Neural Comput & Applic (2013) 23:1909–1914
DOI 10.1007/s00521-012-1193-x

ISNN2012

Multi-robot task allocation using CNP combines with neural network

Quande Yuan · Yi Guan · Bingrong Hong · Xiangping Meng

Received: 30 April 2012 / Accepted: 18 September 2012 / Published online: 6 October 2012

Abstract Contract Net Protocol is a suitable method for multi-robot task allocation problems. However, it is difficult to find a function to evaluate robots' bids when each robot gives more than one bid price to reflect its different abilities. We propose a method to fuse these prices and to decide which robot is the successful bidder using a BP neural network. The experiment result shows that the method is effective.

Keywords Multi-robot · Task allocation · CNP · BP neural network

1 Introduction

Multi-robot system try to complete complex tasks, which is difficult, if not impossible, for only one robot, by means of cooperation of many relatively simple robots. In these systems, cooperation of the robots is a vital factor, while task allocation is a foundation technology of cooperation.

There are several methods for task allocation problem, which can be treated as an optimal allocation problem, linear programming, scheduling problem, combinatorial optimization problem, and so forth [1]. Most methods of Multi-robot task allocation are centralized in early times. There is a leader robot responsible for allocating tasks to other robots. These methods can get global optimal solutions some time, but there are some shortcomings. They are unavailable in unknown or dynamic environments.

As the development of AI technology, distributed task allocation methods have drawn attention of researchers because of their flexibility, robustness, and efficiency. Compared with centralized methods, these methods can get solutions more quickly although it cannot guarantee most of these solutions are optimal.

Contract Net Protocol, shorted for CNP, is an effective distributed task allocation method. It is a negotiation model proposed by Smith and Davis [2]. CPN solves resources conflict through allocating tasks to robots using market mechanisms.

In this method, the decision strategy of which robot is the successful bidder, thats to say assigning which robot, to complete tasks, is the key. It usually uses a decision-making function in CPN. However, it is difficult to find a function to evaluate robots bids when each robot gives more than one bid price to reflect its different abilities. In this paper, we introduce a BP neural network into CNP to fuse these prices and decide which robot is the successful bidder.

2 Relative works

There are many multi-robot task allocation methods already, including allocation methods based on behavior, swarm intelligence, methods based on linear programming, methods based on market mechanisms.

Q. Yuan · Y. Guan · B. Hong
School of Computer Science and Technology, Harbin Institute of Technology, 150001 Harbin, China

Q. Yuan (✉) · X. Meng
School of Electrical Engineering and Information Technology, Changchun Institute of Technology, 130012 Changchun, China
e-mail: yuanquande@gmail.com

ALLIANCE [3] and Broadcast of Local Eligibility (BLE) [4] are typical representatives of behavior-based task allocation method. They both allocate task through behavioral inhibition. Fault tolerance, real time, and robustness make these architectures be good choices for systems including heterogeneous robots performing missions composed of loosely coupled subtasks. They can gain sub-optimal solutions most of the time.

There are multi-robot systems perform missions cooperatively through indirect communication using swarm intelligence [5, 6]. Through designing interactive rules between robots, the robots behaviors will self-evolve and lead to the emergence of cooperative behaviors. These methods are suitable for systems including a large number of robots with low intelligence.

Mixed-Integer Linear Programming Solution can find the optimal solution, but it needs a centralized manager to manipulate information of all robots and tasks, which limit the scale of system and it is difficult to be extended.

The market mechanism methods are based on negotiation to allocate tasks. This kind of methods is suitable for the cooperative solving of distributed problems in medium-scale multi-robot systems. Contract net protocol is a classic method belonged to market mechanism methods. The idea of contract net protocol is inspired by the auction and contract activities in commercial. The process is: once a robot has a task, and it cannot finish by self, it decomposes the task to a sub-task set and tells other robots the existence of these sub-tasks. Then, this robot will act as manager of this task set. The robot who received announces will evaluate the task, if it can complete the sub-task, it bids for this task. The manager robot collects the bidding information and chose the best bidder as winner, contract with winner to complete the task. The manager supervises the process of task, and winner reports to the manager until finishing the task.

