Analytical and Experimental Results on Multiagent Cooperative Behavior Evolution

Jiming Liu, *Senior Member, IEEE* Jianbing Wu, *Student Member, IEEE* Xun Lai

Autonomous Agents and Intelligent Systems Group
Department of Computer Science, Hong Kong Baptist University
Kowloon Tong, Hong Kong *jiming@comp.hkbu.edu.hk*

Abstract

This paper addresses the problem of automatically programming cooperative behaviors in a group of autonomous robots. The specific task that we consider here is for a group of distributed autonomous robots to cooperatively push an object toward a goal location. The difficulty of this task lies in that the task configurations of the robots with respect to the object do not follow any explicit, global control command, primarily due to certain modeling limitations as well as planning costs as in many real-life applications. In such a case, it is important that the individual robots locally modify their motion strategies, and at the same time, create a desirable collective interaction between the distributed robot group and the object that can successfully bring the object to the goal location. In order to solve this problem, we have developed an evolutionary computation approach in which no centralized modeling and control is involved except a high-level criterion for measuring the quality of robot task performance. The evolutionary approach to distributed robot behavioral programming is based on a fittest-preserved genetic algorithm that takes into account the current positions and orientations of the robots relative to the object and the goal, and a week global feedback on the collective task-performing effect in relation to the goal if some new local motion strategies are employed by the robots.

Keywords

Autonomous robots, evolutionary computation, genetic algorithms, cooperative behaviors, Markov chain modeling.

I. Introduction

The motivation behind our present work on cooperating autonomous robot groups comes from the fact that the distributed multi-robot approach has a number of advantages over the traditional single complex robotic systems approach, in that the robot groups can readily exhibit the characteristics of structural flexibility, reliability through redundancy, simple hardware, adaptability, reconfigurability, and maintainability.

In the research on cooperating autonomous robot groups, one of the primary concerns is how to enable individual robots to automatically program their task-handling behaviors adaptive to the dynamic changes in their task environments. Several researchers have started to address the issue of multiple autonomous robot cooperation. Mataric [4], [5] developed a group behavioral learning method in which heterogeneous reward function based reinforcement learning was applied to associate the foraging subset (i.e., summation and/or switching) of six basic behaviors with triggering conditions. Fukuda and Iritani proposed a mechanism for emerging group cooperative behaviors among decentralized autonomous robotic systems, called CEBOT (i.e., Cellular Robots) [1]. Their work simulated the emergence of group behavior based on a globally stable attractor and further the generation of new group behavior based on bifurcation-generated attractors.

In this paper, we describe an approach to evolving robot group behaviors in performing a cooperative pushing task. The proposed approach begins with the modeling of local interactions between the robots and an object in the environment, and then apply a genetic algorithm as a global optimization method for selecting the local motion strategies of the individual robots with an attempt to maximize the overall effectiveness of moving the object toward a goal location.

II. Problem Statement

The problem addressed in this work may be stated as follows: Given a group of three autonomous robots capable of sensing and changing their relative positions and orientations with respect to an object in their workspace, develop an evolutionary computation mechanism that will enable the individual robots to synthesize and execute their local motion strategies for moving toward new positions and orientations. It is required that as a result of their local motions, the object will be collectively pushed by the three robots toward a goal location.

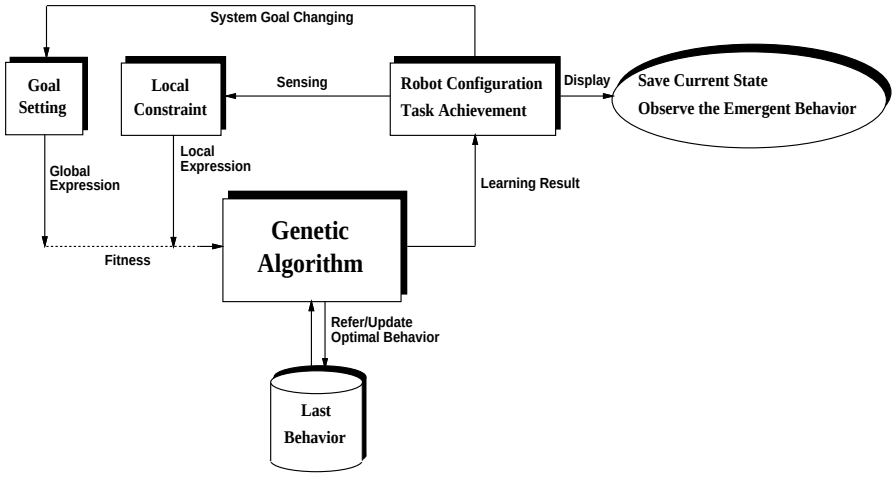

Fig. 1. The schematic diagram of a proposed group behavioral evolution mechanism.

Here, it should be pointed out that with respect to the cooperative object-pushing task, we do not consider the problem of finding the optimal number of autonomous robots for cooperation. Also, in our approach, we do not explicitly model the precise dynamics of the robots and the object interaction. We shall demonstrate our approach with a series of simulations in which we neglect the modeling and control uncertainties in the task environment; this will allow us to rapidly define, validate, and modify our algorithms and to repeatedly observe, analyze, and predict their use and effectiveness without immediately involving expensive hardware. The conclusions and experiences gained from such simulation-based empirical studies will provide us with useful insights into the group behavioral evolution in physical robotic systems.

