Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Cooperative co-evolution based distributed path planning of multiple mobile robots*

WANG Mei (王 梅), WU Tie-jun (吴铁军)

(*Institute of Intelligent Systems & Decision Making, Zhejiang University, Hangzhou 310027, China*)

E-mail: mwang@iipc.zju.edu.cn; tjwu@iipc.zju.edu.cn

Received Aug. 28, 2004; revision accepted Dec. 2, 2004

Abstract: This paper proposes novel multiple-mobile-robot collision avoidance path planning based on cooperative co-evolution, which can be executed fully distributed and in parallel. A real valued co-evolutionary algorithm is developed to coordinate the movement of multiple robots in 2D world, avoiding C-space or grid net searching. The collision avoidance is achieved by cooperatively co-evolving segments of paths and the time interval to pass them. Methods for constraint handling, which are developed for evolutionary algorithm, make the path planning easier. The effectiveness of the algorithm is demonstrated on a number of 2D path planning problems.

Key words: Cooperative co-evolution, Multiple mobile robot, Cooperative collision avoidance, Path planning
doi:10.1631/jzus.2005.A0697 **Document code:** A **CLC number:** TP242.6

INTRODUCTION

In recent years, multi-robot cooperation has become a hot focus in robotics research (Cao *et al*., 1995). One most important aspect in multi-robot cooperation is the space sharing problem that has been studied primarily via multiple-robot path planning, collision and deadlock avoidance problems (Fujii *et al*., 1998; Fujimori and Tani, 2002).

A number of approaches had been proposed to address this problem. Latombe (1991) classified path planning into decoupled planning and centralized planning (Fischer *et al*., 1996). Two approaches, the prioritized planning method and the path coordination method, were suggested for decoupled planning. In prioritized planning, individuals plan their own paths separately, and then adjust them according to defined priorities (Alami, 1996). The path coordination method plans paths by scheduling the configuration space-time resource (Bennewitz *et al*., 2001).

Multi-robot path planning is a large-scale combinatorial optimization problem involving complicated constraints. It is extremely difficult to find optimal solutions within the polynomial time. Evolutionary algorithms (e.g., genetic algorithms (GAs)) have been proposed as a heuristic method for solving problems of this sort. Early evolutionary computation based path planners often used standard evolutionary algorithms and searched paths in discrete maps (Shibata and Fukuda, 1993). They were inflexible and were not adaptive to changes or uncertainties. Evolutionary planners combine the concept of evolutionary computation with problem-specific chromosome structures and operators (Xiao *et al*., 1997; Smierzchalski and Michalewicz, 2000), and search in original and continuous environment to generate paths, which can be made suitable for dynamic environments.

Several co-evolutionary approaches were developed for robot collision avoidance. Berlanga *et al*.(2000) designed a feed-forward neural network based controller to solve a robotic autonomous navigation problem via uniform co-evolution. Sim *et al*.

*Project (No.2002CB312200) supported by the National Basic Research Program (973) of China

(2001) developed a schema co-evolutionary algorithm to extract optimal fuzzy rules for robots. These methods deal with collision avoidance only in a local view, and thus cannot cope with deadlock problems often encountered in multi-robotic navigation. Zheng *et al*.(2002) presented a global path planning method for multiple unmanned air vehicles navigation by introducing cooperative co-evolution. But subpopulations of potential paths are restricted to the same type, and must evolve synchronously in their evolutionary generations, and the benefits and capability of the co-evolutionary technique cannot be fully exploited.

In cooperative co-evolution developed by Potter (1997), the species are represented as several populations. Before finding a satisfactory solution, all the species are evolved separately, just as that in a general evolutionary algorithm. But individuals are not evaluated in isolation. The fitness of each member of each species is evaluated by forming collaborations with sample individuals from other species, thus the individuals will ultimately be judged on how well they work together to solve the target problem. The fitness is assigned strictly to the individual being evaluated and is not shared with the representatives from the other species that participated in the collaboration. Since there are many possible methods for forming representatives, the common one is to select the current best individual from each species.

In this paper we present a collision avoidance path planning method based on cooperative co-evolution, which can be executed fully distributed. Multiple populations are formed, with each population consisting of potential paths of one robot. The fitness of individuals in one population is evaluated by forming collaboration with representatives from other populations. The distances between robots and obstacles are introduced into fitness evaluation to drive the robots away from the obstacles. The algorithms conducting evolution of those populations can be different, and are executed asynchronously and in parallel. Problem-specific chromosomes are designed to avoid time-exhausting C-space search and dead-lock problems. The optimal path of each robot will be the result of the co-evolution in all populations.

The rest of this paper is organized as follows. In the next section, we will begin with an introduction to the workspace and path representation of the multi-robot path planning problem under study. In Section 3, the co-evolution for cooperative path planning is described in detail, including chromosome presentation, fitness evaluation, evolutionary operations and cooperative co-evolutionary process. The performance of the proposed method is then tested in a 5-robot case study in Section 4, followed by our conclusions in Section 5.

PROBLEM DESCRIPTION

Autonomous vehicles and hazardous waste cleanup robots are good examples of distributed multi-robot systems. They are distributed in a large area, and the communication is limited and can be performed only at a certain distance. Some of them may encounter each other in a small common space by chance. It is difficult to build a powerful centralized control unit for this situation. And it is impossible to plan paths for all situations in advance. Then, a fully distributed path-planning algorithm is preferable. In the system, the robots are treated equally, and no supervisor and priority exist. Only the robots nearby take part in the cooperative path planning for collision avoidance.

