

Distributed Lagrangean Relaxation Protocol for the Generalized Mutual Assignment Problem

Katsutoshi Hirayama
Kobe University
5-1-1 Fukaeminami-machi, Higashinada-ku
Kobe 658-0022, Japan
hirayama@maritime.kobe-u.ac.jp

ABSTRACT

We present a new formulation of distributed task assignment, *Generalized Mutual Assignment Problem* (GMAP), which is derived from a typical NP-hard combinatorial optimization problem that has been studied for many years mainly in the operations research community. Then, to solve the GMAP, we introduce a novel distributed solution protocol using *Lagrangean decomposition* and *distributed constraint satisfaction*, where the agents solve their individual optimization problems and coordinate their locally optimized solutions through a distributed constraint satisfaction technique. Next, to produce quick agreement between the agents on a feasible solution with reasonably good quality, we provide a parameter that controls the range of “noise” mixed with increment/decrement in a Lagrangean multiplier and report our experimental results indicating that the parameter may allow us to control tradeoffs between the quality of a solution and the cost of finding it.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*Coherence and coordination*

General Terms

Algorithms

Keywords

distributed combinatorial optimization

1. INTRODUCTION

The distributed task assignment problem, which concerns assigning a set of jobs to multiple agents, has been one of the main research issues from the early days of distributed problem solving [9]. Recently, several studies have been reported in which they formalize the problem as the optimization

problem to be solved by some general solution protocol. [3, 6, 7].

In this paper, we present a new formulation of distributed task assignment which is derived from the *Generalized Assignment Problem* (GAP). The GAP is a typical NP-hard combinatorial optimization problem that has been studied for many years mainly in the operations research community [1, 8]. The goal of the GAP is to find an optimal assignment of jobs to agents such that a job is assigned to exactly one agent and the assignment satisfies all of the resource constraints imposed on individual agents.

To adapt the GAP for a distributed context, we first introduce the *Generalized Mutual Assignment Problem* (GMAP). The GMAP is the distributed problem in which each agent in the GAP has a set of jobs and tries to achieve the goal of the GAP. To put it another way, the agents themselves try to generate an optimal solution of the GAP from a non-optimal (and sometimes infeasible) solution of it. Then, to solve the GMAP, we introduce a novel distributed solution protocol using *Lagrangean decomposition* [4] and *distributed constraint satisfaction* [10]. In this protocol, the agents solve their individual optimization problems and coordinate their locally optimized solutions through a distributed constraint satisfaction technique. One important feature of this protocol is that, unlike other optimization protocols for distributed task assignment, it is a pure distributed protocol where neither the server nor the coordinator exists.

2. FORMALIZATION

A GMAP instance consists of multiple agents each having a finite set of jobs that are to be assigned. The agents as a whole solves the following *integer programming problem* denoted as $\mathcal{GAP}$.

$$\begin{aligned} \mathcal{GAP} \quad & (\text{decide } x_{kj}, \forall k \in A, \forall j \in J) : \\ \text{max.} \quad & \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} \\ \text{s. t.} \quad & \sum_{k \in A} x_{kj} = 1, \quad \forall j \in J, \\ & \sum_{j \in J} w_{kj} x_{kj} \leq c_k, \quad \forall k \in A, \\ & x_{kj} \in \{0, 1\}, \quad \forall k \in A, \forall j \in J, \end{aligned} \tag{1}$$

where $A = \{1, \dots, m\}$ is a set of agents; $J = \{1, \dots, n\}$ is a set of jobs; p_{kj} and w_{kj} are a profit and an amount of resource required, respectively, when agent k selects job j ;

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

c_k is a capacity, an amount of available resource, of agent k . Also, x_{kj} is a decision variable whose value is set to 1 when agent k selects job j and 0 otherwise. A goal of the agents is to find a job assignment that maximizes a total sum of profits such that (1) each job is assigned to exactly one agent (denoted as *assignment constraints*), (2) a total amount of resource required for each agent does not exceed its capacity (denoted as *knapsack constraints*), and (3) each job is assigned or not assigned to an agent (denoted as *0/1 constraints*). We refer to the maximal total sum of profits as the *optimal value* and the assignment that provides the optimal value as an *optimal solution*.

We can obtain the following Lagrangean relaxation problem $\mathcal{LGAP}(\mu)$ by dualizing assignment constraints of $\mathcal{GAP}$ [2].

$$\begin{aligned} \mathcal{LGAP}(\mu) \quad & \text{(decide } x_{kj}, \forall k \in A, \forall j \in J) : \\ \max. \quad & \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} + \sum_{j \in J} \mu_j \left(1 - \sum_{k \in A} x_{kj} \right) \\ \text{s. t.} \quad & \sum_{j \in J} w_{kj} x_{kj} \leq c_k, \forall k \in A, \\ & x_{kj} \in \{0, 1\}, \forall k \in A, \forall j \in J, \end{aligned}$$

where μ_j is a real-valued parameter and generally called a *Lagrangean multiplier* for (the assignment constraint of) job j . Note that a vector $\mu = (\mu_1, \mu_2, \dots, \mu_n)$ is called a *Lagrangean multiplier vector*. Based on the idea of *Lagrangean decomposition* [4], we can divide this problem to produce a set of subproblems $\{\mathcal{LGMP}_k(\mu) | k \in A\}$, where each subproblem $\mathcal{LGMP}_k(\mu)$ of agent k is as follows.

