A Kind of Negotiation Policy for Resource Allocation in Distributed Manufacturing Environment

LV Cixing¹,², ZHU Yunlong¹, YIN Chaowan¹
1 Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China; 2 Graduate School of the Chinese Academy of Sciences, Beijing, China
E-mail: smale@sia.cn, ylzhu@sia.cn, ycw@sia.cn

Abstract

A kind of negotiation policy is proposed for the allocation problem solved by agent-based system in distributed manufacturing environment, in which subtasks in the same task have in-tree precedence constraints and flexible routes. With the formulation of the problem and that of combinatorial auction solution, the design, the procedure and the implementation of combinatorial auction are given. The final part of the paper gives numerical test to demonstrate the policy to be feasible and efficient.

Key words: combinatorial auction, multi-agent system, scheduling, negotiation policy

1. Introduction

Distributed manufacturing is becoming increasingly important since it can respond rapidly to market changes and make resource sharing more efficient among manufacturing partners [1]. In this environment, partners and resources may be located at different geographical locations, and scheduling distributed resources to fulfill production tasks with constraints is a vital task. As one of the important research topics in distributed intelligence, multi-agent system (MAS) has played a key role in the development of distributed manufacturing scheduling system [2-6].

To cope with different information and decision-making in the distributed manufacturing paradigm, effective and efficient coordination among autonomous entities is becoming ever important. As one kind of efficient approach of coordination, negotiation policy is one of the opportunities and challenges in the distributed manufacturing scheduling system [4]. Currently, the Contract Net Protocol (CNP) or its modified versions are the most used negotiation policy. The CNP allocates resources dynamically and balances the workload of system. However, the CNP does not accommodate future events and the performance is unpredictable. To overcome these problems, researchers have proposed various improvements to the basic CNP [4]. But the CNP and its modified versions do not consider the combinatorial requirement of tasks for resources. In [7] and [8], the combinatorial requirement of tasks for resources is considered by means of combinatorial auction when allocating resource. In their problem, the tasks have strict precedence constraints and fixed routes.

In the paper, a kind of Combinatorial Auction Based nEgotiation Policy (CABEP) is proposed for resource allocation problem in which subtasks of the same task have in-tree precedence constraints and flexible routes.

2. Problem formulation and that of combinatorial auction solution

2.1 Resources allocation problem formulation

The problem considered consists of scheduling N tasks on M resources in order to minimize the cost, i.e., the tardiness of the task, etc. The task is partitioned into a set of subtasks \( O(i,j) \) in advance, and each subtask \( O(i,j) \) can be performed on one of a set of alternative resources. The precedence constraints between subtasks of a same task form an in-tree. The variables used presented below:

\[ O(i,j) \text{ — The } j \text{ th subtask of the task } i; \]
\[ T \text{ — The length of the planning horizon; } \]
\[ M \text{ — The number of resources; } \]
\[ t \text{ — The index of time slot; } h \text{ — The index of resource; } \]
\[ \lambda_{ht} \text{ — The price for time slot } t \text{ for resource } h. \]

---

¹ The research is supported by National Basic Research Program of China (No. 2002CB312204) and National Nature Science Foundation of China (No. 70431003)
resource \( h \); \( N \) — The number of tasks; \( n_i \) — Number of subtask of task \( i \); \( p_{i,j,k} \) — processing time of \( O(i,j) \) if it is processed on \( h \in H_{i,j} \), otherwise 0; \( \omega_i \) — Weight of task \( i \); \( H_{i,j} \) — Resource set capable of processing \( O(i,j) \); \( I_{i,j} \) — Set of subtasks preceding immediately \( O(i,j) \); \( TD_{i,j} \) — Cost for task \( i \) to complete in time slot \( t \); \( X_{i,j,h} = 1 \) if \( O(i,j) \) is complete at time \( t \), 0 otherwise; \( Y_{i,j,h} = 1 \) if \( O(i,j) \) is processed on resource \( h \), 0 otherwise. Problem is defined as follows:

\[
J: \min \sum_{j=1}^{n} \sum_{i=1}^{N} \omega_i TD_{i,j} \tag{1}
\]

\[
\sum_{i=1}^{N} \sum_{j=1}^{n} t_i X_{i,j,h,} \leq \sum_{i=1}^{N} \sum_{j=1}^{n} t_i Y_{i,j,h,} - \sum_{j=1}^{n} p_{i,j,k} Y_{i,j,h,}, \quad \forall i, j, l \in I_{i,j} \tag{2}
\]

\[
\sum_{i=1}^{N} \sum_{j=1}^{n} X_{i,j,h,} + \sum_{i=1}^{N} \sum_{j=1}^{n} \sum_{k=1}^{M} X_{i,j,k} \leq 1, \forall h, t \tag{3}
\]

\[
\sum_{i=1}^{N} \sum_{j=1}^{n} t X_{i,j,h,} \geq \sum_{j=1}^{n} p_{i,j,k} Y_{i,j,h,}, \quad \forall i \tag{4}
\]

Constraint set (2) is precedence constraints for each task. Constraint set (3) is capacity constraints which say that each resource can perform at most one subtask at a time. Constraint set (4) is the arrival time constraints that ensure that the first subtask cannot be completed before the task has been launched for at least the time for processing the first subtask.