For the convenience of expression, suppose that there are N mobile robots (r_1, r_2, …, r_N) moving in a two-dimensional workspace. These robots may be different types and have different tasks to accomplish. Suppose also that there are M obstacles (O_1, O_2, …, O_M) in the workspace, presented as circles spreading around the workspace. The center and radius of obstacle O_j, is denoted by $(O_{c,j}, O_{r,j})$, j=1,2,…,M.

A path of robot r_i (i=1,2,…,N), is defined as a 2-tuple $\boldsymbol{P}_i=\{\boldsymbol{X}_i,\boldsymbol{T}_i\}$ where $\boldsymbol{X}_i=\{x_{i,1}, x_{i,2}, \ldots, x_{i,L_i}\}$ is a sequence of L_i positions (or, nodes) the robot must traverse, and $\boldsymbol{T}_i=\{\tau_{i,1},\tau_{i,2},\ldots,\tau_{i,L_i-1}\}$, $\tau_{i,j}<\tau_{i,j+1}$, is a sequence of the transit time at which the robot passes through those nodes correspondingly. For simplicity the path segment from $x_{i,j}$ to $x_{i,j+1}$ is denoted by $s_{i,j}$, and the corresponding time interval in which the robot transits $s_{i,j}$ is denoted by $t_{i,j}=[\tau_{i,j}, \tau_{i,j+1}]$. Please notice that the number of nodes, L_i, in a path of robot r_i is the result of path planning, and normally different from those of other robots.

The goal of path planning considered in this paper is to find the shortest and fastest way for the robots, under the constraint that there is no collision with any obstacle, and no collision among the robots. The movement should be conducted within the speed limitations.

The collision between robot r_i and obstacle O_j can be expressed as the event that the distance between the center $O_{c,j}$ of the obstacle and a path segment $s_{i,k}$ for some k in the path $\boldsymbol{P}_i$ of the robot is less than the radius $O_{r,j}$ of the obstacle. A function C_1 is defined in the following to describe this type of collision.

$$C_1(\boldsymbol{P}_i)=\begin{cases}1, & \exists j \text{ and } k,\ \delta_1(s_{i,k},O_{c,j})\le O_{r,j};\\ 0, & \text{otherwise.}\end{cases} \tag{1}$$

where the function $\delta_1(\cdot,\cdot)$ is used to compute the distance between a line segment and a node in the workplace of the robot system. C_1=1 means that robot r_i will collide with at least one obstacle in its motion.

There are two conjunctive conditions under which robot r_i and robot r_j will collide with each other, i.e., (i) a path segment of robot r_i, say, $s_{i,k}$, intersects a path segment of robot r_j, say, $s_{j,l}$; and (ii) the time interval $t_{i,k}$ in which robot r_i traverses the path segment $s_{i,k}$ overlaps the time interval $t_{j,l}$ in which robot r_j traverses the path segment $s_{j,l}$. A function C_2 is defined for this type of collision as follows.

$$C_2(\boldsymbol{P}_i,\boldsymbol{P}_j)=\begin{cases}1, & \delta_2(s_{i,k},s_{j,l})=0\wedge(\tau^{(s)}\in t_{i,k}\wedge\tau^{(s)}\in t_{j,l}),\\ & \qquad\exists k,l,\tau^{(s)};\\ 0, & \text{otherwise;}\end{cases} \tag{2}$$

where the function $\delta_2(\cdot,\cdot)$ returns the distance between two segments as its arguments (a naught distance indicates an intersection of the segments), and $\tau^{(s)}\in t_{i,k}\wedge\tau^{(s)}\in t_{j,l}$ implies that collision will occur when robot r_i and robot r_j walk on the intersected path segments, respectively, "at the same time $\tau^{(s)}$", in consideration of the volumes of those robots. C_2=1 implies that robot r_i and robot r_j will collide each other in their motion.

With the collision constraints defined above, the multi-mobile-robot path planning problem can be mathematically expressed as

$$\min_{(P_1,P_2,\cdots,P_N)} J=\sum_{i=1}^{N}\|\boldsymbol{P}_i\| \tag{3}$$

subject to

$$C_1(\boldsymbol{P}_i)=0; \tag{4}$$
$$C_2(\boldsymbol{P}_i,\boldsymbol{P}_j)=0; \tag{5}$$
$$v_{i,j}\le v_{i,\max}; \tag{6}$$
$$x_{i,1}=x_{i,s}; \tag{7}$$
$$x_{i,L_i}=x_{i,e}; \tag{8}$$
$$i,j=1,2,\ldots,N;$$

where in Eq.(3) the norm $\|\boldsymbol{P}_i\|$ is defined by

$$\|\boldsymbol{P}_i\|\triangleq\sum_{j=1}^{L_i-1}\left(\|s_{i,j}\|+\|t_{i,j}\|\right) \tag{9}$$

to evaluate the optimality of robot r_i path in terms of the length of the path, $\sum_{j=1}^{L_i-1}\|s_{i,j}\|$, and the time taken by the robot to travel on the path, $\sum_{j=1}^{L_i-1}\|t_{i,j}\|$. $v_{i,j}=\|s_{i,j}\|/\|t_{i,j}\|$ and $v_{i,\max}$ in Eq.(6) are the mean speed and maximal allowable speed of robot r_i traveling on the segment $s_{i,j}$, respectively. In Eqs.(7) and (8) the start node $x_{i,s}$ and the end node $x_{i,e}$ of robot r_i are supposed to be given a priori .

