Motion Planning Using a Memetic Evolution Algorithm for Swarm Robots

Regular Paper

Chien-Chou Lin*, Kun-Cheng Chen and Wei-Ju Chuang

Department of Computer Science and Information Engineering, National Yunlin University of Science and Technology, R.O.C.
* Corresponding author E-mail: linchien@yuntech.edu.tw

Received 09 Sep 2011; Accepted 12 Mar 2012

© 2012 Lin et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract A hierarchical memetic algorithm (MA) is proposed for the path planning and formation control of swarm robots. The proposed algorithm consists of a global path planner (GPP) and a local motion planner (LMP). The GPP plans a trajectory within the Voronoi diagram (VD) of the free space. An MA with a non-random initial population plans a series of configurations along the path given by the former stage. The MA locally adjusts the robot positions to search for better fitness along the gradient direction of the distance between the swarm robots and the intermediate goals (IGs). Once the optimal configuration is obtained, the best chromosomes are reserved as the initial population for the next generation. Since the proposed MA has a non-random initial population and local searching, it is more efficient and the planned path is faster compared to a traditional genetic algorithm (GA). The simulation results show that the proposed algorithm works well in terms of path smoothness and computation efficiency.

Keywords genetic algorithm, memetic algorithm, hierarchical, local motion planner, Voronoi diagram, formation control, swarm robots

1. Introduction

In recent years an increasing number of multi-robot systems have been proposed. Swarm robotics (Yogeswaran and Ponnambalam, 2010; Samitha and Purbudu, 2009; Winfield et al., 2008) is an approach for coordinating multi-robot systems. The swarm shares information about the environment and individual members interact with each other. Cooperative behaviour may be used to complete a task.

Most studies on robot swarm cooperation have focused on formation control, which refers to the task of controlling a group of mobile robots to follow a predefined path or trajectory while maintaining the desired formation pattern, as shown in Fig. 1. Numerous methods have been proposed for formation control, which can be roughly categorized into four basic approaches: behavioural, virtual structure, leader-follower and potential field.

In virtual structure approaches, the robot swarm is considered as a single rigid robot and a rigid geometric relationship among group members is maintained (Beard et al., 2001; Belta and Kumar, 2004; Eren et al., 2002; Lewis...
and Tan, 1997; Ren and Beard, 2004; Zhang et al., 2009). Therefore, the path planning of a robot swarm can be simplified as the path planning of a rigid robot (Lewis and Tan, 1997). The advantage of the virtual structure approach is ease of implementation, however, the approach has low path planning flexibility. Virtual structure approaches were proposed for coordinating multiple spacecraft (Beard et al., 2001; Ren and Beard, 2004). In Bela and Kumar (2004), the algorithm derives scale-parameterized interpolated trajectories for a team of fully actuated mobile robots. The scale parameter controls the distances between robots and minimizes the overall energy consumption due to motion.

For behaviour-based approaches (Balch and Arkin, 1998; Baldassarre et al., 2007; Gazi, 2005; Lawton et al., 2003; Monteiro and Bicho, 2002; Parker, 1998; Veloso et al., 1999; Chen and Luh, 1994; Schneider-Fontan and Mataric, 1998; Sugihara and Suzuki, 1996), several desired behaviours, i.e., movement towards the goal, obstacle avoidance, collision avoidance and keeping formation, are defined for each robot to create its trajectory. The planning of robots can be done concurrently. Since each robot is considered individually, it is difficult to guarantee precise formation control. In Lawton et al. (2003), formation control comprised a sequence of manoeuvres between formation patterns. The algorithm consists of three strategies: (i) using relative position information configured in a bi-directional ring topology to maintain the formation, (ii) injecting inter-robot damping via passivity techniques and (iii) accounting for actuator saturation.

In leader-follower approaches (Burgard et al., 2005; Egerstedt and Xiaoming, 2001; Mariottini et al., 2007; Shao and Xie, 2007; Klocker and Hutchinson, 2006; Moreau, 2005; Olfati-Saber, 2006; Tanner et al., 2004; Vig and Adams, 2006; Ren and Beard, 2005; Chen and Wang, 2005), the ability of a robot depends on its job. In the swarm, one or a few robots act as leaders which move along predetermined trajectories and other robots in the group follow while maintaining the desired relative position with respect to the leader. Generally, leader-follower-based robot systems are implemented as centralized systems. However, most leader-follower approaches are not complete algorithms because the safe path, that which gives a robot sufficient distance from obstacles and other robots, is difficult to derive. In Mariottini et al. (2007), the centralized leader-follower formation control algorithm uses panoramic vision to maintain the swarm formation.