III. The Proposed Approach

This section presents an evolutionary computation approach to the problem of object-pushing by a group of distributed autonomous robots.

A. An Overview

Figure 1 presents a schematic diagram of the proposed group behavioral evolution mechanism. As shown in the figure, the fitness function of our evolutionary strategies involves two terms: one corresponds to the local interactions, i.e., the relative spatial configurations of individual robots relative an object, whereas the other corresponds to the global feedback information, i.e., the displacement changes of the object relative to a goal location of the object as a result of those robot spatial configurations. Hence, based on a current spatial configuration of the robots, the genetic algorithm will, first of all, generate a series of new spatial configurations, evaluate these configurations using the fitness function (as denoted by a dashed line), and then select the high-fitness configurations for the next step of evolution. That is also to say, the local motion strategies for the individual robots get continuously evolved, as represented in terms of the positions and orientations of the robots in the current and next spatial configurations. In our present work, we apply a fittest-preserved genetic algorithm to implement the evolutionary strategies.

B. A Description of Autonomous Robots

In our work, the autonomous robot used for simulation-based experimentation is equipped with a rotating platform on which an ultrasonic sensor is mounted to sense an object of a certain height in its workspace environment. Whenever the robot detects the presence of an object, it immediately estimates its position and orientation relative to the current and desirable locations of that object. This is illustrated in Figure 2 as distance d_i and angle β_i $(i = 1, 2, 3)$.

It should be pointed out that during object-pushing, the information of a global goal location for the object will be broadcasted to all individual robots via radio links. This will allow the robots to estimate their relative

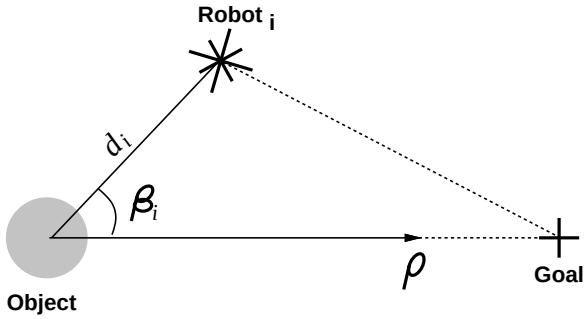

Fig. 2. The autonomous robot is capable of measuring its relative polar coordinates, defined by the current and the desirable locations of a detected object.

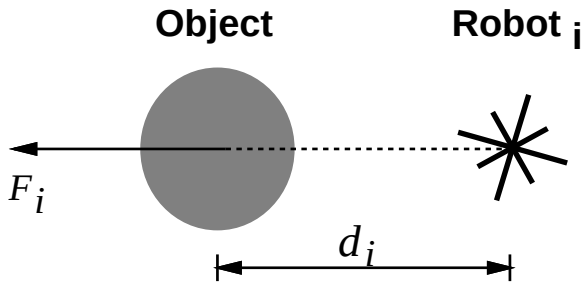

Fig. 3. The pushing force on an object, created by a robot within a distance of impact, l_0, from the object.

positions and orientations, as mentioned above. Apart from the desirable goal location, another important message will also be continuously communicated, that is, the local motion strategies for the robots in the next step, selected from our proposed evolutionary computation mechanism as outlined in Figure 1.

C. Local Interactions between Autonomous Robots and an Object

C.1 The Object and its Motion

Robots can affect the motion of an object if and only if their distances, also called the distances of impact, are less or equal to a predefined constant, l_0. That is, if a robot is situated within an l_0-radius region around the object, it will immediately create a pushing force $\vec{F}_i$ on the object. More specifically, we illustrate the direction of force $\vec{F}_i$ in Figure 3, and assume that its scale, F_i, is proportional to the distance of impact as in the following expression ($i = 1, 2, 3$):

$$F_i = \begin{cases} \eta \times d_i & d_i \leq l_0\,, \\ 0 & otherwise\,. \end{cases} \tag{1}$$

where η is a coefficient, and d_i is the distance between the robot and the object at the time of exerting a pushing impact.

C.2 Pushing Force and its Resulting Motion Created by a Group of Robots

In our present work, we are interested in the cooperation among three autonomous robots in pushing an object toward a desirable goal location. In this case, the pushing force acting on the object should take into account the effects of all three robots. Here, we shall apply the rules of vector composition to combine the pushing forces created by the robots.

More specifically, we consider the motion caused by the collective pushing of the robots on the object, $\sum \vec{F}_i$, as follows: The direction of the motion is the same as the direction of the joint force, and the magnitude of displacement is proportional to that of the joint force, calculated by the following expression:

$$\mathcal{F} = \alpha \times \| \sum_i \vec{F}_i \| \tag{2}$$

where α is a coefficient and $\| \bullet \|$ denotes the modulus.