Contract net protocol has some shortcomings in solving multi-robot task allocation problems. It is difficult to make a decision when the robot publishes task and how to evaluate bid price value. So, in some applications, the CNP are extended, for example, Sandholm introduced ideas of layered structure, boundary cost computation [7], Chen introduced bid threshold [8], Tao introduced acquaintance coalition [9], into CNP.

We introduce Artificial Neural Network (ANN) into CNP. ANNs are powerful computational and modeling tools, which drawn more and more researchers' attention [10, 11, 12, 13, 14, 15]. The BP neural network is the most popular applied ANN. We use it to fuse input data for CNP.

3 Multi-robot patrol problem

3.1 Multi-robot system

We developed a heterogeneous multi-robot system named UMRS-1, which can be used as an experimental platform for robot patrolling or intrusion detection. The communication subsystem of UMRS-1 is a WLAN. Both infrastructure mode and Ad Hoc mode are supported by UMRS-1.

UMRS-1 is a distributed system from aspects of both logical and physical. The member robots act an equal role. There is only temporary task manager. Robots can join into or quit from system dynamically with auto configuration when the system is running. Every robot can launch a task, decompose this task into a set of subtasks and allocate these subtasks to other robots by mean of marketing.

The structure of a member robot is shown in Fig. 1. The robot OS layer is responsible for resource scheduling. Different robot maybe has different OS. Robot communication layer using network communications services of robot OS and UPnP architecture to talk with other robots semantically. We use an agent communication language KQML to encapsulate the content to talk.

Execution layer executes the task sent from decision layer. When the decision layer decides to execute a task, the robot will send a formal description of the task to execution layer. For example, execution layer received a task that moving to (10, 20), it will make a path plan based on current location and goal first, then it will move on the planed path.

3.2 Patrol problem

The task of robots in patrolling is to continuously travel in an area F to protect or supervise it. All robots distribute in F randomly to carry out the patrol task. The multi-robot system should be the response for the appearance of object (intruder). In this paper, robots should round up the intruder, which is one of the tasks in patrolling. This task cannot be finished by only one robot. It needs four robots to form

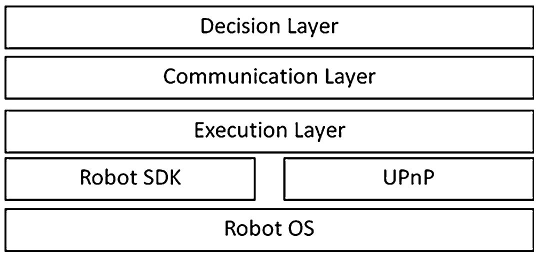

Fig. 1 Architecture of member robot

an alliance firstly. In practice, there are more than four robots patrolling in a big area generally. How to select robot automatically to form rounding alliance is a problem should be solved.

We use CNP to solve this problem. Once a robot finds intruder, it will act a role of task manager, decomposing task, organizing robots to round up intruder. After confirming intruder's running track, the finder will predict the intruder's position (x, y) at time t, and then, it will decide to round up this point. This task can be decomposed to four subtasks, which needs four robots heading for (x, y) from different directions. These robots should arrive at the four points nearby $(x-1, y)$, $(x, y-1)$, $(x+1, y)$, $(x, y+1)$ at time t. These points are named catch point. Once they arrived at the catch point, they will change the work mode from pursuit to follow. The four subtasks are shown in Fig. 2.

4 Task allocation using CNP combined with BP neural network

In UMRS-1, a robot r_i will act a role of manager to begin an auction when this robot has a task T while it cannot complete this task. Other robots that have no task to execute will wait for receiving new task.