In potential-based approaches (Gazi and Passino, 2003; Khatib, 1985; Liu et al., 2000; Yun and Tan, 1997; Zhu et al., 2009), the formation of the robot swarm is modelled by a potential field. In Barnes et al. (2009), artificial potential fields were generated from normal and sigmoid functions. Due to the potential fields, the motion of swarm members and the individual member spacing are limited. In Ge and Fua (2005), queues and formation vertices represented the desired formation pattern. Artificial potential trenches represented the desired pattern and trajectory for the group of robots. In Balch and Hybinette (2000), a set of artificial points was used as beacons that guide the robots to their goal. The approach uses the geometric relationship between beacon points to move the robots in formation. Other examples of potential field approaches can be found in Balch and Hybinette (2000), Elkaim and Kelbley (2006), and Gazi (2005). Pereira et al. (2008) proposed a navigation function with a Lyapunov stable function. Lyapunov functions are used to prove closed-loop stability and to solve the local minima problem of potential fields.

In general there are some different optimal constraints in the path planning problem, e.g., time-optimal, the shortest path and path safety. In this paper, the optimization of path safety is considered.

In order to obtain a safe path for swarm robots, the present paper proposes a hierarchical path planning algorithm. The proposed algorithm consists of a global path planner (GPP) and a local motion planner (LMP). The global path planning algorithm searches for a path, which the centre of the robot swarm should move along, within a Voronoi diagram of the free space. The motion planner is a genetic algorithm based on an artificial potential field. The potential functions are used to keep robots away from obstacles and to keep the robot swarm within a certain distance from each other. With the potential models, the local motion paths derived by the memetic algorithm are safe in terms of being collision-free and away from obstacles. Therefore, the safety of the obtained path consists of a safe global path and safe local motion. The global path derived from a Voronoi diagram is an optimal path in terms of safety because the path is maximally clear of obstacles. As for the local motion, the proposed MA generates an optimal local motion which takes account of the distance to IG, the distance to obstacle and the connectivity of robots.
While some Lyapunov-based formation control algorithms (Pereira et al., 2008) usually use stability to guide robots to their desired configurations, the stability of a GA-based algorithm is defined as the best fitness value of the chromosome when the change of the best fitness value of the chromosome is negligible. In this paper, the optimal configuration of robot swarm is obtained when the evolution reaches a stability state.

The rest of this paper is organized as follows. The GPP and the LMP of the proposed hierarchical path planning algorithm for swarm robots are introduced in Section 2 and 3, respectively. In Section 4, simulation results are given. Finally, conclusions and suggestions for future work are given in Section 5.

2. Global Path Planning

Due to the coupling of global planning and local planning, many existing algorithms require complex search algorithms and can become trapped at local minima. The proposed algorithm consists of a GPP and an LMP. The former determines the primary movement direction of the swarm robots and the latter derives a frame of robot configurations. Details of the two planning algorithms are introduced in the next section.

![Figure 2. Simple path of moving straight from start to goal.](image)

Global path planning can be considered as a planning problem for a point robot. In Fig. 2, a swarm of two robots moves to the goal configuration; the planned path is close to obstacles (Erinc and Carpin, 2007; Ghanbari and Noorani, 2011). In order to obtain a safe path, a Voronoi diagram (VD) is adopted since it is easy to implement and has been shown to work well in many cases. There are many variants of VDs (Choset et al., 1999; Erinc and Carpin, 2007; Kloetzer and Belta, 2007; Wein et al., 2005). In the present study a VD consisting of line segments (Choset et al., 1999) is considered. The VD shows a set of free points which are equidistant to the two closest obstacles. In Wein et al. (2005), the VD was constructed using Voronoi vertices and Voronoi arcs. The Voronoi vertices in this case are points equidistant to the closest features of three (or more) polygons. The vertices are connected by continuous chains of Voronoi arcs. An arc may be equidistant to the two closest vertices or to the two closest obstacle edges or to an obstacle vertex and an obstacle edge.