D. Criteria of Object-Pushing

In the preceding paragraphs, we have mentioned the nature of a cooperative object-pushing task by three robots. In order to further give a more specific high-level requirement for the task, we shall define a set of criteria for performing object-pushing; these criteria are stated as follows:

1. The robots should maximize their joint pushing force on the object.
2. The distance between the object and a desirable goal location, d_{og}, should be minimized; when the object reachs the goal, it will remain at that location. That is, $d_{og} \rightarrow 0$.
3. Whenever a new goal location is defined, the robots should change their motion strategies accordingly.
4. All three robots should keep as close as possible to the object, that is also to say, $\sum d_i \ \searrow$.
5. If possible, the object should be pushed toward the desirable goal location as smoothly as possible, implying that the driving force should not change drastically, that is, $\Delta\theta \ \searrow$ where θ is the direction of the joint force created by the three robots.

It should be noted that the above-mentioned criteria for the task have to be met by all the robots based on their close cooperation. Such cooperative task handling is readily manifested during the process of selecting local motion strategies by the individual robots – the relative spatial configuration of one robot will immediately affect the motion strategies of the rest, and vice versa.

E. Evolving Cooperative Object-Pushing Behaviors

As can be noted from the preceding discussions, each of the individual robots can only sense and control its own spatial configurations even through the task involves three of them. In other words, here we do not explicitly embed in each robot a sophisticated motion planner that will take into consideration the current spatial configurations of others. The "skills" and "intelligence" that each robot has are indeed very primitive that are concerned only with its own basic sensing and motor control.

In order to emerge the high-level intelligence of cooperation among the robots to accomplish the object-pushing task, we utilize an evolutionary computation mechanism that takes as input the current spatial configurations of the individual robots, evaluates their possible local motion strategies, and then broadcasts as output to the robots the highest fitness-of-success ones according to the criteria of task performance. Based on such a mechanism, although the robots remain to behave locally, they can readily produce a certain amount of high-level intelligence, for instance, creating a greater force in order to push the object at a higher speed, and adjusting their joint force in order to create a smoother trajectory for the object.

E.1 The Genetic Algorithm for Behavioral Evolution

The evolutionary computation mechanism for selecting local robot motion strategies is based on a fittest-preserved genetic algorithm (GA). This algorithm is essentially a stochastic search algorithm that draws on a model of fitness-based natural evolution to perform large-search-space optimization. The feature of the algorithm consists in that we emphasize on the highest-fitness individual by recording it directly to the next generation and applying mutation to this individual without any crossover operation. The complete GA as used in our group behavioral evolution is presented in Figure 5. The explicit semantics of the terms and variables as involved in the algorithm will be given in the following subsections.

E.2 Definition of a Chromosome

In the above-mentioned genetic algorithm, each of the chromosomes, i.e., the members as in a population, is represented using a Gray code scheme. Specifically, we assign an 8-bit sub-string to encode the relative spatial configuration of one of the three robots, as shown in Figure 4(a). Furthermore, we subdivide the 8-bit sub-string into two parts, as in Figure 4(b), that correspond to the angle and distance of the robot relative to the object, respectively.

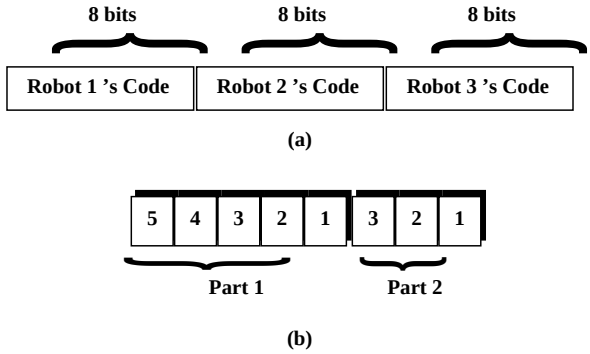

Fig. 4. (a) The definition of a chromosome. (b) The bit assignments for each robot in the chromosome: part 1 encodes orientation angle β_i, whereas part 2 encodes distance d_i.

As already illustrated in Figure 2, the relative spatial configuration of a robot can be specified using orientation angle β_i and distance d_i. For each 8-bit encoding of a robot in the chromosome, the most significant 5 bits are assigned to encode β_i. That is to say, the entire interval of 2π is divided into 32 *units*. Therefore, the highest angular resolution that we assume the robot can have will be $\frac{\pi}{16}$. The rest 3 bits are assigned to encode the distance between the robot and the object. In such a case, the impact distance of the robot is divided into 8 *units*. Therefore, the highest linear distance resolution that we assume the robot can have is $\frac{l_0}{8}$. These assumptions on robot modeling resolutions are reasonable if we consider the measurement and modeling uncertainties inherent in an actual physical robot.

Also, it may be noted from Figure 2 that β_i and d_i will together form a polar coordinate for the robot. In this polar coordinate system, the polar axis is originated at the object, as denoted by '•', pointing toward the goal, as denoted by '+'. Thus, if a robot, i, as denoted by '*' holds a distance of d_i from the object and an orientation angle of β_i in this polar coordinate system, then the relative spatial configuration of the robot, i, can be expressed as follows:

$$\vec{\mathcal{P}}_i = d_i \times e^{j\beta_i}. \tag{3}$$

Accordingly, the direction of a pushing force produced by the robot, $\vec{F}_i$, will become $e^{-j\beta_i}$. The magnitude of this force will satisfy Eq. 1. Thus, we can have:

