20

Multi-Robot Task Allocation Based on Swarm Intelligence

Shuhua Liu[1], Tieli Sun[1] and Chih-Cheng Hung[2]
[1]*Northeast Normal University*
[2]*Southern Polytechnic State University*
[1]*China*
[2]*USA*

1. Introduction

In the field of cooperative robotics, task allocation is an issue receiving much attention. When researchers design, build, and use cooperative multi-robot system, they invariably try to answer the question of which robot should execute which task. This is in fact a multi-robot task allocation problem (MRTA). The task allocation problem addresses the question of finding the task-to-robot assignments that optimize global cost or utility objectives. Finding an optimal task allocation, even in a relatively simplified case, is an NP-hard problem. Therefore, the majority of common approaches are approximate or heuristic in nature. Those approaches usually give suboptimal solutions. MRTA is a fundamental issue of the multi-robot systems, which embodies the high-level system organization and operation mechanism. The quality of task allocation algorithm directly affects the performance of multi-robot system. With an increase in the number of robots and difficulty of tasks within a system, the issue of task allocation has risen to prominence and become a key research topic in the multi-robot domain. In 2005, the International Conference on Robotics and Automation (ICRA 2005) set special panels on multi-robot task allocation, in which the latest research and the progress are discussed.

Gerkey and Mataric (2004) presented a particular taxonomy for the task allocation problem. It is described as follows:

- Single-task robots (ST) vs. multi-task robots (MT): ST means that each robot is capable of executing at most one task at a time, while MT means that some robots can execute multiple tasks simultaneously.
- Single-robot tasks (SR) and multi-robot tasks (MR): SR means that each task requires exactly one robot to achieve it, while MR means that some tasks can require multiple robots.
- Instantaneous (IA) and time-extended (TA) assignment: In the instantaneous assignment, robots do not plan for future allocations and are only concerned with the one task they are carrying out at the moment (or for which they are considering executing). In the time-extended assignment, robots have more information and can come up with longer-term plans involving task sequences or schedules.

Based on above categorization, there are eight types of task allocation combination. ST-SR-IA is the simplest, as it is actually a trivial instance of the Optimal Assignment Problem

(OAP). ST-MR-IA often appears in real world applications; that is, some tasks require the combined effort of multiple robots. These two types of tasks are also called loosely-coupled tasks and tightly-coupled tasks, respectively. Although some approaches for solving either loosely-coupled task or tightly-coupled task allocation have been proposed, few approaches for solving both loosely-coupled and tightly-coupled task allocation have been developed.
In this chapter, we present a task allocation mechanism based on swarm intelligence for the large-scale multi-robot system, with both loosely-coupled and tightly-coupled task allocation. The mechanism adopts a hierarchical architecture. At the high level, we employ an Ant Colony Algorithm to find optimal allocations. Namely, each ant performs a task allocation so as to choose an undertaker for every task. At the low level, each ant forms a task-oriented robot coalition to perform a tightly-coupled task. Ant colony optimization (ACO), the particle swarm and ant colony optimization (PSACO) and the quantum-inspired ant colony optimization (QACO) are adopted to form the coalition. Finally, the algorithm is implemented in the TeamBots simulation platform. Simulation results show that the proposed mechanism can effectively solve loosely-coupled and tightly-coupled task allocation in the large-scale multi-robot system.

2. Related work

Recently a number of solutions have been proposed in the literature to MRTA problems (Zhang & Liu, 2008). These include behaviour based approaches such as ALLCANCE (Parker, 1998), BLE (Werger & Mataric, 2000) and ASyMTRe(Tang & Parker, 2005). The advantage of these approaches possesses real-time, fault-tolerance and robustness; the solution, however, can only be locally optimal. The market-based approach is the current mainstream of task allocation methods. The representative method is CNP (Contract Network Protocol) which proposed by Smith (1980). Other typical examples include First-price auctions (Zlot et al, 2002), Dynamic Role Assignment (Chaimowicz et al, 2002), Traderbots (Dias, 2004), M+ (Botelho & Alami, 1999), MURDOCH (Gerkey & Mataric, 2002a) and DEMiR-CF (Sanem & Tucker, 2006). Because of better scalability, this method is particularly well-suited to the distributed robotic domain. Furthermore, it is guaranteed to produce optimal allocations, but robots must cooperate through explicit communication and more resource consumption. Once the communication is interrupted, the performance of this method will degrade significantly (Kalra & Martinoli, 2006). Therefore, it is suitable for small- to medium- scale task allocation problems. Derived from the behaviours of social insects, the swarm intelligence approach is exhibiting several good features such as self-organizing ability in unknown environments, and emergent and adaptive behaviours through simple interaction among individuals. Since cooperative individuals are distributed and there is no central control and global data in the group, the system will be more robust. The failure of one or several individuals will not affect the whole solution. Additionally, individuals cooperate through implicit communications. As the number of the individuals in the system increases, the amount of communication grows quite slowly. Therefore the swarm intelligence approach is the most suitable for distributed multi-robot systems and as such more and more researchers have applied it to the multi-robot task allocation, especially in dynamic environments. Ding et al. (2003) and Yang &Wang (2004) adopted Ant colony algorithm for multi-robot cooperation. Zhang et al. (2007) employed swarm intelligence for adaptive task assignment. Zhang & Liu (2008 b, 2009) and Liu & Zhang (2009, 2010) conducted intensive research on swarm intelligence and applied it to the task allocation of large-scale multi-robot system.