From Eqs.(3)–(8) it can be seen that, to minimize the objective J without any robot-to-obstacle collision and robot-to-robot collision, and without violating the robotic speed limits, the midway node $x_{i,j}\in\boldsymbol{X}_i$ of a robotic path, and the time $\tau_{i,j}\in\boldsymbol{T}_i$ at which the robot reaches the node $x_{i,j}$, for i=1,2,…,N and j=1,2,…,L_i, must be adjusted. It is not an easy task like that at a glance since this adjustment may cause robot-to-obstacle or robot-to-robot collision in the workplace. Traditional optimization techniques such as gradient-based search or dynamic programming are not good choices for this problem because the collision constraints Eqs.(4) and (5) are very complicated, and the total number of the adjustable variables, i.e., $L=2\sum_{i=1}^{N}L_i-3N$, is not fixed before problem solving.

MULTI-PATH PLANNING VIA COOPERATIVE CO-EVOLUTION

Solution representation and initialization

A solution of the problem described in Eqs.(3)–(8) is composed of a set of paths, each of which is assigned to a robot. The actual path of robot r_i (i=1, 2, …, N), will be the result of co-evolution in population Pop_i.

A real-valued problem-specific chromosome representation is used here. In population Pop_i, there are N_i path candidates from which the best one will be selected for the robot. The jth path candidate $\boldsymbol{P}_{i,j}$ of Pop_i is encoded with a chromosome composed of a sequence of genes in the following form

$$\boldsymbol{P}_{i,j} \triangleq \left\{ g_{i,j}^{(1)}, g_{i,j}^{(2)}, \cdots, g_{i,j}^{(L_{i,j}-1)} \right\} \in Pop_i \tag{10}$$

where the kth gene $g_{i,j}^{(k)} \triangleq \left(x_{i,k}^{(j)}, \left\| t_{i,k}^{(j)} \right\| \right)$, standing for the kth node in the jth path candidate of robot r_i and the length of time interval $t_{i,k}^{(j)}$ in which the robot traverse the path segment from $x_{i,k}^{(j)}$ to $x_{i,k+1}^{(j)}$. It is prescribed that the start and the terminal node for all path candidates are fixed, according to the task requirement of robot r_i. We note that the length $L_{i,j}$ of chromosome $\boldsymbol{P}_{i,j}$ may be different from that of another chromosome, say, $\boldsymbol{P}_{i,r}$, $r \neq j$, and is changeable during population evolution. In other words, very different from traditional evolutionary algorithms such as genetic algorithms, the chromosomes in a population are normally uneven in length. The structure of chromosome $\boldsymbol{P}_{i,j}$ is illustrated in Fig.1.

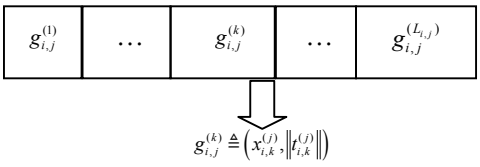

Fig.1 The structure of chromosome $\boldsymbol{P}_{i,j}$ as the representation of a path candidate of robot r_i

The initial individuals in each population, corresponding to a set of initial path candidates of a mobile robot, are constructed randomly. That is to say, in this stage a path candidate of a mobile robot contains a random number of nodes at random positions in the workplace except the start nodes and the target nodes, and random lengths of time intervals for the path segments connecting those nodes.

Solution evaluation via cooperative co-evolution

Solution evaluation by rating individuals in terms of their fitness plays a key role in the cooperative co-evolutionary path planning algorithm proposed in this paper. Three factors are taken into account in the design of a fitness function for evaluating a robot path: the lengths of the path, the time interval taken by a robot to travel on the path, and the constraints on robot speed and collision avoidance. In the last aspect, the robot-to-robot collision avoidance has to be accomplished through the coordination of the solution evaluation processes in different populations because it is concerned with the relationship of the paths of different robots. The fitness function to evaluate the path candidate $\boldsymbol{P}_{i,j}$ defined in Eq.(10) is therefore formulated as

$$\Phi(\boldsymbol{P}_{i,j}) = \mathrm{e}^{-\left[w_l \left\| \boldsymbol{P}_{i,j} \right\| + w_c \psi(\boldsymbol{P}_{i,j}) \right]} \tag{11}$$

where the non-increasing fitness function Φ evaluates the individual $\boldsymbol{P}_{i,j}$ in terms of the optimality of $\boldsymbol{P}_{i,j}$, defined in Eq.(9), and the value of the penalty function $\psi(\boldsymbol{P}_{i,j})$ which is designed to return a large number if $\boldsymbol{P}_{i,j}$ violates any of the constraints Eqs.(4), (5) or (6), or zero if none of the constraints is violated, in order to keep the solution candidate $\boldsymbol{P}_{i,j}$ feasible.

Theoretically, the simplest way to maintain the feasibility of a population in an evolutionary process is to check up whether any individual in the population violates system constraints, and, if it does, just kick it out from the population. But for some problems, the change was not as stable as others. And sometimes, there may be some difficulty in locating a feasible solution first.

Many methods for constraint handling have been proposed for evolutionary algorithms (Michalewicz and Schoenauer, 1996), including methods based on penalty functions (Michalewicz and Attia, 1994), preservation of feasibility (Schoenauer and Michalewicz, 1996), decoders (Koziel and Michalewicz, 1999), distinction between feasible and infeasible solutions (Schoenauer and Xanthakis, 1993), as well as some hybrid techniques (Myung *et al*., 1995). Method of preservation of feasibility requires the

design of specialized operators. Method of decoder is to map an N-dimensional into the feasible search space.