$$\vec{F}_i = F_i \times e^{-j\beta_i}. \tag{4}$$

E.3 Definition of a Fitness Function

In the genetic algorithm as used for evolving the cooperative motion strategies of robots, a function is used to continuously evaluate the fitness for each member of a population. This function represents the high-level criteria, as we have stated earlier, for the task of cooperative object-pushing. Based on the predefined task-performing criteria, our fitness function to be maximized will be composed of three terms, that is,

$$S_f = s_1 \times s_2 \times s_3. \tag{5}$$

In Eq. 5, s_1 specifies the joint pushing force on the object created by the robots, which can be mathematically expressed as:

$$s_1 = \alpha \times \| \sum_i \vec{F}_i \|. \tag{6}$$

In addition, s_2 measures how closely the object is surrounded by the robots during pushing, which is explicitly defined as follows:

$$s_2 = 3 \times [\sum_i d_i]^{-1}. \tag{7}$$

begin
define a fitness function, S_f,
define the maximum number of generations to be evolved, $G = 100$,
define a population size for selection, $P = 42$,
define the probability for the crossover operation, $p_c = 1$,
define the probability for the mutation operation, $p_m = 1/24$,
create an initial population of P members.
for $generation : 1 \longrightarrow G$ **do**
evaluate fitness-of-success S_f in the current $generation$:
for $population : 1 \longrightarrow P$ **do**
propose the new spatial configurations for the robots,
execute the corresponding local motions of the robots,
evaluate the fitness-of-success for the executed motions.
endfor
record the highest-fitness individuals to the next $generation$,
select other $P - 2$ individuals based on their fitness values,
apply a two-point crossover operation to the $P - 2$ individuals
and get $P - 2$ new members,
mutate the $P - 2$ new members with probability p_m
to create the next $generation$.
endfor
end

Fig. 5. The genetic algorithm as used for evolving cooperative group behaviors in performing an object-pushing task.

Finally, s_3 indicates whether or not and how much the motion of the object is directed toward the desirable goal location, or more specifically:

$$s_3 = \begin{cases} cos\theta & \theta \in [-\frac{\pi}{2}, \frac{\pi}{2}], \\ 0 & otherwise. \end{cases} \tag{8}$$

where θ is the direction of the joint force, created by the three robots.

We can note that S_f has the maximal value when all the robots satisfy $d_i = l_0$ and $\beta_i = \pi$.

IV. A Markov Chain Model of the Fittest-Reserved Genetic Evolution

In what follows, we shall provide several analytical results concerning the convergence of the above-mentioned fittest-preserved genetic algorithm. In the discussions, we shall regard the genetic algorithms proposed in [2] as standard genetic algorithms.

A. Analysis of Standard Genetic Algorithms

A.1 The Number of Possible Populations

Suppose that there are P individuals in a population. For each individual, there exists a fitness function value. Each individual is an L-bit binary string. Any binary string k (also called individual k) can be mapped uniquely to an integer given by $\sum_i^L k(i)2^{i-1}$, where $k(i)$ is the value of the ith bit of k. The integer representing the binary string k will also be called individual k. Here, we shall not consider the order of different individuals in a population. The number of possible populations can be calculated as follow: Beginning from the left, for each individual 0, we write a 1, and then we write a 0. For each individual 1, we write a 1, and then we write a 0, and so on. After we write 1 for individual $2^L - 1$, we need not to write a 0, since there is no 1 afterwards. Now, we can get a string which has P 1's and $2^L - 1$ 0's. Thus the number of possible states

is equal to the number of strings with length $2^L + P - 1$, where P 1's and $2^L - 1$ 0's occur. The number of possible states is given by

$$N = C^P_{2^L+P-1} \tag{9}$$

We give each population a label from 1 to N.

A.2 The Transition Matrix of a Markov Chain

We call the set of possible populations the state space of a Markov chain. The next step is to calculate the transition matrix of the Markov chain. Let $c(0, v)$ denote the occurrences of individual 0 in the population labeled as v, $c(1, v)$ denote the occurrences of the individual 1 in population v, and so on. We write the transition matrix as $Q = (Q_{v,w})$, where $Q_{v,w}$ is the probability that population v generates population w. Then $Q_{v,w}$ is given by ([6]):

$$Q_{v,w} = P! \prod_{j=0}^{2^L-1} \frac{b(j, v)^{c(j,w)}}{c(j, w)!} \tag{10}$$

where $b(j, v)$ is the probability of producing individual j from population v.

B. Analysis of Fittest-Reserved Genetic Algorithms

B.1 The Number of Possible Populations

The number of possible populations under the fittest-preserved genetic algorithms is the same as that under the standard genetic algorithms.

B.2 The Transition Matrix of a Markov Chain

We can calculate the transition matrix analogous to the method used in the standard genetic algorithms. But before we show the expression of transition matrix, we give a new labeling rule for the individuals and the populations.

1. As mentioned above, we use $\sum_i^L k(i)2^{i-1}$ to label each individual k, now the new rule to label each individual k becomes: Assign labels $j = 0, 1, \cdots, 2^L - 1$ to individuals, according to the descending order of their fitness function values ($S_f(k)$). That is, we label individual with the highest fitness function value 0, label individual with the second highest fitness function value 1, and so on. When we name individual k it means that we refer to the individual with $(k + 1)$th highest fitness function value. When two individuals have the same fitness function value, we can define certain criteria to let them have different labels.