3. Architecture

Ant Colony Algorithm is a new intelligent optimization algorithm and first proposed by Colorni et al. (1992). In ant colony algorithm, each ant searches for solutions independently in the candidate solution space, and lays some pheromone on the found solution. The better the solution, the more pheromone the ant lays. A solution with higher pheromone has a much greater chance of being chosen, and consequently this gives a kind of positive feedback. Through this positive feedback, ants can eventually find the optimal solution. Via this process the algorithm effectively solves combinatorial optimization problems and performs especially well in solving complicated problems (Jiang et al, 2003; Xia et al, 2005).

The paper adopts a hierarchical architecture, as shown in Fig.1. At the high level, we employ the Ant Colony Algorithm to find optimal allocations. Let an ant denote a task; each ant forms its task allocation so as to choose an undertaker for every task. At the low level, each ant forms a task-oriented robot coalition to perform a tightly-coupled task by the ant colony optimization (ACO), the particle swarm and ant colony optimization (PSACO) and the quantum-inspired ant colony optimization (QACO). It is worth mentioning that the proposed mechanism can not only solve loosely-coupled task allocation, but also tightly-coupled task allocation because ants in the high level denote tasks instead of individual robots. Finally, simulation results give a performance comparison, and then conclusions follow.

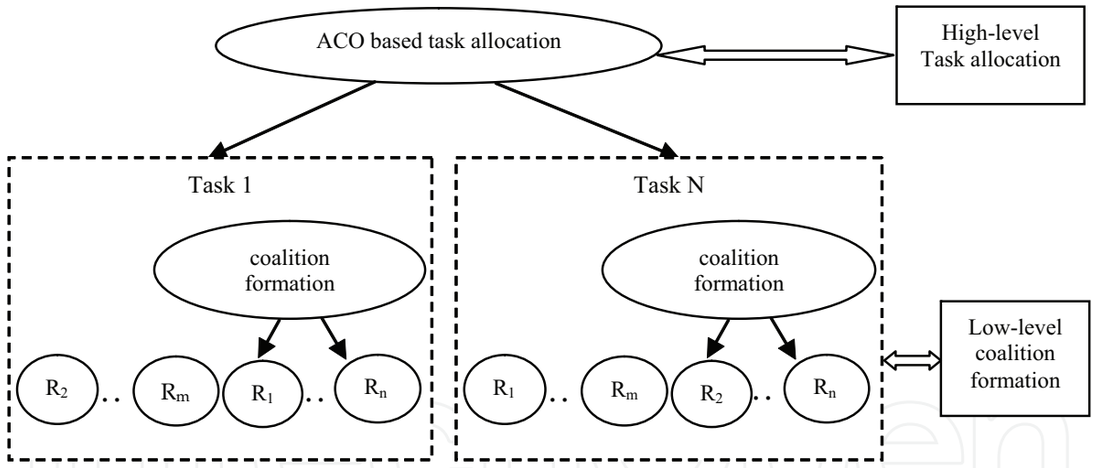


Fig. 1. Hierarchical architecture of the system

4. Key issues of robot coalition formation

4.1 Validity of robot coalition

Similar to agent coalition formation, robot coalition formation also tries to find the robot coalition with the greatest value that can complete a task *t*. A coalition may be formed by several arbitrary robots in the system. However, in order to obtain a satisfactory result, we must consider all or most of the combinations. Therefore it is a complex combinatorial optimization problem. In addition, although there are many similarities between agent coalition and robot coalition, there are also inherent differences which should not be overlooked.

Firstly, software agents are simply code fragments whose capabilities corresponding to software functionality and current data knowledge while robots are tangible entities that occupy physical space and whose capabilities correspond to sensors, actuators, etc. Multi-robot systems must handle real world sensory noise, full or partial robot failures, and communication latency or even loss of communications.

Secondly, agents are allowed to exchange resources, so the formed coalition freely redistributes resources amongst the members. However, this is not possible in a multiple-robot domain. Robot capabilities in handling sensors (camera, laser, sonar, or bumper) and actuators (wheels or gripper) cannot be autonomously exchanged. This implies that a robot coalition that simply possesses the adequate resources is not necessarily up to performing a given task, and other locational constraints have to be represented and met in order for the coalition to succeed.

Finally, correct resource distribution is an important issue in the robot coalition formation. The box-pushing task (Gerkey & Mataric, 2002 b) is used to illustrate this point. Three robots, two pushers (with one bumper and one camera) and one watcher (with one laser range finder and one camera) cooperate to complete the task. The total resource requirements are: two bumpers, three cameras and one laser range finder. However, this information is incomplete, as it does not accurately represent the constraints related to sensor locations. Correct task execution requires that the laser range finder and camera reside on a single robot while the bumper and laser range finder reside on different robots. Therefore each candidate coalition must be verified feasibly.

Checking the feasibility of robot coalition is a Constraint Satisfaction Problem (CSP). It is defined by a set of variables, a set of the domain values for each variable and a set of constraint relationships between variables, which is denoted as (V,D,C). Where V is the set of variables $\{V_1,\ldots,V_n\}$ which are resources and capabilities requirements, in box-pushing task, $V_1,\ldots,V_n$ are the bumper, camera and laser range finder. D is the set of the domain values which is the sum of the available robots possessing the required resources and capabilities, $D=\{D_1,\ldots,D_n\}$, where D_i is the limited domain of V_i's all possible values. C is the set of constraint relationships between variables, $C=\{C_1,\ldots,C_m\}$, each constraint includes a subset of V, that is $\{V_i,\ldots,V_j\}$ and a constraint relationship $R \subseteq D_i\times\ldots\times D_j$. For the box-pushing task, two types of constraints exist, the sensors and actuators must reside either on the same robot or on different robots. As shown in Fig. 2, locational constraints are represented as solid arcs (same robot) and dash arcs (different robot).