### 2.2 Auction design

The time window considered is divided into \( T \) time slots equally. The price of each time slot is \( \lambda \). All resource agents delegate their time slots to a mediator who acts as auctioneer, task agents act as bidders and auction items are the time slots of the resources and defined as a set of pairs (resource, timeslot). Triple \( B_{i,j} \left( m_{i,j}, t_{i,j}, l_{i,j} \right) \) presents requirement of \( O(i,j) \) for \( j \) timeslots, \( m_{i,j} \) presents resource, \( t_{i,j} \) presents start index of time slot, \( l_{i,j} \) presents the number of time slots which means the processing time on the resource. Therefore, the bid of task agent \( i \) can be presented as \( B_i = \{ (B_{i,j}, \bar{B}_{i,j}, p_i) \} \), \( p_i \) is the payment for the bid. Each task competes for time slots with its precedence constraint, process requirement and local utility function. The auctioneer adjusts the price of each time slot to reduce the resource conflicts among the bidder tasks until result of auction is obtained. In the end the resource and complete time for each subtask are determined.

Lagrangian Relaxation approach is adopted to design combinatorial auction. Relaxing the resources capacity constraints (3), the Lagrangian Relaxation problem is obtained:

\[
LR_\lambda = \min \sum_{j=1}^{n} \sum_{i=1}^{N} \omega_i TD_{i,j} + \sum_{j=1}^{n} \sum_{i=1}^{N} \sum_{h=1}^{M} \lambda_{i,j,h} \left( \sum_{j=1}^{n} X_{i,j,h} Y_{i,j,h} - 1 \right) \tag{5}
\]

\[
LR_\lambda = \min \sum_{j=1}^{n} \sum_{i=1}^{N} LR_{i,j} - \sum_{j=1}^{n} \sum_{i=1}^{N} \lambda_{i,j,h} \tag{6}
\]

\[
LR_{i,j,h} = \min \left( \sum_{j=1}^{n} \omega_i TD_{i,j} + \sum_{h=1}^{M} \lambda_{i,j,h} \sum_{j=1}^{n} X_{i,j,h} Y_{i,j,h} \right) \tag{7}
\]

\[
\sum_{j=1}^{n} \sum_{i=1}^{N} t X_{i,j,h} \leq \sum_{j=1}^{n} \sum_{i=1}^{N} t X_{i,j,h} Y_{i,j,h} - \sum_{h=1}^{M} p_{i,j,k} Y_{i,j,h,}, \quad \forall j, l \in I_{i,j} \tag{8}
\]

Subgradient optimization procedure has been most frequently used to improve the Lagrangian lower bound. It starts with an initial value and the method generates a sequence \( \lambda'_{i,j} \) over the iterations \( r \) by the rule:

\[
\lambda'_{i,j} = \max \left\{ 0, \lambda_{i,j} + s_r G_{i,j} \right\} \tag{9}
\]

\[
G_{i,j} = \sum_{j=1}^{n} \sum_{i=1}^{N} \sum_{h=1}^{M} X_{i,j,h} Y_{i,j,h} - 1 \tag{10}
\]

\[
s_r = \alpha \frac{UB - LR_{\lambda}}{\sum_{r=1}^{\infty} (G_{i,j}')} \tag{11}
\]

In (10) and (11), \( G_{i,j} \) is the subgradient of the capacity constraint of resource \( h \) at time period \( t \) defined by the optimal solution \( LR_{\lambda} \), \( \alpha \) is a scalar satisfying \( 0 < \alpha \leq 2 \), and \( UB \) is a target upper bound value for Lagrangian dual which can be updated over the iterations.
2.3 Auction procedure

Auction procedure is presented as follows:
1. After initializing the prices $\lambda_{h,t} = 0$ and the iteration counter $c = 1$, the auctioneer announces an auction.
2. Bid construction
   With the prices $\lambda_{h,t}$, each bidder constructs bid $B_i$ by dynamic programming approach[9] and submits it.
3. Revenue calculation
   With the prices and bids submitted, auctioneer computes the revenue:
   \[ R_c = \sum_{r=1}^{\mathcal{R}} LR_{h,r} - \sum_{r=1}^{\mathcal{R}} \sum_{h=1}^{\mathcal{H}} \lambda_{h,t} \]
4. A market clearing mechanism is used to eliminate the positive excess demand. If $(UB - R) < \epsilon$ or the maximum number of iterations is reached, go to 7; else go to 5.
5. Auctioneer updates the prices $\lambda_{h,t}$ by (10).
6. Increase the iteration counter $c = c + 1$, and go to 2.
7. According to the result of market clearing, timeslots are assigned to corresponding bidders.