Due to their simplicity in concept and their easy application, methods based on penalty functions are the most popular ones. Usually, the penalty function is based on the distance of a solution from the feasible region or on the effort to "repair" the solution (i.e., to force it into feasible region). The former is preferable. The penalty function, constructed here, measures the constraint violation (e.g. the squared or absolute constraint violation).

These methods differ in many important details, such as how the penalty function is designed and applied to infeasible solutions. Here we incorporate the knowledge about problem-specific constraints into penalty functions. In potential field based path planning methods (Salichs and Moreno, 2000; Koren and Borenstein, 1991), a continuous potential field gives a good indication of the distances to obstacles and the shapes of obstacles so that necessary changes in robot position can be done to keep them away from the obstacles. Getting some idea from it, we introduce the penalty of distance from obstacle edges into the algorithm. Bad solutions, with the path segments only "scraping" the verge of the obstacles, are better than those which run into the obstacles deeply. What is more, they can be easily modified by crossover and mutation operators discussed in Section 3.3. Thus, they should have more chance of surviving in the population than the latter ones. The term ψ_2 is set for this purpose.

Three kinds of constraint-violation events will be punished in the penalty function ψ, i.e., overspeed of robot, robot-to-obstacle collision and robot-to-robot collision. Thus

$$\psi(\boldsymbol{P}_{i,j}) \triangleq \psi_1(\boldsymbol{P}_{i,j}) + \sum_{l=1,l\neq i}^{M} \psi_2(\boldsymbol{P}_{i,j}, O_l) + \sum_{l=1,l\neq i}^{N} \psi_3(\boldsymbol{P}_{i,j}, \boldsymbol{P}_l^*) \tag{12}$$

where ψ_1 is defined for the punishment of robot overspeed, i.e.,

$$\psi_1(\boldsymbol{P}_{i,j}) = \begin{cases} \sum_{k^*} s_{i,k^*}^{(j)} \Big/ \left\| t_{i,k^*}^{(j)} \right\|, & \exists k^*,\ s_{i,k^*}^{(j)} \Big/ \left\| t_{i,k^*}^{(j)} \right\| > v_{i,\max}; \\ 0, & \text{otherwise}; \end{cases} \tag{13}$$

ψ_2 is introduced to punish robot-to-obstacle collision, defined by

$$\psi_2(\boldsymbol{P}_{i,j}, O_l) = \begin{cases} \sum_{k^*} \delta_1^{-1}(s_{i,k}^{(j)}, O_{c,l}), & \exists k^*,\ \delta_1\left(s_{i,k^*}^{(j)}, O_{c,l}\right) \le O_{r,l}; \\ 0, & \text{otherwise}; \end{cases} \tag{14}$$

and ψ_3 is the penalty for robot-to-robot collision, computed by

$$\psi_3(\boldsymbol{P}_{i,j}, \boldsymbol{P}_l^*) = \begin{cases} 1, & \exists s_{l,r} \in P_l^*,\ \delta_2\left(s_{i,k}^{(k)}, s_{l,r}\right) = 0; \\ 0, & \text{otherwise}. \end{cases} \tag{15}$$

where $\boldsymbol{P}_l^*$ is the "representative" of the population for robot r_l, l=1,2,…,N. Normally this representative is selected in terms of the best fitness value among the individuals in that population, i.e.,

$$\boldsymbol{P}_l^* = \arg\max_k \left\{ \Phi(\boldsymbol{P}_{l,k}) \right\} \tag{16}$$

From Eqs.(12)–(16) we note that the penalty function $\psi(\boldsymbol{P}_{i,j})$ will return zero if the path candidate $\boldsymbol{P}_{i,j}$ does not violate any speed and collision constraints. The weights w_l and w_c must be set carefully to embody the balance between the optimality of a robot path and the punishment for collision avoidance. Firstly, ψ_1 and ψ_2 were normalized by projecting ψ_1 from the real value $[O_{r,j}^{-1}, +\infty]$ to range [0, 1], and projecting ψ_2 from the real value [0, $2v_{i,\max}$ (or above)] to range [0, 1]. Then, the value of w_c are set larger than that of w_l, i.e. $w_c/w_l=10^{4\sim5}$ or so. The fitness Φ then has a greater value to indicate a better solution candidate.

Evolutionary operations

The solution representation shown in Fig.1 is specially designed to be conducive to the evolutionary operations such as crossover and mutation to yield better path candidates for the multi-mobile-robot path planning problem.