2. The rule for labeling different populations is given as follows: Let $k^*(h)$ denote the individual with the highest fitness function value in population h. Assign labels $j = 1, 2, \cdots, N$ to populations according to the ascending order of $k^*(h)$. When two populations h_1 and h_2 have the same $k^*(h_1) = k^*(h_2)$, we can also define certain criteria to let them have different labels.

After giving the labeling rule for the individuals and the populations, we can now give the transition matrix of a Markov chain under the fittest-preserved genetic algorithms:

$$Q_{v,w} = (P - 1)! \prod_{j=0}^{2^L-1} \frac{b(j, v)^{z(j,w)}}{z(j, w)!} \tag{11}$$

for $k^*(v) \geq k^*(w)$, and 0 for $k^*(v) < k^*(w)$, where

$$z(j, w) = \begin{cases} c(j, w) - 1 & j = k^*(w) \\ c(j, w) & j \neq k^*(w) \end{cases} \tag{12}$$

$Q = (Q_{v,w})$ can be divided into 2^L sub-matrices along its diagonal. Each sub-matrix has size $N(i) \times N(i)$, $j = 0, 1, \cdots, 2^L - 1$, where $N(i)$ is the number of populations in which i is the individual with the highest fitness function value, e.g., $k^*(v) = i$. $N(i)$ is obtained by a similar calculation as the one given in Eq. 9:

$$N(i) = C^{2^L-i}_{P+2^L-i-1} \tag{13}$$

Therefore, the transition matrix under the fittest-preserved genetic evolution, $Q = (Q_{v,w})$, can be written as follows:

$$\begin{bmatrix} \boxed{\text{Q(0)}} & & & & \\ & \boxed{\text{Q(1)}} & & \mathbf{0} & \\ & & \ddots & & \\ & & \overbrace{\boxed{\text{Q(i)}}}^{N(i)} \} N(i) & & \\ & \mathbf{nonzero} & & \ddots & \\ & & & & \boxed{Q(2^L-1)} \end{bmatrix} \tag{14}$$

This Markov chain will have absorbing states. Any initial state will eventually get into the absorbing states. The absorbing states will consist of only populations with individual 0 in it.

V. Characterizing the Transition Matrix Using Eigenvalues

Let $\lambda_1, \lambda_2, \cdots, \lambda_N$ be the N eigenvalues of the transition matrix $Q = (Q_{v,w})$. From matrix theory, we know that there exists a matrix T such that:

$$Q = T^{-1} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} T = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N} \\ b_{21} & b_{22} & \cdots & b_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ b_{N1} & b_{N2} & \cdots & b_{NN} \end{bmatrix} \tag{15}$$

where T^{-1} is the inverse matrix of T, that is,

$$\sum_{l=1}^{N} a_{vl} b_{lw} = \begin{cases} 1 & v = w \\ 0 & v \neq w \end{cases} \quad (v, w = 1, 2, \cdots, N) \tag{16}$$

We can obtain

$$Q_{v,w} = \sum_{l=1}^{N} \lambda_l a_{vl} b_{lw}$$

$$Q^n = (Q_{v,w}^{(n)}) = T^{-1} \begin{bmatrix} \lambda_1^n & & & \\ & \lambda_2^n & & \\ & & \ddots & \\ & & & \lambda_N^n \end{bmatrix} T$$

and

$$Q_{v,w}^{(n)} = \sum_{l=1}^{N} \lambda_l^n a_{vl} b_{lw}$$

where $Q_{v,w}^{(n)}$ is the probability that population w can be generated from population v after n generations. In our fittest-preserved genetic algorithm, $Q_{v,w}^{(n)}$ can be written in an equivalent form:

$$Q_{v,w}^{(n)} = \sum_{i}^{2^L-1} \sum_{j=1}^{N(i)} \lambda_{i,j}^n a_{v,i,j} b_{i,j,w}$$

Let Ω_0 denote the set of populations which include individual 0. Then for $v \in \Omega_0$ and $w \notin \Omega_0$, $\Omega_{v,w}^{(n)} = 0$. More generally, $\Omega_{v,w}^{(n)} = 0$ when $k^*(v) < k^*(w)$.

If the initial state is $q^{(0)} = (q_1^{(0)}, q_2^{(0)}, \cdots, q_N^{(0)})$, then the state in the next generation will become: $q^{(1)} = (q_1^{(1)}, q_2^{(1)}, \cdots, q_N^{(1)}) = q^{(0)}Q$ and the state after n generations will be $q^{(n)} = (q_1^{(n)}, q_2^{(n)}, \cdots, q_N^{(n)}) = q^{(0)}Q^n$. $\sum_{w\in\Omega_0} q_w^{(n)} = \sum_{w=1}^{N(0)} q^{(n)}$ represents the probability that a population including individual 0 will be generated after n generations, while $\sum_{w\notin\Omega_0} q_w^{(n)} = 1 - \sum_{w\in\Omega_0} q_w^{(n)}$ represents the probability that no population including individual 0 will be generated. We can show $\sum_{w\notin\Omega_0} q_w^{(n)}$ is upper-bounded. (derived from (1))