As shown in Fig. 3, all edges and vertices of obstacles are used to construct the VD. The computation complexity is proportional to the total number of features of obstacles. Only a partial VD is used for global path planning for swarm robots. An efficient approach for constructing the partial VD is proposed in this paper.

Unlike approaches which construct the whole Voronoi diagram of the free space and then search for the path, the proposed scheme constructs a partial VD of the region of interest. As shown in Fig. 4(a), the proposed approach explores Voronoi vertices constructed from obstacles which are near the straight line from start to goal. Then, the Voronoi vertices are connected by a Voronoi arc which is formed by the nearest edges along the line, as shown in Fig. 4(b). The approach significantly reduces the computation complexity. Since a VD is the medial axis of the free space, the global path derived using a VD for swarm robots is the safest path.

![Figure 3. Voronoi graph for Fig. 2.](image)

![Figure 4. Voronoi vertices constructed using obstacles which are near the straight line from start to goal. The vertices are connected by a Voronoi arc which is formed by the nearest edges along the line.](image)
3. Local Motion Planning

The global path obtained in the previous stage can be sampled as a series of positions, denoted as \((q_1, q_2, q_3, ..., q_t)\), which the centre of the swarm robots should follow. These positions can be considered as the intermediate goals (IGs). For each position \(q_i\), the memetic algorithm (MA)-based local motion planner plans a set of configurations for the robots to which the centre of swarm is fixed at point \(q_i\).

![Flow chart of proposed memetic algorithm](image)

**Figure 5.** Flow chart of proposed memetic algorithm.

Basically, the operators, mutation and crossover of GA can avoid becoming trapped in local minima. The MA, a modified GA, uses a local search scheme for searching optimal solutions. Thus, the proposed MA-based planner is faster than the traditional GA-based planner without the local minima issue. The flow chart of the proposed memetic algorithm is shown in Fig. 5. The algorithm is:

**Potential-based Memetic Algorithm**

Begin  
\( i = 1 \); /* Initialize the first intermediate goal */  
\( t = 0 \); /* Initialize the evolutionary generations */  
Randomly generate an initial population \( P_i \) (\( t = 1 \));  
fitness\( (P(t)) \);  
Repeat until (reach the final goal \( q_f \)) Do \( P_{i+1}(t) = P_i(t) \);  
Repeat until (reach the intermediate goal \( q_i \)) Do  
select \( P(t+1) \) from \( P_i(t) \);  
crossover\( (P(t+1)) \);  
mutate \( (P(t+1)) \);  
fitness\( (P(t+1)) \);  
apply FT Local Search to \( P(t+1) \)  
\( t = t + 1 \);  
end  
\( i = i + 1 \);  
end  
End

To apply an MA search for the optimal configurations for robots, the coordinates of the robot swarm are encoded into one chromosome. The configuration of \( k \) robots is defined as their displacements, denoted as \((x_i, y_i), (x_2, y_2), ..., (x_k, y_k)\). The proposed algorithm can be extended to 3-D workspaces without significant modification. For example, the gene of a robot can be represented as \((x, y, z)\).

A. **Initialization**

The population, \( P_i(0) \), of the first intermediate goal is generated randomly. The initial populations, \( P_i(0), i > 1 \), of other intermediate goals are partially obtained from the last generation of the preceding intermediate goal and are partially randomly generated. Since these initial populations are eugenic and inherit from ancestors, the evolution time is reduced.

B. **Natural Selection**

Natural selection is a genetic operator that chooses a chromosome from the current generation’s population for inclusion in the next generation’s population. Before making it into the next generation’s population, selected chromosomes may undergo crossover and/or mutation (depending upon the probability of crossover and mutation) in which case the offspring chromosome(s) are actually the ones that make it into the next generation’s population.

The aim of selection is to preserve optimal chromosomes and abandon suboptimal ones. Generally, selection is performed according to the fitness of every chromosome, where the fitness evaluation of the GA is an objective function for chromosomes. There are several types of selection, including roulette, tournament, best, random, and top percent. In the present study, the top percent scheme is adopted. The top 10 percent of the population is reserved as the next generation’s population and the others are selected randomly.