4.1 Bid price evaluation based on BP neural network

There are different decision functions for different domain problems. Usually, they make decisions only based on single bid price. Because this single bid price is provided by bidder robot, there may be not so accurate and cannot provide enough information. For example, assume the speed of robot r_1 is 0.5 m/s, the distance between robot r_1 and goal is 2 m, while the speed of robot r_2 is 0.25 m/s, and

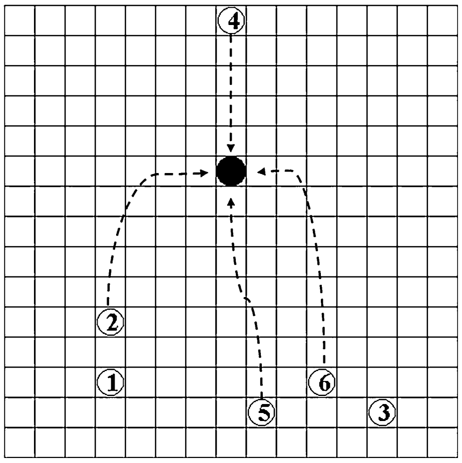

Fig. 2 Robots round up intruder

distance between robot r_1 and goal is 1 m, and they can arrive at goal using the same long time. Who will be the winner in bidding? Because the speed is important in pursuit, the former is a good choice. So bidder only provide how long can it arrive goal is not enough to make a decision. To solve this problem, we want each bidder robot to provide more than one bidder price; each price value reflects its ability of one aspect.

This is a problem about how to rate and classify the bidder robots and select the top ones. It is difficult to find a formula to rate while an elaborately designed artificial neural network (ANN) can do this. It is a good solution to find members of alliances using ANN. Here, we use BP ANN. We put bidding prices into ANN to fuse them. The output of ANN will indicate which robot is suitable to complete the task that robots bid for.

In this system, the information that bidder robot should input to ANN denoted as $B = \{B_1, \ldots, B_n\}$, where B_i is bid price of Robot r_i, $B_i = \{spl_i, ms_i, bl_i\}$. $spl_i \in SPL$, SPL is set of the shortest path length that robots from intruder. $ms_i \in MS$, MS is set of maximum speeds of robots. $bl_i \in BL$, BL is set of battery life of robots.

Here, we use a three-layer BP neural network. The BP neural network has 18 neurons in input layer, which means that the manager only accepts the first 6 biddings, 10 neurons in hidden layer and 6 neurons in output layer. In fact it is enough that the manager only considers robots with the fastest response in most time in hunting task.

The form of input data is a scalar like $(spl_1, ms_1, bl_1, \ldots, spl_6, ms_6, bl_6)$, where i is robots' ID. If robot r_i is busy, it will not participate in bidding, then the value of triples B_i will be $(max\ length\ of\ field,\ 0, 0)$.

The training set data can be obtained manually. Table 1 is part of training set. In Table 1, the network outputs evaluation of each bidder. By sorting the outputs in descending order, we can obtain the top four most suitable bidders. The last column of table showed the result of bidding. There are four winners labeled with 1 and the others with 0.

To avoid affect by different unit of input data, we should normalize these input data using Eq. (1):

$$\bar{X} = \frac{X_i - X_{i\min}}{x_{i\max} - X_{i\min}} \times d_1 + d_2 \tag{1}$$

where $X_{i\min}$, $x_{i\max}$ is the max and min values of input data. d_1 and d_2 are factor parameters, where

$$d_2 = \frac{1 - d_1}{2} \tag{2}$$

There are 6 output nodes. The form of output data is a scalar $O = (o_1, o_2, \ldots, o_n)$, where $o_i \in [0, 1]$.

In stage of back-propagation, we use formula 2 to update weights of nodes.