$$\begin{aligned}
\sum_{w\notin\Omega_0} q_w^{(n)} &= \sum_{w\notin\Omega_0}\sum_{v=1}^{N} q_v^{(0)} Q_{v,w}^{(n)} && (17)\\
&= \sum_{w\notin\Omega_0}\sum_{v=1}^{N}\left(q_v^{(0)}\sum_{i}^{2^L-1}\sum_{j=1}^{N(i)}\lambda_{i,j}^n a_{v,i,j} b_{i,j,w}\right) && (18)\\
&\le \left(\sum_{w\notin\Omega_0}\sum_{v=1}^{N}\left(q_v^{(0)}\sum_{i}^{2^L-1}\sum_{j=1}^{N(i)} a_{v,i,j} b_{i,j,w}\right)\right)|\lambda_{\max}|^n && (19)\\
&= \left(\sum_{w\notin\Omega_0} q_v^{(0)}\right)|\lambda_{\max}|^n \quad (derived\ from\ 16) && (20)\\
&= Const\ |\lambda_{\max}|^n && (21)
\end{aligned}$$

where $Const$ is a positive constant less than 1. Then, we can get:

$$\sum_{w\in\Omega_0} q_w^{(n)} \ge 1 - \ Const\ |\lambda_{\max}|^n \ge 1 - |\lambda_{\max}|^n \qquad (22)$$

The next step is to evaluate the absolute value of $\lambda_{\max}$. It is obvious that λ is the eigenvalue of Q if and only if λ is the eigenvalue of one of the submatrices $Q(i)$, as the determinant of Q is equal to the multiplication of the determinants of $Q(i)$. Then from Markov chain theory and matrix theory [3], we know that

$$|\lambda_{\max}| \le \max_{0\le i<2^L-1}\ \max_{1\le j\le N(i)} \sum_{v=1}^{N(i)} Q(i)_{j,v} \qquad (23)$$

where $\sum_{v=1}^{N(i)} Q(i)_{j,v}$ represents the probability that the highest fitness function value of population $\sum_{k=0}^{i-1} N(k)+j$ will not get better or worse (of cause will not get worse because we preserve the fittest individual) in the next generation, while $1 - \sum_{v=1}^{N(i)} Q(i)_{j,v}$ represents the probability that the highest fitness function value of population $\sum_{k=0}^{i-1} N(k) + j$ will get better in the next generation.

Now let us calculate the least probability that a population will get better in the next generation in our robot-pushing-object case. From our coding method, we can prove that changing only one bit of the chromosome can make the individual corresponding to the chromosome have a higher fitness function value. The detailed proof will be not be given here, due to the limited space.

In other words, no matter under what circumstances, we need only one bit of the chromosome to be mutated in order to obtain a new individual with a higher fitness function value. If we apply this rule to the individual with the highest fitness function value, then we are able to obtain a new population which will get better. The probability that the mutation happens on the right bit and does not happen on the other bits is the least probability that a new population will get better. In fact, the probability is much higher because crossover and mutation on the other individuals can also generate some individuals with higher fitness function value than the original fittest individual. But it is enough for us to get a formula to consider mutation on the individual with the highest fitness function value. Since we let the individual with the highest fitness value in a generation to get into mutation directly, the least probability that a population will get better in the next generation is:

$$(1 - p_m)^{23} p_m \qquad (24)$$

That is:

$$1 - \max_{0 \le i < 2^L - 1} \max_{1 \le j \le N(i)} \sum_{v=1}^{N(i)} Q(i)_{j,v} \ge (1 - p_m)^{23} p_m \tag{25}$$

Thus, we can have:

$$|\lambda_{\max}| \le \max_{0 \le i < 2^L - 1} \max_{1 \le j \le N(i)} \sum_{v=1}^{N(i)} Q(i)_{j,v} \le 1 - (1 - p_m)^{23} p_m \tag{26}$$

Differentiate the right side of the inequality above and let the differentiation be equal to 0, we obtain $(1 - p_m)^{22} (1 - 24 p_m) = 0$. The right side of the inequality will get its minimal value if $p_m = 1/24$. That is the reason why we select $p_m = 1/24$ in the algorithms we presented earlier. Then $|\lambda_{\max}| \le 1 - (1 - 1/24)^{23} \times 1/24 = 0.985$. Now let us revisit Eq. 22 and rewrite it as follows:

$$\sum_{w \in \Omega_0} q_w^{(n)} \ge 1 - |\lambda_{\max}|^n \ge 1 - 0.985^n \tag{27}$$

Therefore, we can have:

$$\begin{aligned} n = 50 \qquad & \sum_{w \in \Omega_0} q_w^{(n)} \ge 1 - 0.985^5 0 \approx 0.53 \\ n = 100 \qquad & \sum_{w \in \Omega_0} q_w^{(n)} \ge 1 - 0.985^1 00 \approx 0.78 \\ n = 150 \qquad & \sum_{w \in \Omega_0} q_w^{(n)} \ge 1 - 0.985^1 50 \approx 0.9 \end{aligned}$$

We know from the above list that no matter what initial population will be, we can have a probability greater than 50 percent to get a population with individual 0 in it after 50 generations, a probability greater than 78 percent after 100 generations, a probability greater than 90 percent after 150 generations. When we have a population with individual 0, the direction of the joint force by the three robots is exactly towards the goal. The object can be pushed towards the goal smoothly.