The gene based crossover exchanges several genes in two chromosomes, making the corresponding path candidates change their trajectories and time

intervals. For example, suppose $\boldsymbol{P}_{i,l}=\left\{g_{i,l}^{(1)},g_{i,l}^{(2)},\cdots,g_{i,l}^{(L_{i,l}-1)}\right\}$ and $\boldsymbol{P}_{i,s}=\left\{g_{i,s}^{(1)},g_{i,s}^{(2)},\cdots,g_{i,s}^{(L_{i,s}-1)}\right\}$ are selected to exchange at their kth genes, $g_{i,l}^{(k)}$ and $g_{i,s}^{(k)}$. According to the definition of gene intra-structure in Section 3.1, this crossover operation will yield two new path candidates, i.e., $\boldsymbol{P}'_{i,l}$ and $\boldsymbol{P}'_{i,s}$. Then the trajectory of $P_{i,j}$ will be changed from $X_{i,l}=\left\{x_{i,1}^{(l)},x_{i,2}^{(l)},\cdots,x_{i,k-1}^{(l)},x_{i,k}^{(l)},x_{i,k+1}^{(l)},\cdots,x_{i,L_{i,l}}^{(l)}\right\}$ to $X'_{i,l}=\left\{x_{i,1}^{(l)},x_{i,2}^{(l)},\cdots,x_{i,k-1}^{(l)},x_{i,k}^{(s)},x_{i,k+1}^{(s)},\cdots,x_{i,L_{i,l}}^{(s)}\right\}$, and the total time to traverse the trajectory from $\left\|t_{1,L_{i,l}}^{(i,l)}\right\|=\sum_{j=1}^{L_{i,l}-1}\left\|t_{i,j}^{(l)}\right\|$ to $\left\|t'^{(i,l)}_{1,L_{i,l}}\right\|=\sum_{j=1}^{k-1}\left\|t_{i,j}^{(l)}\right\|+\sum_{j=k}^{L_{i,l}-1}\left\|t_{i,j}^{(s)}\right\|$. The trajectory of $\boldsymbol{P}_{i,s}$ and the total time to traverse this trajectory will be also changed correspondingly. Thus, two individuals will locally modify their path trajectories and the total times to traverse those trajectories by one-point crossover. The position of chromosomes for gene swapping must be selected carefully because the lengths of the chromosomes may be uneven. In fact the maximal crossover position number is min($L_{i,l}$, $L_{i,s}$)−1. From this fact we see that there exists the possibility that a part of a path trajectory cannot be adjusted during the crossover operation. Fortunately, the lengths of chromosomes are changeable in an evolutionary process with the help of mutation operations, which makes the crossover operation effective in the schemata inheritance process (Goldberg, 1989).

In consideration of the characteristics of the multi-robot path planning problem, six problem-specific mutation operators (Zheng *et al*., 2002), denoted by μ_i (i=1,2,…,6), are suggested to maintain population diversity, i.e., μ_1: the perturbation mutator, imposing random change of gene parameters in a chromosome; μ_2: the intra-swap mutator, exchanging randomly selected adjacent genes except those containing start nodes or terminal nodes in a chromosome; μ_3: the swap mutator, splitting a chromosome into two parts randomly and then interchanging them except the start node and terminal node; μ_4: the insertion mutator, embedding a new gene into a chromosome randomly; μ_5: the deletion mutator, eliminating a gene in a chromosome randomly; μ_6: the smoothing mutator. This mutator is designed to smoothen a path trajectory by "cutting corners", see Fig.2. For example, suppose $\boldsymbol{P}_{i,l}$ is selected for the smoothing mutator at its kth gene. Then the trajectory of $\boldsymbol{P}_{i,l}$ will be changed from $X_{i,l}=\left\{x_{i,1}^{(l)},x_{i,2}^{(l)},\cdots,x_{i,k-1}^{(l)},x_{i,k}^{(l)},x_{i,k+1}^{(l)},\cdots,x_{i,L_{i,l}}^{(l)}\right\}$ to $X'_{i,l}=\left\{x_{i,1}^{(l)},x_{i,2}^{(l)},\cdots,x_{i,k-1}^{(l)},x_{i,k}^{(s1)},x_{i,k}^{(s2)},x_{i,k+1}^{(l)},\cdots,x_{i,L_{i,l}}^{(l)}\right\}$, where $x_{i,k}^{(s1)}$ is the midpoint of the segment $[x_{i,k-1}^{(l)},x_{i,k}^{(l)}]$ and $x_{i,k}^{(s2)}$ is the midpoint of the segment $[x_{i,k}^{(l)},x_{i,k+1}^{(l)}]$. The total number of the genes changes from $L_{i,l}$ to $L_{i,l}$+1 accordingly.

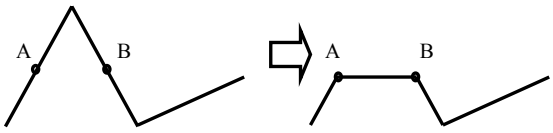

Fig.2 The smoothing mutator

The original path is shown on the left. A and B are the midpoints of the segments respectively. The new path is shown on the right, connects the midpoints to cut the corner of the edge and smooth the path selected

Among those mutation operators, the execution of μ_1, μ_2 or μ_3 changes the original gene parameters in a chromosome while keeping the total number of genes (i.e., the nodes in a path trajectory) unchanged. μ_4, μ_5 and μ_6 on the other hand change the total number of nodes in a path.

Selection strategy influences directly the convergence of a genetic algorithm. Actually, there are two different evolutionary processes in the proposed co-evolution algorithm. In a local evolutionary process, each population can have its own "local fitness evaluation" of which individual is the best, in consideration of local targets and constraints for this robot other than those defined in Eqs.(3)–(8). In this stage each population conducts its own evolutionary processes independently to meet the demands with respect to its robot type, task type and local environment, until a representative individual, normally the one with the best local fitness value so far, has been selected. After the representatives of all populations are available, a global evolutionary process is then launched. All the individuals in each population are globally evaluated one by one in terms of Eqs.(11)–(15) to meet the target defined in Eqs.(3)–(8). In this process each individual of a population will be checked with respect to all the representatives of other

populations except its own, in order to make sure that there is no robot-to-robot collision in the whole multi-robot system. Individuals with "globally" better fitness values [computed by Eq.(11)] will be selected to form a next generation, based on which a new round of "local evolutionary process" is initiated.

