628-638.

R. Szlapczynski, J. Szlapczynska, “On evolutionary computing in multi-ship trajectory planning,” *Applied Intelligence* 37(2)(2012): 155-174.

R. Tiwari, A. Shukla, R. Kala, *Intelligent Planning for Mobile Robotics: Algorithmic Approaches* (Hershey, PA: IGI Global Publishers, 2013).

M. Wang, T. Wu, “Cooperative co-evolution based distributed path planning of multiple mobile robots,” *Journal of Zhejiang University - Science A* 6(7)(2005): 697–706.

E. K. Xidias, P. N. Azariadis, “Mission design for a group of autonomous guided vehicles,” *Robotics and Autonomous Systems* 59(1)(2011): 34-43.