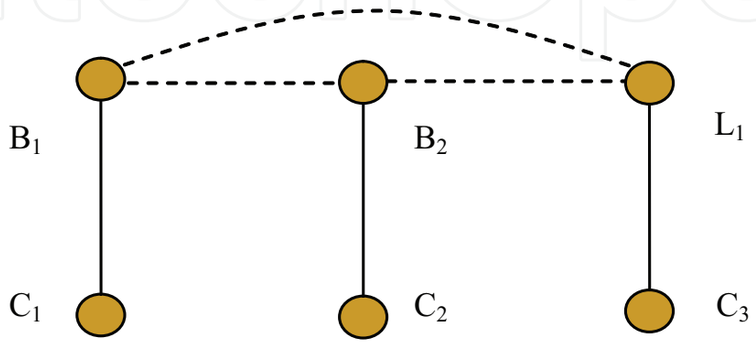


Fig. 2. Box-pushing task constraint graph

4.2 The evaluation criteria of robot coalition formation

Because robots are typically unable to redistribute their resources, it is possible that the coalition will have one or a few robots as main resource providers. This kind of coalition tends to be heavily dependent on these members for task execution that these dominating members become indispensable. Such coalitions should be avoided in order to improve fault tolerance. The coalition imbalance is defined as the degree of unevenness of resource contributions made by individual members to the coalition. The perfectly balanced coalition is where each member contributes equally (taskvalue/n) to the task. The Balance Coefficient (BC) quantifies the coalition imbalance level. The BC can be calculated as follows:

$$\mathrm{BC} = \frac{\gamma_1 \times \gamma_2 \times \cdots \times \gamma_n}{\left[\frac{\mathrm{taskvalue}}{n}\right]^n} \tag{1}$$

where $(\gamma_1,\gamma_2,...,\gamma_n)$ is a resource distribution with a coalition C. For the coalitions of the same size, the higher BC, the more balanced the coalition is.

In general, larger coalitions imply that the average individual contribution and the capability requirements from each member are lower; thus larger coalitions are more balanced. However, larger coalitions have much more costs and therefore it is necessary to consider coalition balance and coalition size simultaneously. The Fault Tolerance Coefficient (FTC) metric can be used to solve this problem and it is defined as follows:

$$\mathrm{FTC} = \delta \mathrm{BC} + \mu f(n) \tag{2}$$

where $\delta+\mu=1$, $f(n)=1-e^{-\lambda n}$ is the function of coalition size. After a particular point, increasing n will not result in a significant increase to the function value. This means that enlarging coalition size does not yield improved performance when the number of robots increases beyond a threshold value. This, as one might imagine, is in accordance with a realistic robot application.

4.3 The description of robot coalition formation problem

1. The Ability Description of Robots

All robots in the system form a robot set $R=\{R_1,R_2,\ldots,R_n\}$. The ability vector of R_i is $B_{Ri}=(b_{i1},b_{i2},...,b_{im})^T$, and the ability cost vector is $cost_{Ri}=(cost_{i1},cost_{i2},...,cost_{im})^T$,where $cost_{ij}$ is the cost of the ability b_{ij}. When $b_{ij}=0$, it denotes R_i without the ability b_{ij}. The cost of R_i is $\sum_{j=1}^{m} \mathrm{cost_{ij}} \mathrm{b_{ij}}$, which has m kinds of abilities.

2. The Ability Description of Robot Coalition

Robot coalition is a set of robots in which robots can cooperate to complete a task. A coalition C is the nonempty subset of R. Based on the different ability attributes of the robots, there are different ability vectors of the coalition. For the additive capacity (such as handling, etc.), the ability of the coalition C is as follows:

$$\mathrm{B_C} = \sum_{\mathrm{R_i} \in C} \mathrm{B_{R_i}} \tag{3}$$

For the merger capacity (such as video distance, etc.), the ability of the coalition *C* is as follows:

$$B_C = \bigcup_{R_i \in C} B_{R_i} \tag{4}$$

The cost of the coalition ability is defined as follows.
The additive capacity:

$$D(C) = \sum_{R_i \in C} \sum_{j=1}^{m} cost_{ij} b_{ij} \tag{5}$$

The merger capacity:

$$D(C) = \bigcup_{R_i \in C} \sum_{j=1}^{m} cost_{ij} b_{ij} \tag{6}$$

3. The Requirement Description of Task Capacity

There are *K* tasks, denoted by $T = \{t_1, t_2, \cdots, t_k\}$. The task *t* has the ability requirement vector: $B_t = (b_1, b_2, \cdots, b_m)^T$.
The essential condition for the coalition *C* to finish the task *t* is as follows：$B_C \geq B_t$.

4. The Definition of Coalition' s Income

We define a *reward* function which is a mapping from the set of tasks to the set of real numbers, denoted by *reward*: $T \rightarrow R^+$. A *cost* function is defined as *cost*: $C \rightarrow R^+$, which is a mapping from the set of coalitions to the set of real numbers. We consider two types of cost:

- A *coalition-inherent* cost measures the inherent cost (e.g., in terms of energy consumption or computational requirements) of using particular capabilities of the coalition. Here the main consideration is the consumption of the robot's ability to accomplish the tasks, including the communication between the robots in the coalition and the cost of the coalition ability. We denote it by *C_cost*.
- A *task-specific* cost measures cost according to task-related metrics, such as time, distance, etc. Here we mainly consider the distance. We denote the cost of the coalition performing the task by *T_cost*.