VI. EXPERIMENTS

In this section, we shall describe our experiments in which the above-mentioned GA-based group behavioral evolution mechanism is implemented and validated. In order to have a closer examination into the process of cooperative behavioral emergence, we shall present and trace several snapshots of the behavioral selection and motion execution in a typical case study.

A. Robot Workspace

In our experiments, we have defined the workspace for the three robots as a square arena, $\mathcal{R}$, of size $10\ units \times 10\ units$. Whenever the robots or the object hit the boundary of the workspace, the task of object-pushing will be aborted. At the beginning, the three autonomous robots and the object will be randomly placed inside square $\mathcal{R}$. A desirable goal location will be arbitrarily defined within the workspace.

B. A Case Study

Now, let us take a look at a typical case of object-pushing by a group of three autonomous robots. Figure 6 provides a series of snapshots showing the process of group behavioral emergence based on the GA computation, from generation 4 till generation 72 (taken every 4 generations). We can note that three robots continuously modify their spatial configuration in order to satisfy the high-level fitness-of-success requirement. As a result, the object, as denoted by '•', is gradually brought toward a desirable goal location, as denoted by '+'. The object is pushed to reach the goal at generation 64.

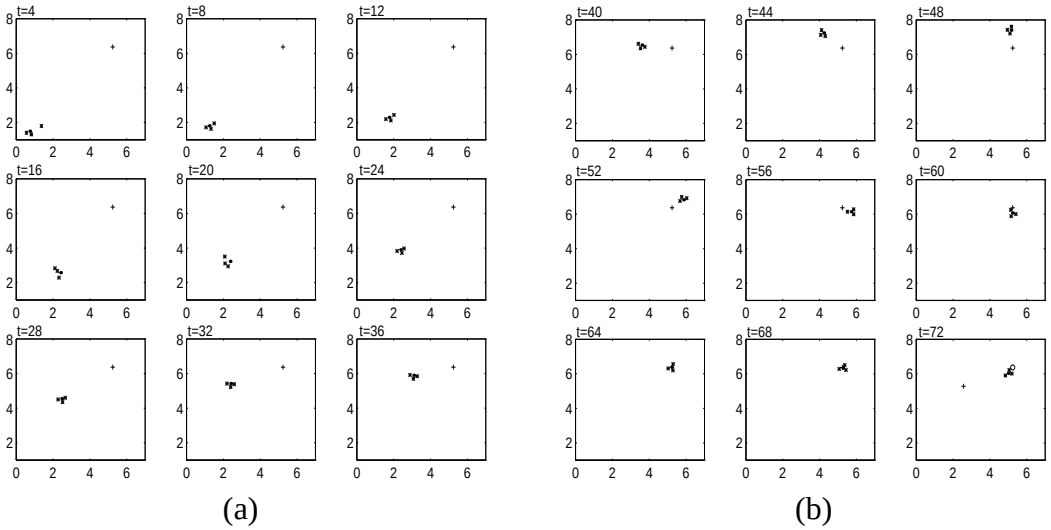

Fig. 6. (a) The snapshots of object-pushing by three robots taken every 4 generations from generations 4 to 36. (b) The snapshots of object-pushing by three robots taken every 4 generations from generations 40 to 72.