C. **Crossover and Mutation Operators**

Selection alone cannot generate any new chromosomes for the population. The reproduction operators, crossover and mutation, are used to generate new offspring for the next generation. Crossover is performed between two selected chromosomes, called parents, by exchanging parts of their genomes to form two new chromosomes, called offspring. The most popular types of crossover operations are one-point, two-point, uniform, and blending. In this paper, since the \( i \)-th gene of a chromosome represents the position of robot \( i \), the crossover operator exchanges similarly positioned genes of a pair of chromosomes.
For the mutation operator, an arbitrary bit in a genetic sequence is changed with a probability. The purpose of mutation in an evolutionary algorithm (EA) is as a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next, while attempting to avoid local minima.

### D. Fitness Function

In general selection is conducted according to the fitness of every chromosome, where the fitness evaluation of the GA is an objective function for chromosomes. The simple definition of a fitness function is the distance between the robots and the goal. Therefore, the fitness function of a configuration can be defined as:

\[
V_q = \sum_{i=1}^{k} D'_{iq}
\]

where \(D'_{iq}\) is the distance between robot \(i\) and the intermediate goal of the swarm centre, \(q\). The distance can be simplified as:

\[
D'_{iq} = \sqrt{\left(T_x - B_x\right)^2 + \left(T_y - B_y\right)^2}
\]

\[
= \left|T_x - B_x\right| + \left|T_y - B_y\right|
\]

where \((T_x, T_y)\) are the coordinates of the intermediate goal and \((B_x, B_y)\) are the coordinates of robot \(i\). In this paper, an optimal chromosome, a frame of configurations of swarm robots, should be collision-free and should reach the goal. In other words, a good chromosome should be close to the goal and not collide with obstacles. Motions that lead to collisions with obstacles or other robots should be removed. Thus, the fitness function is defined as:

\[
V_q = \left\{ \begin{array}{ll}
\sum_{i=1}^{k} D'_{iq} + \rho U_{rep}(q), & \text{if no collision} \\
V_{max}, & \text{if collision}
\end{array} \right.
\]

where \(\rho\) is a constant and \(U_{rep}(q)\) is the repulsive potential of swarm robots from obstacles. When a configuration collides with obstacles, the collision function, \(f_{coll}(q)\), is equal to \(V_{max}\), which is a penalty; otherwise, it is equal to 1. The potential \(U_{rep}(q)\) can be calculated analytically (Moreau, 2005; Schneider-Fontan and Mataric, 1998) as:

\[
U_{rep}(q) = \sum_{i=1}^{k} U_{rep}^{i}(q)
\]

where \(U_{rep}(q)\) is the repulsive potential of robot \(i\) from the nearest obstacle.

\[
U_{rep}^{i}(p) = \begin{cases} 
0.5 \left( \frac{1}{\text{Diss}(p)} - \frac{1}{Q^*} \right)^3, & \text{Diss}(p) \leq Q^* \\
0, & \text{Diss}(p) > Q^*
\end{cases}
\]

where \(Q^*\) is the minimum distance from obstacles and \(\eta\) is a gain of the repulsive gradient. \(\text{Diss}(i)\) is the distance between robot \(i\) and the closest obstacle.

For swarm cohesion, a robot in the swarm should keep a certain distance from the swarm centre and not stray far from other robots, as shown in Fig. 6. Thus, a spring function is adopted as a repulsive/attractive potential function in the fitness function. The fitness function can be rewritten as:

\[
V_q = \left\{ \begin{array}{ll}
\sum_{i=1}^{k} D'_{iq} + \rho U_{rep}(q) + \frac{1}{2} \kappa \sum_{i=1}^{k} X_i^2, & \text{if no collision} \\
V_{max}, & \text{if collision}
\end{array} \right.
\]

where

\[
X_i^2 = (l - l_i)^2
\]

where \(l\) is the distance between robot \(i\) and the nearest neighbour robot, and \(l\) is the safe distance which should be kept between robots. The desired formation of the swarm robots can be maintained using the spring function. Two weighting factors, \(\rho\) and \(\kappa\), are added to the repulsion and the spring function, respectively. The selection stage of the GA is performed according to the fitness function in (7).