Thereby, the *cost* function of the coalition *C* performing task *t* is denoted as:

$$Cost(C,t) = \varpi_1 C\_cost + \varpi_2 T\_cost \tag{7}$$

where ϖ_1 and ϖ_2 are weighted coefficient of both the *coalition-inherent* cost and *task-specific* cost, $\varpi_1 > 0$, $\varpi_2 > 0$. According to the differences between agent coalitions and robot coalitions, the income of the robot coalition should be defined as:

$$Inc(C) = FTC \times \left[rew(t) - Cost(C,t) \right] \tag{8}$$

where FTC is the Fault Tolerance Coefficient, rew(t) is the reward after robots accomplish task *t*.

5. Low-level coalition formation

At the low level, we employ the ant colony optimization (ACO), the particle swarm and ant colony optimization (PSACO) and the quantum-inspired ant colony optimization (QACO)

for the coalition formation. Their performance of forming robot coalition for tightly-coupled task is compared by simulation results.

5.1 Forming robot coalition by ant colony algorithm

Put m ants on n robots at random, the probability of ant k located on the Robot i choosing Robot j is defined as follows:

$$p_{ij}^{k} = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[1/d_{ij}\right]^{\beta}}{\sum_{u\in J_k}\left[\tau_{ij}(t)\right]^{\alpha}\left[1/d_{iu}\right]^{\beta}}, j \in J_k \tag{9}$$

where J_k is the robot set that ant k has not chosen; $\tau_{ij}(t)$ is the quantity of pheromone remaining on the line between robot i and robot j; d_{ij} (i,j=1,2,…,n) is the distance between robot i and robot j, called communication cost; a and β control the relative weights of pheromone and communication cost. The ant will stop seeking a route when it arrives at a certain robot and finds that the current robot coalition can accomplish the task. When all ants have formed their task-oriented coalitions, one loop finishes. Then each candidate coalition is checked to verify its feasibility. Update the maximal income and the intensity of pheromone according to the following Equation.

$$\tau_{ij}(t+1) \leftarrow \rho\tau_{ij}(t) + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{10}$$

Here $\Delta\tau_{ij}^{k}$ is the increment of the familiar degree between robot i and robot j given by ant k in this loop and it is defined as:

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Inc(C_k)}{\sum_{k=1}^{m} C_k}, & \textit{if the coalition formed by ant k includes robot i and j} \\ 0, & \textit{others} \end{cases} \tag{11}$$

$Inc(C_k)$ is the income of the coalition formed by ant k. The optimal combination of parameters a, β and ρ in this algorithm can be determined by the experimental method. The program termination may be controlled by a fixed evolving generation or when the evolving trend is inconspicuous. The time complexity degree is $O(NC.m.n^2)$, NC is the number of loops.

5.2 Forming robot coalition by particle swarm and ant colony optimization

Particle Swarm Optimization (PSO) was proposed by Eberhart and Kennedy (1995). Inspired by foraging behaviours of birds, birds are viewed as particles of swarm and their motion is affected by their own velocity, best position of individual and population in the past. As a result, an optimal solution can be obtained in a complex solution space.

The system is initialized with a population of random particles and then the best solution can be found through iterations. In each time step, particles update their velocity and position by the following formula:

$$v_{k+1} = c_0 v_k + c_1\left(pbest_k - x_k\right) + c_2\left(gbest_k - x_k\right) \tag{12}$$

$$x_{k+1} = x_k + v_{k+1} \tag{13}$$

where, *pbest* denotes the optimal position of single particle, *gbest* denotes the optimal position of whole population, v_k is the velocity of the particle, x_k is the current position of the particle, c_0, c_1 and c_2 are weight coefficients.

1. Particle Swarm and Ant Colony Optimization (PSACO)

PSO is suitable for dealing with continuous optimal problems, but for discrete optimal problems it is difficult to express the velocity of a particle. Therefore, inspired by Genetic Algorithms, $c_0 v_k$ is viewed as variation operator, while $c_1(pbest_k - x_k) + c_2(gbest_k - x_k)$ is viewed as the crossover operator of current solution with the individual optimal value and the global optimal value respectively.

The PSACO takes an ant as a particle. Ants choose their cooperative ants based on their own information, *pbest* and *gbest*. Then the current coalition executes crossover operations with individual optimal coalition and global optimal coalition to form new coalition. Finally, the new coalition executes a variation operator.

The adopted crossover strategy is to choose a random position from the second string as a crossover point. In addition, the variation rule is constructed so as to choose a random position, if the variation bit is -1 (the robot is not chosen), its value is set 1 (the robot is chosen), and vice versa.

2. The PSACO Algorithm

The PSACO algorithm is described as follows:

Step 1. Initialization

Set $NC = 0$, $J_k = \{1, 2, \cdots, n\}$. Execute ACO to form *m* initial coalitions and then compute the fitness *Income0* of each coalition according to Eq. (8). Treat current fitness as the individual optimal value *ptbest* and treat current coalition as the individual optimal value coalition *pcbest*. Then, find the global optimal value *gtbest* and global optimal value coalition *gcbest* via *ptbest*.

Step 2. Put *m* ants on *n* robots randomly.

for *k* = 1 to *m*

{Initialize robot coalition consisting of robots which ants initially are located and delete these robots from J_k. Then calculate the capability vector B_{C_k} of each initial coalition.}

Step 3. for *k* = 1 to *m*

while ($B_{C_k} < B_t$)

{Choose a robot *j* according to probability $p_{ij}^{\ k}$ by Eq. (9) and put it into current coalition. Delete j form J_k. Increase the capability vector of coalitions. }

Step 4. for *k* = 1 to *m*

Coalition $C_0(k)$ formed by the *K-th* ant crossovers with *gcbest* thus produces $C_1^{'}(k)$, and then $C_1^{'}(k)$ crossovers with *pcbest* to produces $C_1^{''}(k)$. After the variation operator applied to $C_1^{''}(k)$, a new coalition $C_1(k)$ is formed. If $C_1(k)$ can perform the task, compute the fitness *Income*1 according to Eq. (8). If $Income1 > Income0$, the new value is accepted, otherwise keep $C_0(k)$ as the coalition of ant k. Update the values of *ptbest*, *pcbest*, *gtbest*, *gcbest*.