$$\begin{aligned} \mathcal{LGMP}_k(\mu) \quad & \text{(decide } x_{kj}, \forall j \in R_k) : \\ \max. \quad & \sum_{j \in R_k} p_{kj} x_{kj} + \sum_{j \in R_k} \mu_j \left(\frac{1}{|S_j|} - x_{kj} \right) \\ \text{s. t.} \quad & \sum_{j \in R_k} w_{kj} x_{kj} \leq c_k, \\ & x_{kj} \in \{0, 1\}, \forall j \in R_k, \end{aligned}$$

where R_k is a set of jobs that may be assigned to agent k and S_j is a set of agents to whom job j may be assigned. We can assume $S_j \neq \emptyset$ because the job with an empty S_j does not have to be considered in the problem. When we consider that agent k decides the value of x_{kj} for $\forall j \in R_k$ (in other words, the agent that may be assigned job j has a right to decide whether it will undertake the job or not), agent k is able to solve $\mathcal{LGMP}_k(\mu)$ completely by itself since $\mathcal{LGMP}_k(\mu)$ includes only k 's decision variables.

On the relation between the entire problem and a set of subproblems, there exist the following properties which are very useful in designing a distributed solution protocol for the GMAP.

PROPERTY 1. For any values for μ , a total sum of the optimal values of $\{\mathcal{LGMP}_k(\mu) | k \in A\}$ provides an upper bound of the optimal value of $\mathcal{GAP}$.

PROPERTY 2. For any values for μ , if all of the optimal solutions to $\{\mathcal{LGMP}_k(\mu) | k \in A\}$ satisfy the assignment constraints, $\sum_{i \in A} x_{ij} = 1, \forall j \in J$, then these optimal solutions constitute an optimal solution to $\mathcal{GAP}$.

3. PROTOCOL

Property 2 prompted the development of a distributed solution protocol, which is called the *distributed Lagrangean relaxation protocol*, where the agents start with $t = 0$ and $\mu^{(0)} = (0, \dots, 0)$; then, they alternate the following series of actions (called a *round*) in parallel until all of the assignment constraints are satisfied.

1. Each agent k finds an optimal solution to $\mathcal{LGMP}_k(\mu^{(t)})$ (solves the *primal problem*).
2. Agents exchange these solutions with their neighboring agents.
3. Each agent k updates the Lagrangean multiplier vector from $\mu^{(t)}$ to $\mu^{(t+1)}$ (solves the *dual problem*) and increases t by one.

It must be noted that the overall behavior of this protocol is similar to the *distributed breakout algorithm* for solving the DisCSP [5]. We describe the key ideas of the protocol in this section.

3.1 Neighborhood and Communication Model

In order to solve primal and dual problems, agent k needs, for each job j in R_k , the values of Lagrangean multiplier $\mu_j^{(t)}$, the size of set S_j , and the decision variables of the related agents (i.e., $\{x_{ij} | i \in S_j, i \neq k\}$). The value of $\mu_j^{(t)}$ is locally computed as we will explain later, and the size of set S_j is given to agent k as prior knowledge. On the other hand, the values of the decision variables are obtained through communication with a set of agents denoted as $\bigcup_{j \in R_k} S_j \setminus \{k\}$. We refer to this set of agents as agent k 's *neighbors* and allow an agent to communicate only with its neighbors. Thus, the protocol assumes a peer-to-peer message exchange model, in which a message is never lost and, for any pair of agents, messages are received in the order in which they were sent.

3.2 Primal Problem

Once agent k determines the values of the associated Lagrangean multipliers, agent k searches for an optimal solution to $\mathcal{LGMP}_k(\mu^{(t)})$ to determine the values of its own decision variables $\{x_{kj} | j \in R_k\}$. This search problem is equivalent to the *knapsack problem* whose goal is to select the most profitable subset of R_k such that the total resource requirement of the subset does not exceed c_k . In the problem, for each job j in R_k , the profit is $p_{kj} - \mu_j^{(t)}$, and the resource requirement is w_{kj} .

After finding an optimal solution, agent k sends the solution to its neighbors via an *assign* message.