Global evolution process

As described in Section 3.3, in the co-evolution algorithm the collaboration among the populations of different robots takes place in an asynchronous way. The individuals in different populations can be evolved separately at the same time, and, if the mobile robots are heterogeneous, different evolutionary algorithms such as evolution strategies, evolutionary programming, genetic algorithms or genetic programming can be used. The only relationship among the individuals in different populations is the global fitness evaluation depicted in Section 3.2. The evolution of different populations can be processed asynchronously and in parallel, while the representatives of those populations should be selected synchronously.

The whole procedure of the cooperative co-evolutionary path planning algorithm is described in Fig.3.

EXPERIMENTS

Test

The effectiveness of the proposed multi-robot path planning algorithm was verified by several experiments. Two kinds of experiments were performed. In both experiments, 5 mobile robots and 5 obstacles are deployed in a 100×100 cm^2 rectangle workplace. All the obstacles are represented as circles. The maximum motion speed of all the robots is assumed to be 10 cm/s.

The cooperative co-evolutionary path planning algorithm can be executed fully distributed and parallel, which needs no supervisor. It is more suitable for real mobile system. The evolutionary process of each mobile path can be conducted in its own computer asynchronously and separately. The only communication among them is to sample representatives synchronously and periodically.

In heterogeneous robot system, the mobiles differ with each other in computing capacity and velocity constraints. So, each population has its own execution speed and fitness evaluation. The numbers of evolution generation executed in one cycle (period of sampling representatives) are varied accordingly. The radii of the mobiles are different, some of them cannot be considered as particles any more. Suppose that the radium RI_i of robot r_i is large enough that it cannot be ignored, ψ_2 in Eq.(14) has to be adjusted accordingly to

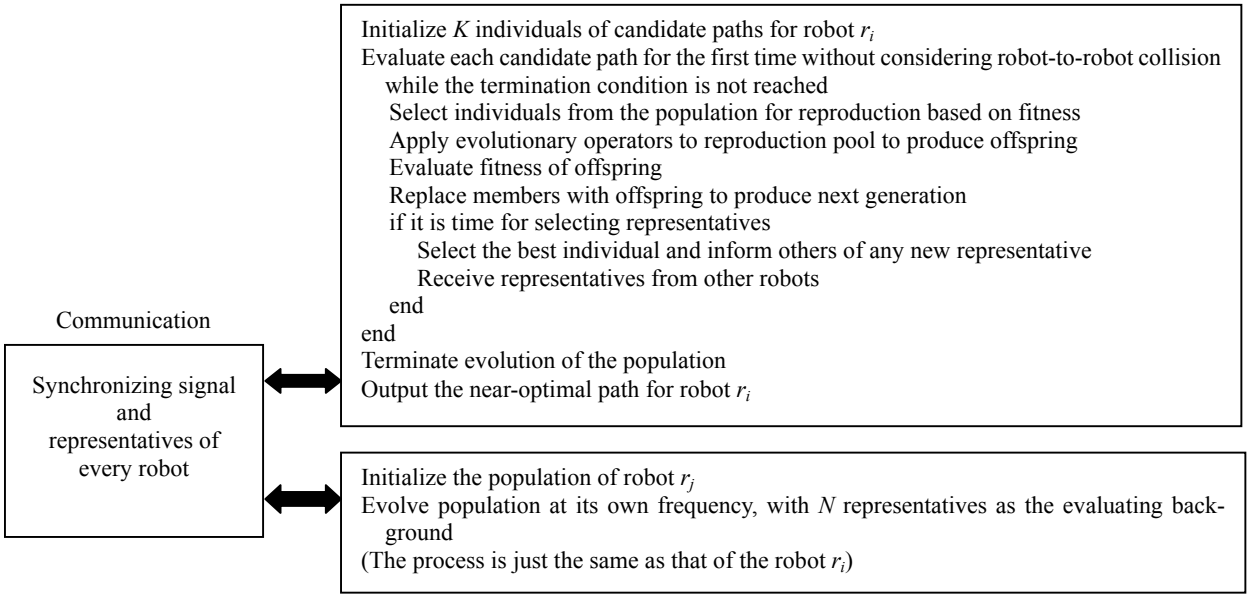

Fig.3 The cooperative co-evolutionary path planning algorithm

$$\psi_2(\boldsymbol{P}_{i,j}, O_l) = \begin{cases} \sum_{k^*} \delta_1^{-1}(s_{i,k}^{(j)}, O_{c,l}), & \exists k^*, \delta_1(s_{i,k^*}^{(j)}, O_{c,l}) \le O_{r,l} + RI_i; \\ 0, & \text{otherwise}; \end{cases}$$

The algorithm is flexible and can save more execution time than the serial ones.

Result

The centers and radii of the obstacles were first generated randomly. The starting and target points of the robots were randomly scattered out among the obstacles.

The algorithm solves "easy" problems rapidly and "difficult" problems systematically with a gradual increase of computation time. We measure task difficulty empirically by the complexity of the solution path, e.g. the number of monotone segment comprising the path and the clearance to obstacles along the path. In most cases, we can get the reasonable paths after we select representatives 16 times on average, and more than 50 times in some complicated cases (such as those in Figs.4c and 4d).