Step 5. Compute the coalition income $Inc(C_k)$ by Eq. (8) and save the best solution.

Step 6. Update the pheromone by Eqs. (10) & (11).

Step 7. Set $t = t + 1$, $NC = NC + 1$, $\Delta\tau_{ij} = 0$

Step 8. if （ $NC < NC_{\max}$ ）

$J_k = \{1, 2, \cdots, n\}$ ；

Goto Step 2.

Step 9. Output the optimal coalition and its income.

5.3 Forming robot coalition by quantum-inspired ant colony optimization

Quantum-Inspired evolutionary algorithm (QEA) was proposed by Kuk-Hyun Han (2002). It is based on the concept and principles of quantum computing (Grover, 1994) such as a quantum bit and superposition of states. QEA performs well even with a small population and without premature convergence as compared to the conventional genetic algorithm.

QEA is also characterized by the representation of the individual, the evaluation function, and the population dynamics. However, instead of using the binary, numeric and symbolic representation, QEA uses Q-bit as a probabilistic representation which is defined as the smallest unit of information. A Q-bit individual is defined by a string of Q-bits. The Q-bit individual has the advantage that can represent a linear superposition of states (binary solutions) in search space probabilistically. Thus, the Q-bit representation has a better characteristic of population diversity than other representations.

1. Encoding with Q-bits

A number of different representations can be used to encode the solutions onto individuals in evolutionary computation. QEA uses a new representation, called Q-bit, for a probabilistic representation. The representation is based on the concept of Q-bit; a Q-bit individual as well as a string of Q-bits are defined below.

Definition 1: A Q-bit is the smallest unit of information in QEA, which is defined with a pair of numbers (α,β) as

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

where $|\alpha|^2 + |\beta|^2 = 1$. $|\alpha|^2$ gives the probability that the Q-bit will be found in the '0' state and $|\beta|^2$ gives the probability that the Q-bit will be found in the '1' state.

A Q-bit may be in the '0' state, in the '1' state, or in a linear superposition of the two.

Definition 2: An individual Q-bit as a string of Q-bits is defined as

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{bmatrix}$$

where $|\alpha_i|^2 + |\beta_i|^2 = 1, i = 1, 2, ..., m$.

The Q-bit representation has the advantage that it is able to represent a linear superposition of states. If there is, for instance, a three-Q-bit system with three pairs of amplitudes such as

$$\left[\begin{array}{c|c|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & \frac{\sqrt{3}}{2} \end{array}\right]$$

Then the states of the system can be represented as

$$\frac{1}{4}|000\rangle+\frac{\sqrt{3}}{4}|001\rangle-\frac{1}{4}|010\rangle-\frac{\sqrt{3}}{4}|011\rangle+\frac{1}{4}|100\rangle+\frac{\sqrt{3}}{4}|101\rangle-\frac{1}{4}|110\rangle-\frac{\sqrt{3}}{4}|111\rangle$$

The above result means that the probabilities to represent the states $|000\rangle,|001\rangle,|010\rangle,|011\rangle$ $,|100\rangle,|101\rangle,|110\rangle,|111\rangle$ are 1/16, 3/16, 1/16, 3/16, 1/16, 3/16, 1/16, and 3/16, respectively. Therefore, the three-Q-bit system contains the information of eight states.

Evolutionary computing with Q-bit representation has a better characteristic of population diversity than other representations, since it can represent linear superposition of states probabilistically. Only one Q-bit individual is enough to represent eight states, but in binary representation at least eight strings, (000), (001), (010), (011), (100), (101), (110), and (111) are needed.

2. Quantum-Inspired Ant Colony Optimization

Wang & Li (2007) proposed a novel quantum genetic algorithm for TSP. The basic idea of quantum-inspired ant colony optimization is to make ants which have quantum characteristics, that is, every ant is a quantum individual and encoded by the probability of choosing cooperative robots instead of Q-bit. The QACO is added to the corresponding observation process and repairing process (Han, 2002).

The probability coding is defined as:

$$\left[\begin{array}{c} P_0 \\ P_1 \end{array}\right]$$

where $P_0+P_1=1$. The individual is denoted as:

$$q_k=\left[\begin{array}{c|c|c|c} p_{k_{10}} & p_{k_{20}} & p_{k_{j0}} & p_{k_{m0}} \\ p_{k_{11}} & p_{k_{21}} & p_{k_{j1}} & p_{k_{m1}} \end{array}\right] \tag{14}$$

where $k=1,2,\cdots,n$, $j=1,2,\cdots,m$, $P_{k_{j1}}=P_{ij}^{k}$, $P_{k_{j0}}=1-P_{k_{j1}}$. The t-th generation population of QACO is denoted as: $Q(t)=\left\{q_1^{t},q_2^{t},\cdots,q_n^{t}\right\}$. $P(t)=\left\{X_1^{t},X_2^{t},\cdots,X_n^{t}\right\}$, where X_k^{t} is the state of observing k-th individual, $X_k^{t}=\left\{x_{k1}^{t}x_{k2}^{t}\cdots x_{km}^{t}\right\}$, x_{kj}^{t} is either 0 or 1. When its value is 0, it means that robot j is not chosen while the value 1 means robot j is chosen.