### E. Local Search of a Memetic Algorithm using a Gradient between Swarm Robots and IGS

Conventional genetic algorithms (CGAs) do not have a fine-tuning (FT) process to get closer to optimal solutions. Unlike CGAs, an MA is an EA with a local search process to refine individuals. In this paper, the local research scheme is used to adjust the position of the centre of the robot swarm moving toward the IG. The top 20 percent of chromosomes are reserved for the next generation. The chromosomes are fine-tuned before the next evolution. Consider the position of the swarm centre of a chromosome as \(R^c = (R_x^c, R_y^c)\) and the current intermediate goal as \(IG_i = (IG_{ix}, IG_{iy})\). The best movement direction is defined as:

\[
\vec{p} = IG_i - R^c = (IG_{ix} - R_x^c, IG_{iy} - R_y^c)
\]

Therefore, the fine-tuning procedure of the genes of the chromosomes is defined as:

\[
gen_{i} = \begin{cases} 
gen_i + p, & \text{if } gen_i + p \leq gene_{max} \\
gene_{max}, & \text{if } gen_i + p > gene_{max}
\end{cases}
\]
4. Simulation Results

The proposed algorithm consists of the GPP and the LMP. The former was implemented using a modified Voronoi algorithm. The latter was implemented using the JGAP 3.3.4 full API, an open source Java-based GA and genetic programming package. The simulations were run on a PC with a Core2 Duo 2.83-GHz CPU running the Windows XP operating system. The population was 100 and the maximum number of generations was set to 120. The probabilities of mutation and crossover were both 10%. The safe distances, $l$ and $Q^*$, were set to 10 pixels and 2 pixels, respectively. The range of the genes was 30 to -30.

**A. Swarm Robot Path Planning by Genetic Algorithm and Memetic Algorithm**

**Case 1**
The first example is the 3-robot swarm shown in Fig. 7(a). There are 5 IGs and 3 obstacles in this case. The simulation took 8.609 seconds to plan a 10-configuration collision-free path. A similar simulation of the CGA is shown in Fig. 7(b). The CGA took 10.11 seconds to plan a 12-configuration path. The proposed algorithm is faster and more efficient.

**Case 2**
An 8-robot swarm example is shown in Fig. 8(a). There are 4 IGs and 2 obstacles in this case. The obtained path is straighter than that obtained using the CGA shown in Fig. 8(b). The simulation took 178.594 seconds to plan an 85-configuration collision-free path. A similar simulation of the CGA is shown in Fig. 8(b). The simulation took 877.938 seconds to plan a 272-configuration path.

**B. Comparison between CGA and MA motion planning**

According to the simulation results of the above four cases, the proposed algorithm is more efficient than CGA. Fig. 9 shows the distances from the centre of the swarm to IG1, IG2, IG3, IG4 and IG5 for the two algorithms, respectively. The proposed algorithm requires fewer generations than the CGA. In Fig. 9(a), peaks are caused by the random initialization of genes when the swarm reaches an IG and moves to the next IG.
5. Conclusion

A potential-based memetic algorithm was proposed for the motion planning of swarm robots. The proposed algorithm consists of a global path planner (GPP) and a local motion planner (LMP). The GPP searches for a path, which the swarm robots should follow, from the start to the goal within a Voronoi diagram of the workspace. The LMP is an MA-based planner based on a fine-tuning (FT) structure. The repulsion keeps robots away from obstacles and a spring function maintains the robot swarm within a certain distance from each other. The proposed MA has a non-random initial population and fine-tuning based local searching, which make it more efficient and faster than the traditional CGA.

Since the proposed approach is a hierarchical algorithm which plans the global path and local motion individually, the robot swarm moves toward the goal by sequentially traversing a sequence of IGs and intermediate IGs. Therefore, the robot swarm can avoid becoming trapped in local minima. Simulation results demonstrate that the proposed algorithm can plan collision-free paths for swarm robots.

In future works we will focus on smoothing the planned paths to reduce redundant movements, especially for 8‐robot swarms. Some constraints may be considered in the planning stage, e.g., a minimum curvature constraint. With this modification, the proposed algorithm should be more efficient.

6. Acknowledgments

This work was supported by the National Science Council of Taiwan under grants NSC 99-2220-E-224-007 and NSC 100-2218-E-224-009-MY3.

7. References