3.3 Dual Problem

After receiving all of the current solutions of neighbors, agent k solves the dual problem by updating the related Lagrangean multipliers. In this work we use a *subgradient optimization method*, a well-known technique for systematically updating a Lagrangean multiplier vector [2]. In the method, under the current values of the related decision variables, agent k first computes *subgradient* g_j for an assignment constraint of each job $j \in R_k$ as follows:

$$g_j = 1 - \sum_{i \in S_j} x_{ij}.$$

Then, using $|S_j|$ and *step length* $l^{(t)}$, which may decay at rate r ($0 < r \leq 1$) as the number of rounds t proceeds (i.e., $l^{(t+1)} \leftarrow rl^{(t)}$), agent k updates the multiplier of job j as

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - l^{(t)} \frac{g_j}{|S_j|}. \quad (4)$$

Notice that since multiplier μ_j is attached to job j all of the agents in S_j must agree on a common value for it. If the agents in S_j assign different values to μ_j , then neither property 1 nor 2 holds any more. To set such a common value, we give all of the agents a common initial value for μ , a common value for initial step length $l^{(0)}$, and a common value for decay rate r as their prior knowledge, and prohibit each agent from working at round $t + 1$ until it receives all of the assign messages issued from its neighbors at round t . By doing this, instead of implementing explicit communication among S_j , we can force agents to automatically set a common value to a Lagrangean multiplier.

3.4 Convergence Detection

The protocol should be terminated when all of the optimal solutions to $\{\mathcal{LGMP}_k(\mu^{(t)}) \mid k \in A\}$ satisfy all of the assignment constraints, but it is not so simplistic to detect this fact because no agent knows the entire solving processes. In this protocol we use a termination detection procedure of the distributed breakout algorithm [5] to detect this fact.

3.5 Convergence to Feasible Solution

Unfortunately, the protocol may be trapped in an infinite loop depending on the parameter values in rule (4). If that case, the protocol must be forced to terminate after a certain number of rounds, but it has not yet discovered any feasible solution on the way to an optimal solution. Therefore, we make rule (4) stochastic so that the protocol can break an infinite loop and produce quick agreement between the agents on a feasible solution with reasonably good quality. More specifically, we let the agents in S_j assign slightly different values to μ_j by introducing a parameter δ ($0 \leq \delta \leq 1$) that controls the range of “noise” mixed with increment/decrement in a Lagrangean multiplier. The noise, denoted as N_δ , is a random variable whose value is uniformly distributed over $[-\delta, \delta]$. Multiplier updating rule (4) is thus replaced by

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - (1 + N_\delta)l^{(t)} \frac{g_j}{|S_j|}. \quad (5)$$

This rule diversifies agents’ views on the value of μ_j and thus the protocol can break an infinite loop. On the other hand, since properties 1 and 2 do not hold anymore under this rule, the protocol may converge to a non-optimal (but feasible) solution. Note that rule (5) is equal to rule (4) if δ is set to zero.

4. EXPERIMENTS

Through experiments, we observed the actual performance of the protocol when varying the values of δ and the *assignment topologies*. We adopted two problem suites, one is hand-made but has various assignment topologies and the other is from the GAP benchmark instances of OR-Library but has a specific assignment topology.

Consequently, we observed that the performance of the protocol dramatically changed when δ was set to one of the non-zero values. The results showed that the cost got lower

while the solution quality was kept at a reasonable level, suggesting that by using the protocol with those settings, the agents can quickly agree on a feasible solution with reasonably good quality.

It is also true that the protocol with those settings may fail to find an optimal solution. In the experiments, it failed to find an optimal solution at every non-zero value of δ for 3 complete topology instances of the hand-made problem suite and 5 instances of the benchmark problem suite.

For almost all of the instances, we observed that when δ increased both the cost and the solution quality were reduced. This means that increasing δ may generally influence the agents to rush to reach a compromise of lower quality. This indicates that parameter δ may allow us to control tradeoffs between the quality of a solution and the cost of finding it.

5. CONCLUSION

We presented a new formulation of distributed task assignment and a novel distributed solution protocol, where the agents solve their individual optimization problems and coordinate their locally optimized solutions. Furthermore, we introduce a parameter of the protocol that may allow us to control tradeoffs between the quality of a solution and the cost of finding it.

6. REFERENCES

- [1] D. G. Cattrysse and L. N. V. Wassenhove. A survey of algorithms for the generalized assignment problem. *European J. of Operational Research*, 60:260–272, 1992.
- [2] M. L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.
- [3] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *Intl. J. of Robotics Research*, 23(9):939–954, 2004.
- [4] M. Guignard and S. Kim. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39:215–228, 1987.
- [5] K. Hirayama and M. Yokoo. The distributed breakout algorithms. *Artificial Intelligence*, 161(1–2):89–115, 2005.
- [6] E. Kutanoglu and S. D. Wu. On combinatorial auction and Lagrangian relaxation for distributed resource scheduling. *IEE Transactions*, 31(9):813–826, 1999.
- [7] T. Nishi, M. Konishi, and S. Hasebe. An autonomous decentralized supply chain planning system for multi-stage production processes. *J. of Intelligent Manufacturing*, 16:259–275, 2005.
- [8] I. H. Osman. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *OR Spektrum*, 17:211–225, 1995.
- [9] R. G. Smith. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Trans. on Computers*, 29(2):1104–1113, 1990.
- [10] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Trans. on Knowledge and Data Engineering*, 10(5):673–685, 1998.