Table 1 Part of training set

Original bidding prices	Expected outputs	Results
{3.0, 1.0, 15, 7.0, 3.0, 23, 6.0, 3.0, 3, 9.0, 1.0, 38, 6.0, 1.0, 40, 5.0, 3.0, 24}	{0.7, 0.8, 0.4, 0.5, 0.6, 0.9}	{1, 1, 0, 0, 1, 1}
{5.0, 2.0, 49, 8.0, 1.0, 19, 13.0, 1.0, 41, 7.0, 1.0, 14, 14.0, 2.0, 4, 9.0, 3.0, 24}	{0.8, 0.6, 0.5, 0.7, 0.4, 0.9}	{1, 1, 0, 1, 0, 1}
{9.0, 2.0, 34, 13.0, 1.0, 13, 13.0, 3.0, 44, 5.0, 2.0, 39, 3.0, 1.0, 28, 12.0, 3.0, 35}	{0.5, 0.4, 0.7, 0.9, 0.6, 0.8}	{0, 0, 1, 1, 1, }
{7.0, 3.0, 26, 13.0, 2.0, 28, 4.0, 1.0, 20, 3.0, 3.0, 10, 10.0, 1.0, 18, 6.0, 1.0, 19}	{0.8, 0.7, 0.6, 0.9, 0.4, 0.5}	{1, 1, 1, 1, 0, 0}
{11.0, 2.0, 36, 13.0, 1.0, 19, 1.0, 1.0, 44, 6.0, 2.0, 31, 10.0, 1.0, 17, 13.0, 2.0, 16}	{0.7, 0.4, 0.9, 0.8, 0.5, 0.6}	{1, 0, 1, 1, 0, 1}
{6.0, 1.0, 13, 12.0, 2.0, 38, 9.0, 3.0, 25, 9.0, 2.0, 20, 11.0, 1.0, 17, 13.0, 3.0, 25 }	{0.5, 0.6, 0.9, 0.7, 0.4, 0.8}	{0, 1, 1, 1, 0, 1}

$$\begin{cases}\Delta W(n+1) = -\eta \frac{\partial E}{\partial W(n)} + \alpha \Delta W(n) \\ \Delta W(n+1) = -\eta \frac{\partial E}{\partial \theta(n)} + \alpha \Delta \theta(n)\end{cases} \quad (3)$$

where η is learning factor of network, α is power factor. We use mean squared error to present network error.

Figure 3 is the training performance of the decision network.

The converged weights of input layer are:

$$IW\{1,1\} = \left[\begin{array}{ccccccc}
0.9419 & 0.6975 & 2.8875 & 1.1041 & -0.0011 & 2.0501 & 0.4456 \\
0.2510 & 3.5463 & 0.6534 & -0.4174 & 2.4782 & 0.9060 & 1.0443 \\
1.4917 & 1.2116 & 0.2066 & 1.9332 & & & \\
0.1090 & -0.2042 & 0.1387 & 0.2951 & 0.0002 & -0.3035 & 0.1073 \\
0.0568 & 0.7150 & -0.0724 & -0.2775 & 1.0942 & -0.3108 & 0.9468 \\
0.8279 & 0.1038 & -0.3601 & 0.4238 & & & \\
-1.0909 & 0.0600 & -3.6795 & -1.8899 & -0.7430 & -4.8481 & -1.0521 \\
0.2017 & -3.9820 & -1.0519 & -0.7624 & -2.1374 & -1.7298 & -0.5732 \\
-2.3629 & -1.3364 & -0.7490 & -3.3277 & & & \\
0.7771 & 0.5850 & 2.7870 & 1.1568 & -0.2646 & 1.9796 & 0.2926 \\
-0.0482 & 2.8761 & 0.3338 & 0.1218 & 1.7589 & 0.8896 & 1.3825 \\
1.3408 & 1.0176 & 0.3404 & 1.6571 & & & \\
1.1892 & -0.2390 & 3.7301 & 1.4742 & -0.0881 & 3.2491 & 0.3864 \\
-0.1430 & 4.5822 & 1.0009 & 0.6359 & 3.1888 & 1.2921 & 1.3177 \\
2.2387 & 1.4884 & 0.6619 & 2.4057 & & & \\
-0.7312 & -0.5801 & -1.9681 & -1.3031 & -0.2651 & -3.6465 & -0.7136 \\
-0.5630 & -3.0316 & -0.8220 & -0.0468 & -2.4425 & -1.2324 & 1.1584 \\
-1.9330 & -1.2065 & -0.0839 & -2.4732 & & & \\
1.1294 & 0.1624 & 5.1597 & 1.8841 & -0.1006 & 4.9563 & 1.9132 \\
0.9909 & 6.0547 & 1.4192 & 0.0518 & 3.9287 & 1.8943 & \\
-0.7476 & 2.7603 & 1.9408 & 0.4822 & 4.7021 & & \\
-0.4436 & 0.0414 & -2.4154 & -0.9985 & -0.4669 & -2.1057 & -0.4783 \\
0.1100 & -1.8760 & -0.5601 & -0.4358 & -1.8095 & -0.9263 & -1.3296 \\
-1.9275 & -0.5391 & 0.3748 & -1.9153 & & & \\
0.1408 & 0.2828 & 0.6993 & 0.4368 & 0.7020 & 0.8224 & 0.4033 \\
0.6899 & 0.4193 & 0.1437 & -0.5565 & 2.2964 & 0.0993 & -0.6999 \\
2.2109 & 0.3281 & 0.2825 & 1.3767 & & & \\
1.1052 & 0.6107 & 5.5991 & 2.0105 & 0.4571 & 4.9807 & 1.7589 \\
0.1534 & 5.2424 & 1.3749 & -0.2085 & 3.5458 & 1.9809 & -0.2291 \\
2.8397 & 1.7923 & 0.0606 & 4.4110 & & &
\end{array}\right]$$