Two instances of obstacle deployment are illustrated in Figs.4a–4d. Fig.4a and Fig.4c show the paths of the robots in operational space respectively. The mobile robots move along the line segments, from Δ to *. Fig.4b and Fig.4d give the same paths in *X*-*Y*-time space. In order to show the situation clearly, the obstacles are ignored in Fig.4b and Fig.4d, and the two shorter paths of r_4 and r_5 are omitted in Fig.4b.

In Fig.4a, the path of r_1 intersects the paths of r_2 and r_3, and in Fig.4c, the path of r_2 intersects the paths of r_3 and r_5. But it can be seen that, in Fig.4b and Fig.4d, the robots do not go near to each other at any time.

It means that the mobile robots do not collide with each other. They cross the intersections at different time. We can see that the obtained paths are collision free and reasonable.

Secondly, a rather complicated situation, as shown in Figs.4c and 4d, was selected for a repeated experiment. It was conducted in the same environment during all the 200-times executing process; that is, the start and target points, as well as the obstacles configurations are constant. We terminate the execution after selecting representatives for 90 times. And in most cases, we can get the reasonable paths after we select representatives for 57 times on average. We get results easily, with 3% failure, with very few collisions at the obstacle edges. However, some times the results are stacked in sub-optima and three kinds of generalization results in the Fig.4c, Fig.4e and Fig.4f reflect these situations. Moreover, it can be seen that, though the paths in Fig.4c and Fig.4f are shorter, they have more risk of obstacle collision, compared with the results of Fig.4e. Table 1 shows the average lengths of the paths and the average time to transit them in 200 runs. It can be noticed that the evolution of the paths which can go directly from the start point to the target one without any collision, is faster.

Table 1 The average lengths of mobile robots paths

	Average length	Average time
r_1	23.364	8.0789
r_2	170.59	33.525
r_3	75.77	9.6155
r_4	35.249	8.8409
r_5	115.7	26.999

In the case of cooperative co-evolution, two main goals have been achieved. The results obtained are very encouraging.

CONCLUSION

Based on the preliminary experimental results, cooperative path planning based on cooperative co-evolution was shown to be a simple and effective method to realize the conflict-free motion of several mobile robots operating in a common workspace. Due to the separate evolution of different populations, parallel path computation of robot paths is permitted. As a heuristic method, collision penalty drive the robots away from each other and the obstacles.

Although co-evolutionary algorithm provides good solutions to combinatorial optimization problems, they are not combinatorial explosive like exhaustive search algorithms are. Another advantage of this algorithm is that the dead-lock situation, that bother some of the traditional methods, does not exist.

As a local search algorithm, the evolutionary algorithm does not guarantee the optimality of the solution but the near optimal one. Care should be