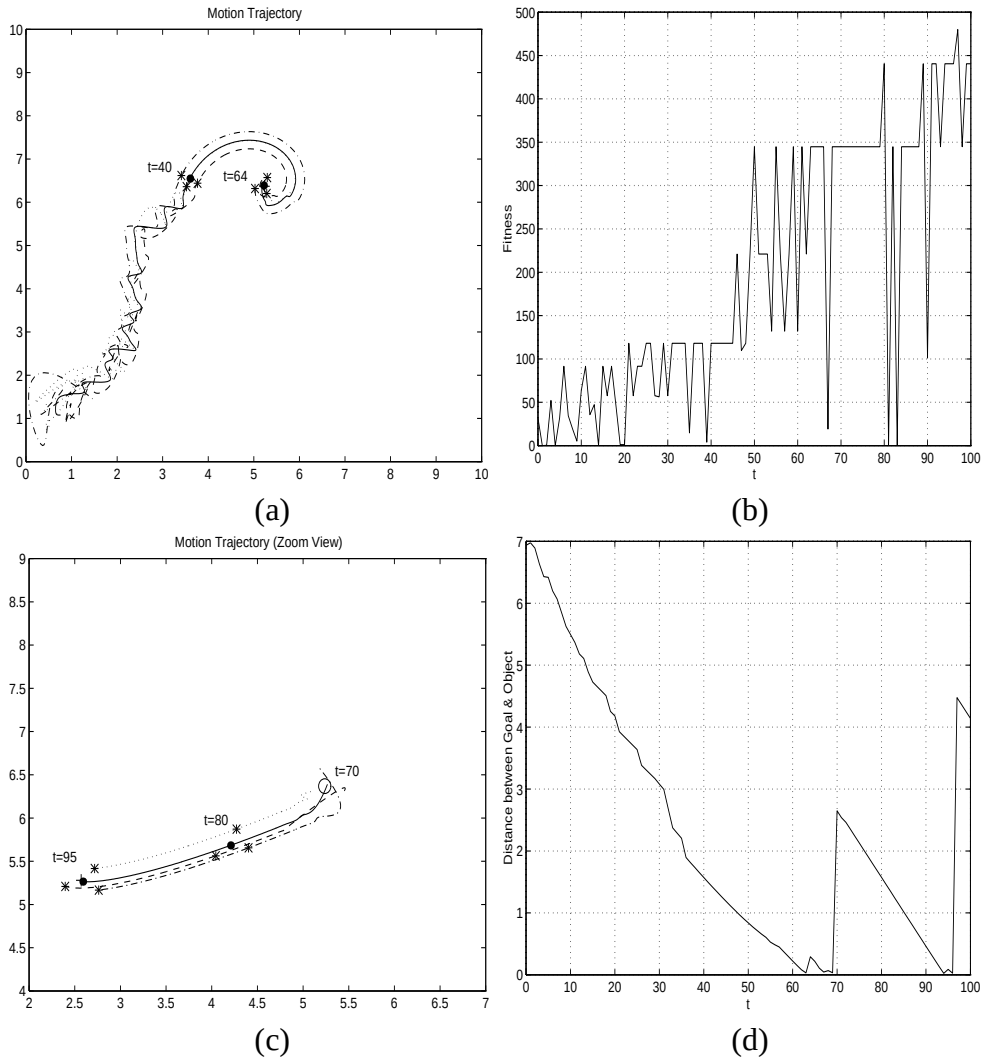

Fig. 7. (a) The object-pushing trajectories created the robots from the beginning of the task until the object reaches a desirable goal location. (b) A plot of fitness function over the entire evolution process of cooperative object-pushing. (c) The object-pushing trajectories, created by the robots, after the goal is changed to a new location at generation 70. (d) The distance between the object and the goal recorded in our experimental case study.

C. Emergence of Cooperative Pushing Behaviors

Figure 7(a) presents the trajectories for all three robots and the object being pushed. In the figure, the solid line denotes the trajectory of the object, and the dashed line, dotted line and dash-dot line correspond to those of the three robots. Here, we may note that at the beginning of object-pushing, the robots moved in a rather chaotic manner. It takes more than 20 generations for the group of distributed robots to emerge effective group behaviors satisfying the fitness function given in Eq. 5. The motion trajectories are getting increasingly smoother, as the evolution-embedded object-pushing continues. After about 40 generations, as marked in the figure, the cooperation among the robots becomes most apparent.

This observation can also be validated from the fitness function plot over the entire process of behavioral evolution, as given in Figure 7(b). Here, we may identify four different stages in the evolution. The first stage occurs at the early 20 generations, where the robots execute their motion strategies quite chaotically. The reason is that the chromosome has just been randomly selected, and no high-fitness configuration is found with the genetic operations. The next stage happens in the following 19 generations, i.e., from generations 21 to 39. During this period, the robots find several group behaviors, but with some low-fitness ones. Then, in a later task performance, from generations 40 to 64, their cooperative behaviors are mostly emerged, even though there are few slight oscillations. Finally, after generation 64, the fitness values reach a new level, even when the goal location is changed twice: one at generation 70 and another at generation 97.

D. Adapting to Dynamically Changing Goals

In our present case study, we are also interested to see whether the robots can adapt to the change of the goal location. Figure 7(c) presents the traced motion trajectories of the robots after the goal is changed to a new location at generation 70. In such a case, the robots can immediately bring the object to the new goal location, implying that the process of GA-based evolutionary behavioral emergence can readily adapt to the changes in the task environments. In addition, such adaptation is also robust, as reflected in the fitness function plot of Figure 7(b). In Figure 7(c), symbol 'o' at t=70 denotes the previous goal location.

The distance between the object and the goal during the entire 100 generations of evolutionary cooperation is given in Figure 7(d).

VII. Conclusion

In this paper, we have described a GA-based evolutionary computation approach to the emergence of cooperative group behaviors in distributed autonomous robots, and given both analytical and experimental results concerning cooperative behavior evolution. We have shown that the object-pushing direction can have a high probability to be optimal after several generations.

The problem of object-pushing is a variant of many representative object manipulation problems, the solutions of which can readily be scaled up to other more complex tasks, such as material handling and mechanical assembly, that require motion planning and control for coordination by cooperating robots. On a broader perspective, the methodology developed in this work will have significant bearings on how to creating goal-directed, task-driven behaviors for a group of distributed autonomous agents.

References

[1] Toshio Fukuda and Go Iritani. Construction mechanism of group behavior with cooperation. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '95)*, Pittsburgh, PA, 1995.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley publishing company, Reading, MA, 1989.

[3] John G. Kemeny and J. Laurie Snell. *Finite Markov Chains*. 1960.

[4] Maja Mataric. *Interaction and Intelligent Behavior*. Ph.D. dissertation, Department of Electrical Engineering and Computer Science, MIT, USA, 1994.

[5] Maja Mataric. Reward functions for accelerated learning. In William W. Cohen and Haym Hirsh, editors, *Proceedings of the Eleventh International Conference on Machine Learning*, San Francisco, CA, 1994. Morgan Kaufmann Publishers.

[6] A.E. Nix and M.D. Vose. Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5.