where bias vector is:

$$b\{1\} = \begin{bmatrix} -5.8642 & 5.0489 & 0.4395 & -5.1575 & 1.8807 & -5.7789 & 0.0167 \\ 4.2028 & 0.8021 & -0.6591 \end{bmatrix}'$$

The weights between hidden layer and output layer are:

$$LW\{2,1\} = \begin{bmatrix} -0.5857 & 0.4363 & 0.3188 & 0.4715 & 0.4901 & 0.3862 & -0.4679 \\ -0.3693 & 0.2930 & -0.5173 \\ 0.3684 & 0.1759 & -0.0544 & 0.0945 & -0.5155 & -0.4705 & 0.4231 \\ -0.2179 & -0.8472 & 0.7583 \\ 0.4379 & 0.0922 & -0.2209 & -0.9530 & 0.5191 & 0.2980 & 0.8242 \\ 0.2898 & -0.6008 & -0.1128 \\ -0.8751 & -0.4436 & -0.1917 & 0.8554 & 0.1121 & -0.9016 & 1.0288 \\ 0.1577 & -0.6457 & 0.3146 \\ 0.1340 & -0.5070 & -0.3921 & 0.1797 & -0.0255 & -0.0147 & 0.2452 \\ 0.1305 & -0.2857 & 1.1414 \\ 0.5518 & -0.5993 & 0.6071 & -0.6716 & 0.1891 & -0.1549 & -0.3257 \\ 0.0722 & 0.8770 & 0.6285 \end{bmatrix}$$

where bias vector is:

$$b\{2\} = [0.8656 \quad -0.5502 \quad 0.8101 \quad -0.6321 \quad -0.5084 \quad 0.5247]'$$

4.2 The procedure of task allocation

When robot r_i computed the position to round up intruder, it will begin bidding for each subtask. The procedure of bidding negotiation is as follows:

1. Publishing tender notice: Publish the information of task T_i, including ID of robot r_i, contract ID, goal position to go, deadline to response Ttimeout and the bid prices scalar.
2. Robot received the bidding information of task T_i will decide whether to bid based on its own state. The robot responses to the bidding documents if it is free and it can complete the task. Otherwise, it should reply to robot r_i that it cannot participate in this auction.
3. If the robot r_i received no bid, r_i will stop auction and turn to step (7).
4. Else, r_i will normalize the bid prices, put these scalar data into trained BP neural network, and publish the ID of successful bidder robot according to the outputs of neural network.
5. If the successful bidder robot wants to perform the task T_i, it should make a confirmation to robot r_i, else if it does not send the confirmation, robot r_i will think the successful bidder robot abandon this task and then turn to step (7).