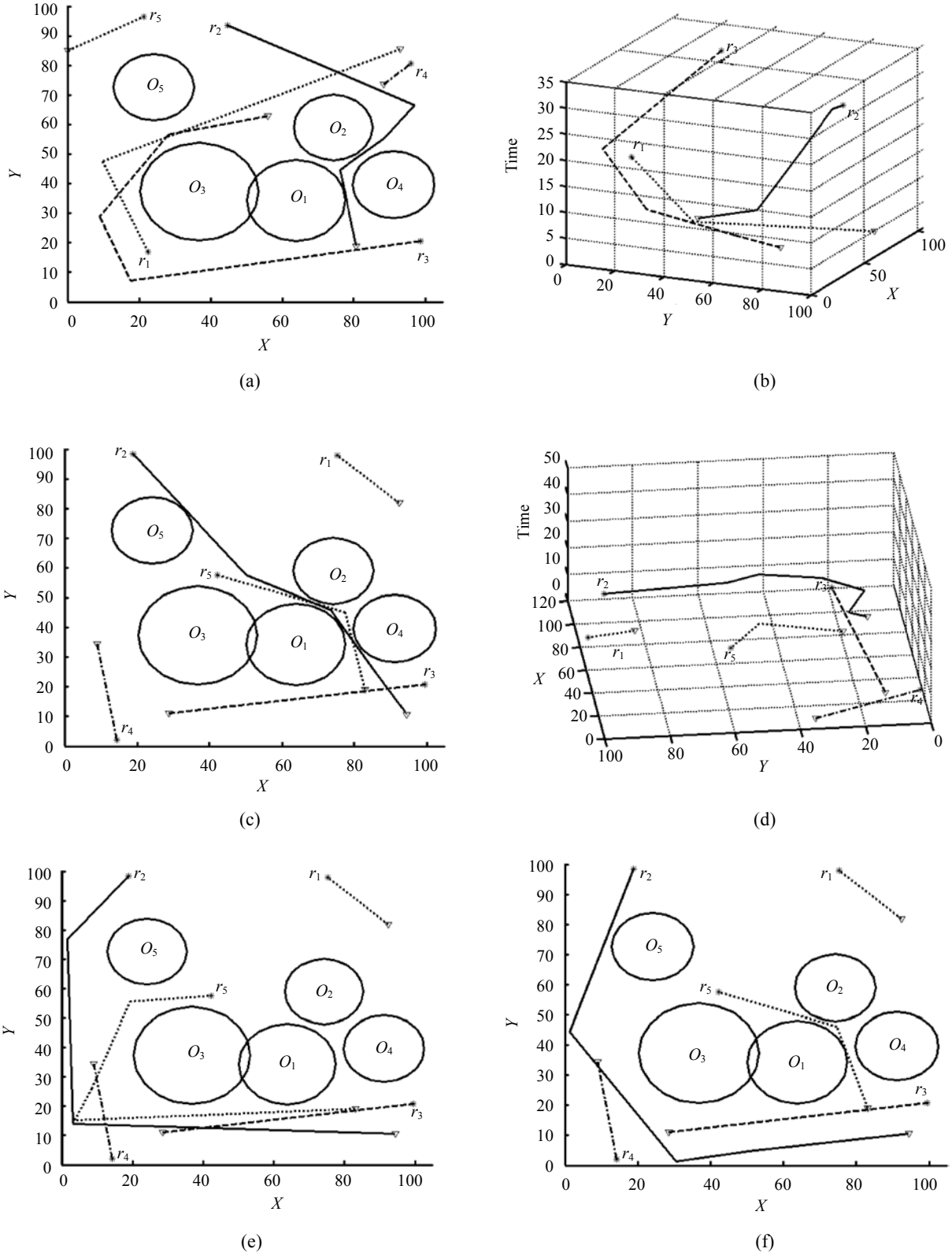

Fig.4 Results of path planning. (a), (c), (e) and (f) are paths illustrated in operational space, while (b) and (d) are paths in *X*-*Y*-time space, correspondingly; (e) and (f) are repeated experiments of instance in (c)

taken when applying these techniques to any specific problem. The main disadvantage is that, they may end up in a locally optimum in search space. While at the same time, a globally optimum solution is at some other point farther away from this onto a flat plateau. When all the neighboring points have the same value, the search loses its direction.

References

Alami, R., 1996. A Fleet of Autonomous and Cooperative Mobile Robots. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, **3**:1112-1117.

Berlanga, A., Isasi, P., Sanchis, A., Molina, J.M., 2000. Co-evolutive Adaptation of Fitness Landscape for Solving the Testing Problem. IEEE International Conference on Systems, Man and Cybernetics, p.3846-3851.

Bennewitz, M., Burgard, W., Thrun, S., 2001. Optimizing Schedules for Prioritized Path Planning of Multi-robot Systems. Proceedings of IEEE International Conference on Robotics and Automation, **1**:271-276.

Cao, Y.U., Fukunaga, A.S., Kahng, A.B., Meng, F., 1995. Cooperative Mobile Robotics: Antecedents and Directions. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, **1**:226-234.

Fischer, C., Buss, M., Schmidt, G., 1996. Hierarchical Supervisory Control of Service Robot Using Human-Robot Interface. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, p.1408-1416.

Fujii, T., Arai, Y., Asama, H., Endo, I., 1998. Multilayered Reinforcement Learning for Complicated Collision Avoidance Problems. Proceedings of IEEE International Conference on Robotics and Automation, p.2185-2191.

Fujimori, A., Tani, S., 2002. A Navigation of Mobile Robots with Collision Avoidance for Moving Obstacles. IEEE International Conference on Industrial Technology, **1**:1-6.

Goldberg, D.E., 1989. Genetic Algorithm in Search optimization and Machine Learning. Wesley Publishing Company.

Koren, Y., Borenstein, J., 1991. Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. Proceedings of IEEE International Conference on Robotics and Automation, **2**:1398-1404.

Koziel, S., Michalewicz, Z., 1999. Evolutionary algorithms, homomorphous mappings and constrained parameter optimization. *Evolutionary Computation*, **7**(1):19-44.

Latombe, J.C., 1991. Robot Motion Planning. Kluwer Academic, Boston.

Michalewicz, Z., Attia, N., 1994. Evolutionary Optimization of Constrained Problems. Proceedings of 3rd Conference on Evolutionary Programming, p.98-108.

Michalewicz, Z., Schoenauer, M., 1996. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, **4**(1):1-32.

Myung, H., Kim, J.H., Fogel, D., 1995. Preliminary Investigation on Constrained Problems. Proceedings of 4th Conference on Evolutionary Programming, p.449-463.

Potter, M.A., 1997. The Design and Analysis of a Computational Model of Cooperative Coevolution. Ph.D Thesis, George Mason University.

Salichs, M.A., Moreno, L., 2000. Navigation of mobile robots: open questions. *Robotica*, **18**:227-234.

Schoenauer, M., Michalewicz, Z., 1996. Evolutionary Computation at the Edge of Feasibility. Proceedings of 4th Conference on Parallel Problems Solving from Nature, **1141**:245-254.

Schoenauer, M., Xanthakis, S., 1993. Constrained GA Optimization. Proc. of 5th Int. Conf. on Genetic Algorithms, p.573-580.

Shibata, T., Fukuda, T., 1993. Intelligent Motion Planning by Genetic Algorithm with Fuzzy Critic. Proceeding of 8th IEEE Symp. on Intelligent Control, p.565-570.

Sim, K.B., Chun, H.B., Lee, D.W., 2001. Dynamic Behavior Control of Autonomous Mobile Robots Using Schema Co-evolutionary Algorithm. Proceedings of IEEE International Symposium on Industrial Electronics, **1**:560-565.

Smierzchalski, R., Michalewicz, Z., 2000. Modeling of ship trajectory in collision situations by an evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, **4**(3):227-241.

Xiao, J., Michalewicz, Z., Zhang, L., Trojanowski, K., 1997. Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation*, **1**(1):18-28.

Zheng, C.W., Ding, M.Y., Zhou, C.P., 2002. Cooperative Path Planning for Multiple Air Vehicles Using a Co-evolutionary Algorithm. Proceedings of International Conference on Machine Learning and Cybernetics, Beijing, **1**:219-224.