2.4 Distributed rule implementing combinatorial auction

An agent executes its local actions with local constraints and coordinates with other agents to achieve global goals. Agents in the system are defined as follows:
- **Domain Manage Agent (DMA):** It acts as a bridge between the system and other system and manages the lifecycle of the task agents and resource agents. When launching the task/resource agent, the task/resource agent template is used.
- **Resource Mediator Agent (RMA):** It acts as the mediator of RA and the auctioneer in the auction;
- **Control Agent:** It deals real-time resources allocation in the system in emergency situation;
- **Resource Agent (RA):** A resource agent represents the current state of a resource. All RAs have similarity structure but different attributes and capacities according to the resource. It delegates its resources to the RMA for auction.
- **Task Agent (TA):** A task agent is launched temporarily by the DMA for a special task. All TAs have similarity structure but different attributes. It acts as a bidder in the auction.

According to the design of auction and the agents in the system, the rule implementing combinatorial auction is given:

Message and variable: Resource breakdown—Br, Emergent Task—E; Task cancel—C, Iteration count—Ic, Step size—Ss, Initial step size—Ss0, Message for start auction—Sa, All bids are submitted—Pb, Message for stop auction—Pa, Auction is completed—Af, Announce bid—P, Market clearing is made—Mc, Auction result is adopted—R.

Logic symbol: —NOT, —AND, —OR.
Predication: $\exists$ —Exist, NOTIFY, REGISTER, ANNOUNCE, UPDATE, SUBMIT, CLEAR.
- **Start/stop auction rules**, which decides the auction’s start or stop.
  - Start auction rule, IF $\exists(Br \ E \ C) \ THEN$ NOTIFY RA (REGISTER) AND RMA (initiate auction).
  - Stop auction rule, IF $\exists(Br \ E \ C) \ THEN$ NOTIFY RA (stop auction) AND RMA (using reactive rule to schedule production task).
  - Result adoption rule, IF $(Ct > Ct0) \ (Ss < Ss0) \ THEN$ NOTIFY RMA (adopt auction result).
- **Resource mediation rule.**
  - Register resource agent rule, IF $\exists Sa$ THEN RA REGISTER.
- **Auctioneer rule**, RMA acts as auctioneer.
  - Announce bid rule, IF $\exists P \ (Sa \ Ct \ Ct0 \ Ss \ Ss0 \ Pa) \ THEN$ ANNOUNCE (price of auction item) TO TAs.
  - Market clearing rule, IF (NUMBER(TAs) == NUMBER(Bids received)) THEN CLEAR market. (The dynamic programming approach was adopted for market clearing [11]).
  - Update price rule, IF $\exists(Mc \ P) \ THEN$ UPDATE (price of auction item).
- **Pause auction rule**, IF $\exists Pa$ THEN NOTIFY MA (iteration counts and step size).
  - Construct final solution rule, IF $\exists Af \ R \ THEN$ NOTIFY TA (final solution).
  - Construction bid rule, task agents act as bidder constructing bids.
    - Construct bid rule, IF $\exists P \ THEN$ SUBMIT ($B_i$) TO RMA.

3 Test results

Numerical testing has been performed to demonstrate the negotiation policy to be feasible and efficient. Test parameter setting was described here. Test data in benchmark in [2] was adopted to compare the negotiation policy with dispatching rules. In Figure 2, the results of SPT and EDD was cited from [2], in
which distributed rule was used. And CABEP denotes
the tests result after adopting CABEP. Figure 2 shows
that the maximum flow time is 65, average flow time is
45, and average delay rate is 6%, which are
comparative reduced compare to SPT and EDD
distributed rule.

Figure 1. Test result of example 1

4 Conclusion

Combinatorial auction based negotiation policy, is
used to solve distributed scheduling problem in the
distributed manufacturing environment. With the
negotiation policy, autonomous agents solve local
problem with local constraints, at the same time, the
distributed resources are scheduled in the global view
in the time window by the means of the combinatorial
auction. The test result shows that the negotiation
policy is effective and efficient.

5 References

[1] Zhang Y. F., Jiang P. Y. and Zhou G. H., GA-
driven part e-manufacturing scheduling via an online e-
service platform. Integrated Manufacturing Systems,
2003, 14, pp. 575-585
Agent and distributed ruler based approach to
production scheduling of agile manufacturing systems.
International Journal Computer Integrated
Manufacturing, 2003,16, pp.81-92
distributed-rules based agile production scheduling
approach. Control Theory & Application (in Chinese),
2004, 21, pp.526-530
Scheduling Using Intelligent Agents. IEEE Intelligent
[5] Rabelo R. J., Camarinha-Matos L. M. and
systems for intelligent manufacturing: a state-of-the-art
survey. Knowledge and Information Systems,1999,1,
pp.129-156
combinatorial auction and Lagrangean relaxation for
distributed resource scheduling. IIE Transactions.
1999, 31, pp.813 –826
distributed scheduling in a dynamic job shop
environment. International Journal of Production
Research, 2002, 40, pp.1173-1191
[9] Haoxun Chen, Chengbin Chu and Jean-Marie
Proth: An improvement of the Lagrangean relaxation
approach for job shop scheduling: a dynamic
programming method, IEEE transactions on robotics
and automation, 1998,14, pp.786-795