The algorithm of QACO is given as follows:

Step 1. Initialize $t=0$, $NC=0$, $NC_{\max}=N$, $numAnt=m$, $numRobot=n$, $\Delta\tau_{ij}=0$, $\tau_{ij}(0)=\tau_0$

Step 2. Put m ants on n robots randomly

for $k=1$ to m

for $j=1$ to n

{ if ant k starts from robot j, then $P_{k_{j1}}=1$. According to Eq. (9), calculate

the probability of choosing cooperative robots, $P_{k_{j1}} = P_{ij}^{k}$, $P_{k_{j0}} = 1 - P_{ij}^{k}$ }

Step 3. Observe the individuals of $Q(t)$ and get the states $P(t)$.

Step 4. Check whether every state in $P(t)$ is a solution, if not then go to Step10 and repair it.

Step 5. According to Eq. (8), calculate the income $Inc(X_j^t)$ of X_j^t .

Step 6. Save the optimization coalition b and its income $Inc(b)$.

Step 7. Update the pheromone by Eqs. (10) & (11).

Step 8. Set $t = t + 1$, $NC = NC + 1$, $\Delta\tau_{ij} = 0$

Step 9. If $(\mathrm{NC} < \mathrm{NC}_{\max})$ and not keep evolving for a long time then go to Step 2, else output the optimization coalition and its income.

Step 10. Repair the state which is not a solution through repairing process. If states in $P(t)$ are all solutions, then go to Step 5.

6. High-level task allocation

The following parameters are introduced; m denotes the number of ants, each task is denoted as node 0, and the candidate robots or robot coalition are labelled as node 1 to n. The probability that ant k moves from node 0 to node j is formulated below:

$$\mathrm{p}_{\mathrm{ij}}^{\mathrm{k}} = \frac{\left[\tau_{\mathrm{ij}}(\mathrm{t})\right]^{\alpha}\left[1/\mathrm{cost}_{\mathrm{ij}}\right]^{\beta}}{\sum_{\mathrm{u}\in \mathrm{J}_{\mathrm{i}}}\left[\tau_{\mathrm{iu}}(\mathrm{t})\right]^{\alpha}\left[1/\mathrm{cost}_{\mathrm{iu}}\right]^{\beta}}, j \in J_i \tag{15}$$

where $\mathrm{J_i}$ is the set of candidate robots or robot coalition to task i, and $\mathrm{cost_{ij}}$ is the cost of robots or robot coalition to finish task i. If the task can be completed by a single robot, the cost is both the distance of the robot to the task and its ability consumption. Otherwise, the cost Cost(C,t) is the cost of robot coalition to complete the task. For each ant k, the first task node in the task list is the beginning point for the optimization. After ant k chooses an undertaker, it moves to next task to choose an undertaker for next task, and so on. When ant k has chosen undertakers for all tasks, one task allocation is finished. When all ants have completed a solution, one cycle is completed. The solution with the maximal income is the optimal solution, and then updates the intensity of pheromone according to Eq. (10).

However, $\Delta\tau_{ij}^{k}$ is defined as follows:

$$\Delta\tau_{\mathrm{ij}}^{\mathrm{k}} = \frac{\mathrm{Q}}{\sum_{\mathrm{k}=1}^{\mathrm{m}}\mathrm{cost}_{\mathrm{kj}}}, \quad Q \text{ is a constant} \tag{16}$$

The detailed task allocation algorithm is as follows:

Step 1. Initialization

Set $t = 0, NC = 0, \tau_{ij}(0) = \tau_0, \Delta\tau_{ij} = 0, numTask = s, numAnt = m, numRobot = n$, the capability requirement of each task, capability vector and cost vector of each robot.

Step 2. for i=1 to s

for k=1 to m do

{Ant k starts from the first task and determines whether the current task i is a tightly-coupled task. If it is, then go to step 7, else choose an undertaker from

J_i according to p_{ij}^k by Eq. (15) and calculate the income. Then, ant k moves to next task and repeats the above process until all tasks have been allocated to undertakers.}

Step 3. Calculate total income of the task allocation formed by each ant. Then, update the maximal income and the allocation schema.

Step 4. For ant k=1 to m do
Update the intensity of pheromone $\tau_{ij}(t+1)$ according to Eqs. (10) & (16).

Step 5. Set $t = t+1, NC = NC+1, \Delta\tau_{ij} = 0$.

Step 6. If (NC<NC_{max}) and (still keep evolving) then go to step 2
else output the allocation schema with the maximal income and stop the program.

Step 7. Call coalition formation algorithm ACO, PSACO and QACO to form a coalition for task i, then goto Step 2.

The allocating process is finished by the algorithm above. If current task is tightly-coupled, the high-level algorithm will call the low-level algorithm to form a coalition formation.

7. Deadlock elimination

Because robots are fully distributed in the system with equal status among them, it is likely to appear deadlock due to robots waiting each other at different task position. We employ a simple strategy to avoid the deadlock. Each robot has a task queue. Robots perform tasks in the same order as the tasks are allocated.

8. Simulation

In order to verify the effectiveness of proposed algorithms, we implement the algorithms in the TeamBots platform developed by Carnegie Mellon University and Georgia Institute of Technology. The implementation runs on a PC with M CPU 750, 1.8GHz Intel Pentium processor. Based on the transportation mission, there are some tasks in the environment. Some of them can be carried out by a single robot (loosely-coupled task) and the others must be completed by multiple robots (tightly-coupled task). Tables 1 and 2 list the capability of robots and task requirement.