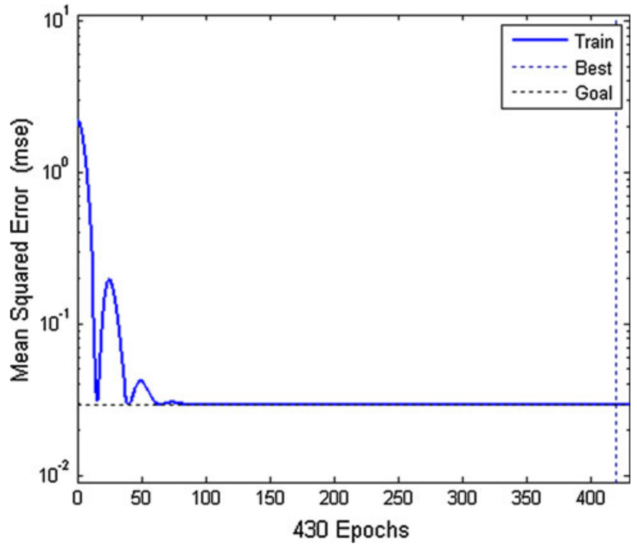

Fig. 3 MSE versus epochs

6. If robot r_i received confirmation notice, after all tasks finished bidding, it will command all of the robot accepted tasks into action, else turn to step (8).
7. A procedure of bidding negotiation finished.
8. If robot r_i decides to start a new auction, turn to step (1).

5 Further discussion

The problem that who will act as a manager to launch bidding when more than one robot finds the same intruder robot at almost the same time should be solved. In our system, we decide who will be the manager using time-stamps. Once a robot r_i finds an intruder, it will broadcast what he found including the location of the intruder and what time it found the intruder. The robot r_j who received the broadcast information will make a comparison. If robot r_j found the intruder later than robot r_i, it will do nothing, else, it will notice robot r_i through broadcasting.

6 Conclusion and future works

Task allocation is a foundation technology of multi-robot systems, especially of cooperative ones. In this paper, we let robot bidders provided more than one bid price; these values reflect the different abilities of robots. It is difficult to construct a decision function to evaluate these bid prices, so we propose a method of using a BP neural network to fuse these prices and decide which robot is the successful bidder. In our experiment, the intruder can be rounded up quickly using this method.

Acknowledgments This work is supported by National Natural Science Foundation of China grant NO. 60974055, 61075077 and 61075076, Department of Education of Jilin Province of China grant NO.2011242.

References

1. Gerkey BP (2003) On multi robot task allocation. Dissertation, University of Southern California
2. Smith RG (1980) The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Trans Comput C-29:1104–1113
3. Parker LE (1998) ALLIANCE: an architecture for fault tolerant muhirobot cooperation. IEEE Trans Rob Autom 14:220–240
4. Gerkey B, Mataric MJ (2000) Broadcast of local eligibility behavior based control for strongly cooperative multi-robot teams. In: Proceedings of the fourth international conference on Autonomous agents doi:10.1145/336595.336621
5. Ding YY, He Y, Jiang JP (2003) Multi-robot cooperation method based on the ant algorithm. Robot 25:414–418
6. Oliveira et al. (2004) A swarm based approach for task allocation in dynamic agents organizations. In: Proceedings of the third international joint conference on autonomous agents and multi-agent systems. New York 3:1252–1253
7. Sandholm T, Lesser VR (1997) Coalition among computationally bounded agents. Artif Intell 94:99–137
8. Chen XG (2004) Further extensions of FIPA contract net protocol:threshold plus DoA. In: Proceeding 2004 ACM symposium on applied computing. doi:10.1145/967900.967914
9. Tao HJ, Wang YD (2006) A multi-agent negotiation model based on acquaintance coalition and extended contract net protocol. J Comp Res Dev 43:1155–1160
10. Zhang HG, Quan YB (2001) Modeling, identification, and control of a class of nonlinear systems. IEEE Trans Fuzzy Syst 9:349–354
11. Sun QY, Li ZX, Yang J, Luo YH (2010) Load distribution model and voltage static profile of smart grid. J. Cent. South Univ. Tech 17:824–829
12. Zhang HG, Wang ZS, Liu D (2008) Global asymptotic stability of neural networks with multiple time-varying delays. IEEE Trans Neural Netw 19:855–873
13. Zhang HG, Wang ZS, Liu D (2008) Robust stability analysis for interval Cohen–Grossberg neural networks with unknown time-varying delays. IEEE Trans Neural Netw 19:1942–1955
14. Zheng CD, Zhang HG, Wang ZS (2009) Novel delay-dependent criteria for global robust exponential stability of delayed cellular neural networks with norm-bounded uncertainties. Neurocomputing 72:1744–1754
15. Dong M, Zhang HG, Wang YC (2009) Dynamics analysis of impulsive stochastic Cohen-Grossberg neural networks with Markovian jumping and mixed time delays. Neurocomputing 72:1999–2004