According to Tables 1 and 2, we can find that tasks T1, T3, T6 and T9 must be completed by multiple robots. Simulation parameters are as follows:

The high-level ant colony size m=20, low-level colony size n=20, the maximal iteration number $NC_{max} = 500$, $Q = 1$, $rew(T_i) = 1000$, $\delta = \mu = \lambda = 0.5$, $\varpi_1 = \varpi_2 = 1$, $\alpha = 1.5$, $\beta = 2$, and $\rho = 0.9$

The task allocation algorithm was run 10 times. A comparison of three coalition formation algorithms is given in Fig. 3 and Table 3.

From Fig. 2 and Table 3, the following conclusions can be made:

1. The effectiveness of ACO is poor and it is easy to enter into premature convergence
2. The quality of PSACO is best, however, because each ant takes longer time than other two methods to finish a cycle, the runtime is relative long
3. QACO can find a good solution in a short time, so it is suitable for large-scale multi robots systems

Robot	Capacity	Cost	Robot	Capacity	Cost
R_0	1, 0, 1	1, 2, 1	R_8	3, 2, 1	2, 1, 3
R_1	1, 1, 1	1, 1, 2	R_9	3, 1, 1	2, 3, 2
R_2	2, 1, 2	2, 3, 1	R_{10}	2, 0, 1	1, 4, 3
R_3	1, 2, 1	3, 2, 1	R_{11}	1, 3, 3	2, 2, 1
R_4	0, 1, 1	1, 1, 1	R_{12}	2, 1, 3	1, 3, 1
R_5	1, 1, 2	2, 1, 1	R_{13}	0, 2, 1	3, 1, 2
R_6	0, 1, 1	3, 2, 3	R_{14}	1, 2, 3	4, 2, 1
R_7	2, 2, 1	3, 2, 1			

Table 1. Capacity vector and Cost Vector of Robots

Task	Capacity Required	Position	Task	Capacity requirement	Position
T_0	1 1 1	(13,10)	T_5	1 1 2	(15,0)
T_1	3 2 3	(0,0)	T_6	3 3 2	(0,-10)
T_2	1 2 1	(10,-10)	T_7	1 2 3	(20,-10)
T_3	3 3 1	(-10,10)	T_8	2 1 2	(20,20)
T_4	2 1 1	(-8,-3)	T_9	3 2 4	(20,0)

Table 2. Task requirement information

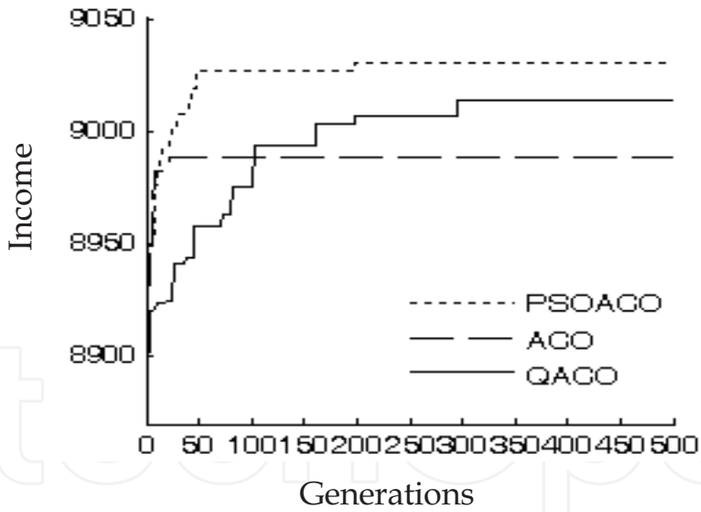


Fig. 3. Optimal evolution curves

Algorithm	Best (Generations)	Worst (Generations)	Average (Generations)	Average runtime（Sec）
ACO	9012	8953	8988	7.37
PSACO	9030	9030	9030	8.52
QACO	9022	8971	8997	3.75

Table 3. Results comparison among ACO, PSACO and QACO

9. Conclusion

This paper discusses the key issues of robot coalition formation. A task allocation mechanism based on swarm intelligence is proposed. This allocation method adopts a hierarchical architecture. At the high level, we employ Ant Colony Algorithm to find optimal allocations; each ant forms a task allocation so as to choose an undertaker for every task. At the low level, each ant forms a task-oriented robot coalition to perform a tightly-coupled task. ACO, PSACO and QACO are used to form the coalition. The algorithm is implemented in the TeamBots platform. Simulation results show that the proposed approaches can effectively achieve loosely-coupled and tightly-coupled task allocation in large-scale multi-robot systems. PSACO achieves the best solution, but its running time is the longest. On the other hand, although QACO is somewhat inferior to PSACO in the solution quality, its running time is only half of two other methods. Therefore, QACO is more suitable for the large-scale multi-robot system. Our future work is to improve the performance of algorithms and accelerate their convergence.

10. Reference

Botelho S. & Alami, R. (1999). M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '99),* pp. 1234–1239, Detroit, Michigan, USA, May 1999.

Chaimowicz, L. et al(2002). Dynamic Role Assignment for Cooperative Robots, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 293-298, Washington, America, May 2002.

Colorni, A.; Dorigo, M. & Maniezzo, V(1992). An investigation of some properties of an ant algorithm, *Proceedings of the Parallel Problem Solving from Nature Conference*, pp. 509-520, Brussels, Belgium, September,1992.

Dias, M. B. (2004). TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments. *PhD thesis*, Robotics Institute, Carnegie Mellon University, January 2004.

Ding,Y.Y. et al (2003). Multi-Robot Cooperation Method Based on the Ant Algorithm, *Robot*, Vol. 25, No.5, pp. 414-418, 2003.

Eberhart R. C. & Kennedy, J.(1995) A new optimizer using particles s warm theory, *Proceedings of the Sixth International Symposium on Micro Machine and Human Science,* pp. 39-43, Nagoya,Japan,1995.

Gerkey, B. & Mataric, M.J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, Vol. 23, No.9, pp.939-954, 2004.

Gerkey, B. & Mataric, M.J.(2002 a). Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, Vol. 18, No.5, pp. 758–786, 2002.

Gerkey, B. & Mataric, M.J. (2002 b). Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination, *In Proceedings of IEEE International Conference on Robotics and Automation*, pp. 464-469, Washington, America, May 2002.

Grover, L K. (1994). Algorithms for quantum computation：discrete logarithms and factoring. *In Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, New Jersey, November, 1994.

Han K. H. (2002). Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization, *IEEE Transactions on Evolutionary Computation*, Vol.6, No.6, pp.580-593, 2002.

Jiang, J.G. et al(2003).Solution to travelling agent problem based on improved ant colony algorithm, *Pattern Recognition and Artificial Intelligence*,Vol.16, No. 1, pp.6-12, 2003.

Kalra, N. & Martinoli, A. (2006). A Comparative Study of Market-Based and Threshold-based Task Allocation. *Proceedings of Distributed Autonomous Robotic Systems (DARS)*, Minnesota, USA, July 2006.

Liu, S. H. & Zhang Y.(2009). Multi-robot task allocation based on particle swarm and ant colony optimal , *Journal of Northeast Normal University,* Vol.41, No.4, pp. 68-72, 2009.

Liu, S. H. & Zhang Y.(2010). Multi-robot task allocation based on swarm intelligence, *Journal of Jilin University,* Vol.40, No.1, pp. 123-129, 2010.

Parker L. E. (1998). ALLIANCE：an architecture for fault tolerant multirobot cooperation, *IEEE Transactions on Robotics and Automation*, Vol.14, No.2, pp.220—240,1998.

Sanem, S.& Tucker, B.(2006). A Distributed Multi-Robot Cooperation Framework for Real Time Task Achievement, *Proceedings of Distributed Autonomous Robotic Systems*, Minnesota, USA, July 2006.

Smith R. G. (1980). The Contract Net Protocol: High level communication and control in a distributed problem solver. *IEEE Transaction on Computers*, 29, 12, pp. 1104-1113, 1980.

Tang, F. & Parker, L.E. (2005). ASyMTRe: Automated Synthesis of Multi-Robot Task Solution Through Software Reconfiguration, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1501-1508, Barcelona, Spain, May 2005.

Wang Y. P. & Li Y. H. (2007). A Novel Quantum Genetic Algorithm for TSP, *Chinese Journal of Computers*, Vol.30, NO.5, pp. 748-755, 2007.

Werger, B. & Mataric, M.J.(2000). Broadcast of local eligibility: Behavior based control for strongly cooperative multi-robot teams, *In Proceedings of Autonomous Agents,* pp. 21-22, Barcelona, Spain, June 2000.

Xia, N. ; Jiang, J.G. & Wei, X. (2005). Searching for Agent coalition for single task using improved ant colony algorithm, *Journal of Computer Research and Development*, Vol.42, No.5, pp. 734-739, 2005.

Yang, D. &Wang, Z.(2004). Improved Ant Algorithm for Assignment Problem, *Journal of Tianjing University*, Vol.37, No.4, pp. 373-376, 2004.

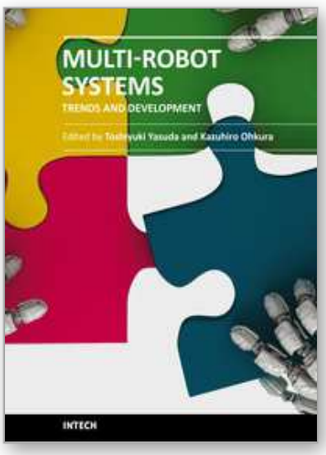
Zhang, D.D. et al (2007). Adaptive task assignment for multiple mobile robots via swarm intelligence approach, *Robotics and Autonomous Systems*, Vol. 55, No.7 pp. 572-588, 2007.

Zhang, Y. & Liu, S. H. (2008 a). Survey of Multi-Robot Task Allocation. *CAAI Transactions on Intelligence Systems,* Vol.3, No.2, pp.115-120, 2008.

Zhang, Y. & Liu, S. H. (2008 b). Large-scale Multi-robot Task Allocation Based on Ant Colony Algorithm, *Proceedings of Chinese Control and Decision Conference*, pp.2057-2062, Yantai, China, July 2008.

Zhang Y. & Liu, S. H. (2009). A Quantum-Inspired Ant Colony Optimization for Robot Coalition Formation, *Proceedings of Chinese Control and Decision Conference,* pp.632-637, Guilin, China, June 2009.

Zlot, R. et al (2002). Multi-Robot Exploration Controlled by a Market Economy, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3016-3023, Washington, DC, May 2002.



Multi-Robot Systems, Trends and Development

Edited by Dr Toshiyuki Yasuda

ISBN 978-953-307-425-2
Hard cover, 586 pages
Publisher InTech
Published online 30, January, 2011
Published in print edition January, 2011

This book is a collection of 29 excellent works and comprised of three sections: task oriented approach, bio inspired approach, and modeling/design. In the first section, applications on formation, localization/mapping, and planning are introduced. The second section is on behavior-based approach by means of artificial intelligence techniques. The last section includes research articles on development of architectures and control systems.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Shuhua Liu, Tieli Sun and Chih-cheng Hung (2011). Multi-Robot Task Allocation Based on Swarm Intelligence, Multi-Robot Systems, Trends and Development, Dr Toshiyuki Yasuda (Ed.), ISBN: 978-953-307-425-2, InTech, Available from: http://www.intechopen.com/books/multi-robot-systems-trends-and-development/multi-robot-task-allocation-based-on-swarm-intelligence

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen