An α-approximation 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

This paper presents a new distributed solution protocol, called DisLRP_α, for the *Generalized Mutual Assignment Problem* (GMAP). The GMAP is a typical distributed combinatorial optimization problem whose goal is to maximize *social welfare* of the agents. Unlike the previous protocol for the GMAP, DisLRP_α can provide a theoretical guarantee on global solution quality. In DisLRP_α, as with in the previous protocol, the agents repeatedly solve their local problems while coordinating their local solutions using a distributed constraint satisfaction technique. The key difference is that, in DisLRP_α, each agent is required to produce a feasible solution whose local objective value is not lower than α $(0 < \alpha \leq 1)$ times the local optimal value. Our experimental results on benchmark problem instances show that DisLRP_α can certainly find a solution whose global objective value is higher than that theoretically guaranteed. Furthermore, they also show that, while spending extra communication and computation costs, DisLRP_α can produce a significantly better solution than the previous protocol if we set α appropriately.

Introduction

Cooperative distributed problem solving deals with a multi-agent problem whose goal is shared by all of the agents. If such a shared goal is related to optimization and the problem is inherently discrete, the entire problem may be formulated as a *distributed combinatorial optimization problem*. Although a number of studies have been made on solving the problem (Androulakis & Reklaitis 1999; Kutanoglu & Wu 1999), there has been not so much work on *distributed solution protocol*. By a distributed solution protocol we mean a distributed algorithm, which can work without using any global entity (such as server or coordinator).

Recently, several researchers have proposed distributed solution protocols for the *Distributed Constraint Optimization Problem* (DCOP) (Modi *et al.* 2003). The DCOP provides a general and simple model for distributed combinatorial optimization, but their complete distributed solution protocols are generally not very efficient.

On the other hand, the robotics research community has recently developed market-based techniques for coordinating a team of multiple robots (Dias *et al.* 2006). Some of them can be viewed as (practical) distributed solution protocols for a distributed combinatorial optimization problem, but their solution quality is generally not very high.

As a new model for distributed combinatorial optimization, we have proposed the *Generalized Mutual Assignment Problem* (GMAP) along with distributed solution protocols (Hirayama 2006a; 2006b). The GMAP is a distributed formulation of the *Generalized Assignment Problem* (GAP). The GAP is a centralized NP-hard problem whose goal is to find the most profitable 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 (Nauss 2006; Yagiura & Ibaraki 2006). The GMAP, on the other hand, can be considered a distributed problem, where the agents themselves, each of which has a set of jobs, try to solve the entire GAP while communicating with each other.

For the GMAP, we have presented two distributed solution protocols. One is $\mathrm{DisLRP}_\mathrm{L}$ yielding a solution that provides a lower bound of the global optimal value (Hirayama 2006b) and the other is $\mathrm{DisLRP}_\mathrm{U}$ yielding an upper bound of the global optimal value (Hirayama 2006a). These protocols follow the same underlying procedure, where the agents repeatedly solve their local problems while coordinating their local solutions using a distributed constraint satisfaction technique. Note that, unlike other protocols for similar problems (Androulakis & Reklaitis 1999; Kutanoglu & Wu 1999), these are pure distributed solution protocols. Namely, they do not rely on any global entity (i.e., server or coordinator) that would become a performance bottleneck and reduce robustness of the system.

One drawback of $\mathrm{DisLRP}_\mathrm{L}$ is solution quality. It exploits randomness in solving local problems in order to produce quick agreement between the agents on a global solution. By controlling the degree of randomness one can control the quality of the solution to some extent, but no theoretical guarantee is provided on that quality. This paper presents a new distributed solution protocol, called DisLRP_α, for the GMAP that can provide a theoretical guarantee on global solution quality. DisLRP_α accepts a certain value of α $(0 < \alpha \leq 1)$ as an input and can produce a global solution

whose objective value is not lower than α times the global optimal value. Note that by using DisLRP_α every agent assures herself that an obtained global solution has that property even if any agent does not know what the global solution is and what the global optimal value is.

This paper first provides a formal definition of the GMAP followed by a new property. Then, it presents the new protocol of DisLRP_α, which exploits the proved property. Next, it gives experimental results on benchmark instances showing the actual performance of the protocol and finally concludes this work.

Generalized Mutual Assignment Problem

The goal of the GAP is to find the most profitable assignment of n jobs to m agents such that every job is assigned to exactly one agent and the assignment satisfies all of the resource constraints imposed on individual agents. The GAP is NP-hard; furthermore, the problem of judging the existence of a feasible solution to the GAP is NP-complete. It can be formulated as the following integer programming problem (Nauss 2006; Yagiura & Ibaraki 2006):

$$
\begin{aligned}
\mathcal{GAP} \quad & (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J): \\
\max. \quad & \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} \\
\text{s. t.} \quad & \sum_{k \in A} x_{kj} = 1, \ \ \forall j \in J, && (1) \\
& \sum_{j \in J} w_{kj} x_{kj} \leq c_k, \ \ \forall k \in A, && (2) \\
& x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J, && (3)
\end{aligned}
$$

where $A = \{1, ..., m\}$ is a set of agents; $J = \{1, ..., n\}$ is a set of jobs; p_{kj} and w_{kj} are the profit and amount of resource required, respectively, when agent k selects job j; c_k is the capacity (amount of available resource) of agent k. x_{kj} is a decision variable whose value is set to 1 when agent k selects job j and 0 otherwise.

The *Lagrangian relaxation problem* for the GAP, denoted as $\mathcal{LGAP}(\mu)$, is obtained by dualizing the *assignment constraints* (1) of $\mathcal{GAP}$ (Fisher 1981):

$$
\begin{aligned}
\mathcal{LGAP}(\mu) \quad & (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J): \\
\max. \quad & \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} + \sum_{j \in J} \mu_j \left(1 - \sum_{k \in A} x_{kj} \right) \\
\text{s. t.} \quad & \sum_{j \in J} w_{kj} x_{kj} \leq c_k,\ \forall k \in A, && (4) \\
& x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J, && (5)
\end{aligned}
$$

where μ_j is a real-valued parameter called a *Lagrange multiplier* for job j and the vector $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ is called a *Lagrange multiplier vector*. It is known that for any value of μ the optimal value of $\mathcal{LGAP}(\mu)$ provides an upper bound of the optimal value of $\mathcal{GAP}$.

Since, in $\mathcal{LGAP}(\mu)$, the objective function is additive over the agents and the constraints (4) are separable over the agents, this maximization can be achieved by solving the subproblems, each belongs to agent k (Lasdon 2002):

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

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 that $S_j \neq \emptyset$ (i.e., $|S_j|$ is not equal to zero) because a job with an empty S_j does not have to be considered. $\pi_k(\mu)$ indicates a projection of μ over the jobs in R_k.

The GMAP is assumed to be a distributed problem, where each agent has a set of jobs and tries to assign each of her jobs to some agent such that the total sum of profits (*social welfare*) is maximized. In other words, the agents themselves try to generate an optimal solution of the GAP from a non-optimal (and sometimes infeasible) solution of it. To solve the GMAP, without gathering all information at one place, distributed solution is possible by exploiting the following properties on the relation between the decomposed subproblems and the global problem (Hirayama 2006b).

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

Property 2 *For some value of μ, if all of the optimal solutions to $\{\mathcal{LGMP}_k(\pi_k(\mu)) | k \in A\}$ satisfy the assignment constraints (1) of $\mathcal{GAP}$, then these optimal solutions constitute an optimal solution to $\mathcal{GAP}$.*

This paper presents the following new property resulting a new solution protocol for the GMAP that can provide a theoretical guarantee on global solution quality. It must be noted that this is a generalization of Property 2.

Property 3 *For some value of μ, if all of the feasible solutions to $\{\mathcal{LGMP}_k(\pi_k(\mu)) | k \in A\}$ satisfy the assignment constraints (1) of $\mathcal{GAP}$, and their objective values are not lower than α $(0 < \alpha \leq 1)$ times the respective optimal values, then these feasible solutions constitute a feasible solution to $\mathcal{GAP}$ whose objective value is not lower than α times the optimal value.*

Proof: Let an objective value of a feasible solution to $\mathcal{LGMP}_k(\pi_k(\mu))$ be $v_k(\mu)$, the optimal value of $\mathcal{LGMP}_k(\pi_k(\mu))$ be $opt_k(\mu)$, and the optimal value of $\mathcal{GAP}$ be OPT. Obviously, those feasible solutions constitute a feasible solution to $\mathcal{GAP}$ since they satisfy all constraints of $\mathcal{GAP}$. Furthermore, the objective value of such feasible solution to $\mathcal{GAP}$ is equal to $\sum_{k \in A} v_k(\mu)$. Since $v_k(\mu) \geq \alpha \times opt_k(\mu)$ for any agent k, we obtain

$$
\begin{aligned}
\sum_{k \in A} v_k(\mu) &\geq \sum_{k \in A} \alpha \times opt_k(\mu) \\
&= \alpha \sum_{k \in A} opt_k(\mu) \\
&\geq \alpha \cdot OPT,
\end{aligned}
$$

where we use Property 1 for the second inequality. □

Protocol

We have proposed $\mathrm{DisLRP_L}$ for finding a solution to GMAP (Hirayama 2006b). This section first describes the basic methods of $\mathrm{DisLRP_L}$, which will also be used in the new protocol. Then, it describes the scheme and implementation of the new protocol.

Basic Methods

Property 2 suggests that, to solve $\mathcal{GAP}$, the agents should determine the values of Lagrange multiplier vector so that the optimal solutions to $\{\mathcal{LGMP}_k(\pi_k(\mu))|k \in A\}$ satisfy the assignment constraints of $\mathcal{GAP}$. This may be accomplished by solving a *dual problem* whose goal is to minimize an upper bound of the global optimal value. On the other hand, the problem of solving $\{\mathcal{LGMP}_k(\pi_k(\mu))|k \in A\}$ with a specific Lagrange multiplier vector is called a *primal problem*. An overall behavior of the protocol is that the agents start with $t = 0$ and $\mu^{(0)} = (0, \ldots, 0)$ and alternate solving primal and dual problems while performing local communication with their respective *neighbors*. More specifically, each agent k behaves as follows:

Step 1: sets $t = 0$ and $\mu^{(0)} = (0, \ldots, 0)$,

Step 2: solves $\mathcal{LGMP}_k(\pi_k(\mu^{(t)}))$,

Step 3: exchanges solutions with her neighbors,

Step 4: updates the Lagrange multiplier vector from $\mu^{(t)}$ to $\mu^{(t+1)}$,

Step 5: increases t by one and goes to Step 2.

The loop between Steps 2 and 5 is called a *round* and stops when all of the assignment constraints are satisfied or a prespecified upper bound of rounds is reached. It must be noted that this protocol uses only local communication among agents. It never uses any global entity that can easily interact with the whole agents.

The primal problem, $\mathcal{LGMP}_k(\pi_k(\mu^{(t)}))$ with a specific Lagrange multiplier vector in Step 2, is virtually the *0-1 knapsack problem*, which is known to be NP-hard. The protocol does not provide a specific solver for $\mathcal{LGMP}_k(\pi_k(\mu^{(t)}))$, but we should use a state-of-the-art exact solver since an agent has to solve this problem repeatedly while varying the Lagrange multiplier vector. Fortunately, the 0-1 knapsack problem is said to be a "easier hard" problem (Fisher 1981) and there exist practically efficient solvers.

Neighbors of agent k, mentioned in Step 3, are informally stated as a set of agents with whom agent k shares assignment constraints. They can be formally described as $\bigcup_{j \in R_k} S_j \setminus \{k\}$ (Hirayama 2006b).

To solve the dual problem an agent uses a *subgradient optimization method*. In this method, agent k first computes *subgradient* g_j for each job j in R_k by

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

Obviously, agent k needs to know current solutions of neighbors to compute subgradients. Intuitively, g_j means the gap between the number of agents required for job j (one in this case) and the number of agents that currently select this job. Then, in Step 4, agent k updates μ_j for each job j in R_k by

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - l^{(t)} \frac{g_j}{|S_j|}, \tag{6}$$

where $l^{(t)}$ is a *step length*, a positive scalar parameter that may vary depending on discrete time t, and $|S_j|$ is the number of agents involved in job j. According to this rule, μ_j gets higher when job j attracts too many agents and lower when it attracts too few agents. Namely, μ_j can be viewed as the *price* of job j.

By repeating Steps 2-5 we can expect that an upper bound of the global optimal value of $\mathcal{GAP}$ gradually decreases, and thus the agents eventually reach to a state that is close to a global optimal solution. In the meantime, if all of the assignment constraints are satisfied, the protocol can be terminated because this fact indicates the agents have reached to a global optimal solution. Note that each agent can detect this fact by using the termination detection technique of the distributed breakout algorithm (Hirayama & Yokoo 2005).

Scheme

It has been observed that the protocol converges to a state that is close to a global optimal solution (Hirayama 2006a). However, such a global state is usually infeasible, where the agents violate some assignment constraints. In order to achieve global feasibility, the *noise strategy* has been proposed in (Hirayama 2006b), which allows each agent to use a non-deterministic (i.e. *noisy*) rule instead of the deterministic rule of (6) for updating Lagrange multipliers. In fact, μ_j is distributed over the agents involved in job j, each of which updates its value with a given rule. When the rule is deterministic like (6), the agents involved in job j always keep the same value to their respective μ_js. On the other hand, when the rule is non-deterministic, they tend to assign different values to their respective μ_js. A noisy rule was surprisingly effective for achieving global feasibility at the little cost of solution quality degradation (Hirayama 2006b). However, the previous noisy rule has only a limited range of values on the control parameter, and thus we cannot introduce more drastic noise. Therefore, we first introduce the following new rule for updating Lagrange multipliers:

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - U_\delta \cdot \frac{g_j}{|S_j|}, \tag{7}$$

where U_δ is a *random step length* whose value is uniformly distributed over $[0, \delta)$. With this rule we can introduce more drastic noise in μ_j, while preserving an updating direction given by the sign of g_j.

As with the previous noisy rule, this rule has been successful in finding a global feasible solution and furthermore, by controlling the value of δ, it allows us to control global solution quality to some extent. However, a drawback of this rule (and the previous noisy rule) is that it does not provide any theoretical guarantee on global solution quality. In this paper, we aim at avoiding this drawback and introduce DisLRP_α that can provide a theoretical guarantee on global solution quality.

DisLRP_α accepts a certain value of α $(0 < \alpha \leq 1)$ as an input and can produce a global solution whose objective value is not lower than α times the global optimal value. This protocol relies on Property 3, which says that a sufficient condition for a global solution to have at least the quality of α is: 1) all of the feasible solutions to $\{\mathcal{LGMP}_k(\pi_k(\mu))|k \in A\}$ satisfy the assignment constraints, and 2) the objective values of their feasible solutions are not lower than α times their respective optimal values. This condition can be achieved by the new protocol, where each agent k repeats the following steps until all of the assignment constraints are satisfied:

Step 1: sets $t = 0$ and $\mu^{(0)} = (0, \ldots, 0)$,

Step 2: *finds a feasible solution to* $\mathcal{LGMP}_k(\pi_k(\mu^{(t)}))$ *whose objective value is not lower than* α *times the optimal value,*

Step 3: exchanges solutions with her neighbors,

Step 4: updates the Lagrange multiplier vector from $\mu^{(t)}$ to $\mu^{(t+1)}$,

Step 5: increases t by one and goes to Step 2.

Note that it is different from the basic protocol only in Step 2. While the basic protocol requires an optimal solution in Step 2, the new protocol can relax this requirement.

Implementation

In Step 2 of the new protocol each agent k must find a feasible solution to $\mathcal{LGMP}_k(\pi_k(\mu^{(t)}))$ whose objective value is not lower than α times the optimal value. We use a simple method for finding such feasible solutions to $\mathcal{LGMP}_k(\pi_k(\mu^{(t)}))$. In this method every agent is assumed to have two Lagrange multipliers, μ_j and ν_j, for job j. μ_j is updated with the deterministic rule of (6) and therefore keeps the same value consistently among the agents involved in job j. This means μ_j constitutes μ, a global Lagrange multiplier vector over the jobs. These multipliers define a subproblem for agent k denoted by $\mathcal{LGMP}_k(\{\mu_j|j \in R_k\})$. On the other hand, ν_j is updated with the noisy rule of (7) and therefore may have different values depending on the agents involved in job j. These define another subproblem for agent k denoted by $\mathcal{LGMP}_k(\{\nu_j|j \in R_k\})$. Note that these two subproblems, $\mathcal{LGMP}_k(\{\mu_j|j \in R_k\})$ and $\mathcal{LGMP}_k(\{\nu_j|j \in R_k\})$, have the same feasible region but different coefficients of their objective functions.

Our basic idea is to exploit an optimal solution to the "skewed" subproblem of $\mathcal{LGMP}_k(\{\nu_j|j \in R_k\})$ as a feasible solution to the "true" subproblem of $\mathcal{LGMP}_k(\{\mu_j|j \in R_k\})$. Namely, if an optimal solution to a skewed subproblem has a "true" objective value that is not lower than α times the "true" optimal value, it is certainly a feasible solution whose objective value is not lower than α times the optimal value for the "true" subproblem. Thus, in Step 2 of the new protocol, agent k performs the following:

Step 2.1: finds an optimal solution S^{true} and the optimal value v^{true} for $\mathcal{LGMP}_k(\{\mu_j^{(t)}|j \in R_k\})$,

Step 2.2: finds an optimal solution S^{skew} for $\mathcal{LGMP}_k(\{\nu_j^{(t)}|j \in R_k\})$,

Step 2.3: computes an objective value v^{skew} of S^{skew} by using the objective function of $\mathcal{LGMP}_k(\{\mu_j^{(t)}|j \in R_k\})$,

Step 2.4: if $v^{skew}/v^{true} \geq \alpha$ then adopts S^{skew}; otherwise adopts S^{true} and performs $\nu_j^{(t)} \leftarrow \mu_j^{(t)}$ for $\forall j \in R_k$.

We should point out that an agent solves two different optimization problems in one round (Steps 2.1 and 2.2) and thus the computational load of an agent gets doubled. This may be a drawback of the new protocol, but we assume that each agent is equipped with an efficient solver. Even if a current solver is not efficient enough, we can expect a forthcoming solver will be able to solve the problem efficiently because the centralized optimization technology is still making progress.

It must be also noted that a skewed subproblem does not necessarily provide a feasible solution whose objective value is not lower than α times the optimal value for the "true" subproblem. If a skewed subproblem produces a poor feasible solution whose objective value is lower than α times the optimal value, we consider that this subproblem has been skewed too much by the noisy rule. If this happens, we cancel out the noise by substituting μ_j for ν_j in Step 2.4.

Experiments

We made experiments to compare the performance of $\mathrm{DisLRP}_\mathrm{L}$ and the family of DisLRP_α ($\mathrm{DisLRP}_{0.90}$, $\mathrm{DisLRP}_{0.95}$, and $\mathrm{DisLRP}_{0.99}$) on some GMAP instances, which were derived from GAP benchmark instances. $\mathrm{DisLRP}_\mathrm{L}$ uses rule (7) for updating Lagrange multipliers, while DisLRP_α uses rule (6) for updating μ_j and rule (7) for updating ν_j. Step length $l^{(t)}$ in rule (6) was fixed to 1.

For a GAP instance, we consider the problem, where the agents themselves try to generate an optimal solution from a non-optimal solution. This problem can be considered a GMAP instance, where each agent tries to (re)assign each of her jobs to any of the agents. In this GMAP instance, since any job j may be assigned to any agent, the whole agents are involved in any job j (i.e., $S_j = A$ for $\forall j \in J$). This means that for any agent k $\mathcal{LGMP}_k(\pi_k(\mu))$ forms a 0-1 knapsack instance with all of the jobs in the system. This GMAP instance is called a *complete-topology* instance for which the quality of global solution found by $\mathrm{DisLRP}_\mathrm{L}$ was likely to be poor (Hirayama 2006b).

Experiments were conducted on a discrete event simulator that simulates the behavior of the protocols. In the simulator we used ILOG CPLEX ver 8.1 for each agent to solve a local 0-1 knapsack instance. We made 10 runs for each instance and a run was cut off at 5000 rounds.

Figure 1 shows the quality of global solutions found by each protocol. The quality was computed by dividing an objective value of an obtained global solution by the global optimal value. In these figures, each vertical line represents a range of quality over 10 runs and a symbol on the line represents their mean.

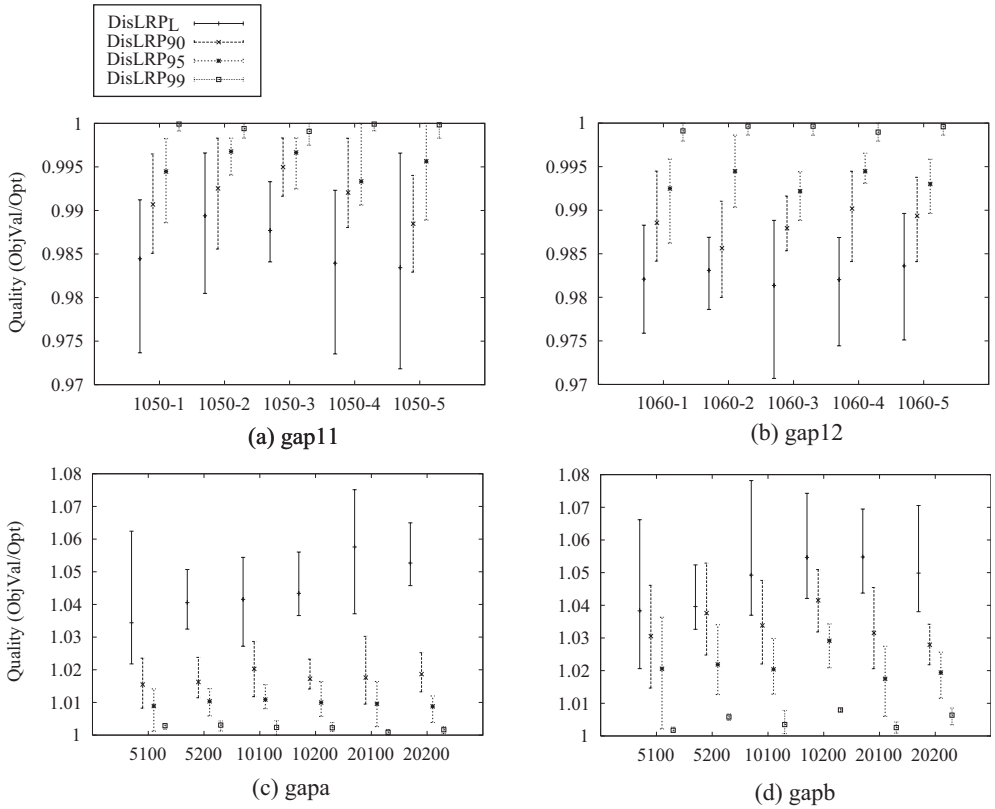

Figure 1: Range of solution quality

The instances of gap11 and gap12 are obtained from http://people.brunel.ac.uk/~mastjjb/jeb/orlib/gapinfo.html. The gap11 includes 5 instances each having 10 agents and 50 jobs, while the gap12 also includes 5 instances each having 10 agents and 60 jobs. On the other hand, the instances of gapa and gapb are obtained from http://www.al.cm.is.nagoya-u.ac.jp/~yagiura/gap/. They include the 6 instances, respectively, of c5100-1 (5 agents and 100 jobs), c5200-2 (5 agents and 200 jobs), c10100-3 (10 agents and 100 jobs), c10200-4 (10 agents and 200 jobs), c20100-5 (20 agents and 100 jobs), and c20200-6 (20 agents and 200 jobs), and are still challenging even for a centralized GAP algorithm. Since the gapa and the gapb have been suggested to be solved as a minimization problem, we applied the protocols, which are designed for maximization, to their instances after transforming them into equivalent maximization instances.

For every instance we can see a clear trend that DisLRP_α can find a global solution with higher quality than $\mathrm{DisLRP}_\mathrm{L}$. We should point out that the quality actually achieved by DisLRP_α is higher than that theoretically guaranteed. For example, $\mathrm{DisLRP}_{0.90}$ guarantees, in theory, that the solution found has an objective value which is not lower than 90 percent of the optimal (for a minimization problem, which is not larger than 10 percent of the optimal), but it actually finds solutions whose objective values are more than 98 percent for the gap11, more than 97 percent for the gap12, less than 4 percent for the gapa, and less than 6 percent for the gapb. Furthermore, we can see that increasing α actually leads to finding global solutions with higher quality and as a result the variance of solution quality is naturally reduced.

Figure 2 shows, for each instance, the average number of rounds at which a global solution was found. For a run that was not finished within 5000 rounds, we counted its cost as 5000. In these figures, the ratio of successful runs is shown at the top of a bar if the ratio is below 1.0.

We can see that DisLRP_α obviously spends much more rounds to find a global solution. Since the number of messages increases with the number of rounds, this indicates DisLRP_α is more expensive in terms of communication cost. Similarly, the number of calling a local solver in an agent also increases with the number of rounds. Besides, an agent in DisLRP_α calls a local solver twice a round. This indicates that a total computation cost for DisLRP_α is much higher than that for $\mathrm{DisLRP}_\mathrm{L}$. However, we consider that these are inevitable costs for finding a global solution with higher quality. One should select a protocol that provides the best balance between solution quality and solution costs depending on one's requirement.

When using rule (7) for updating Lagrange multipliers, we should be careful in selecting a value of δ, which controls the degree of noise, because it is crucial for the performance of the protocol (Hirayama 2006b). In our experiment, δ was set to 3 for the gap11 and the gap12 and 10 for the

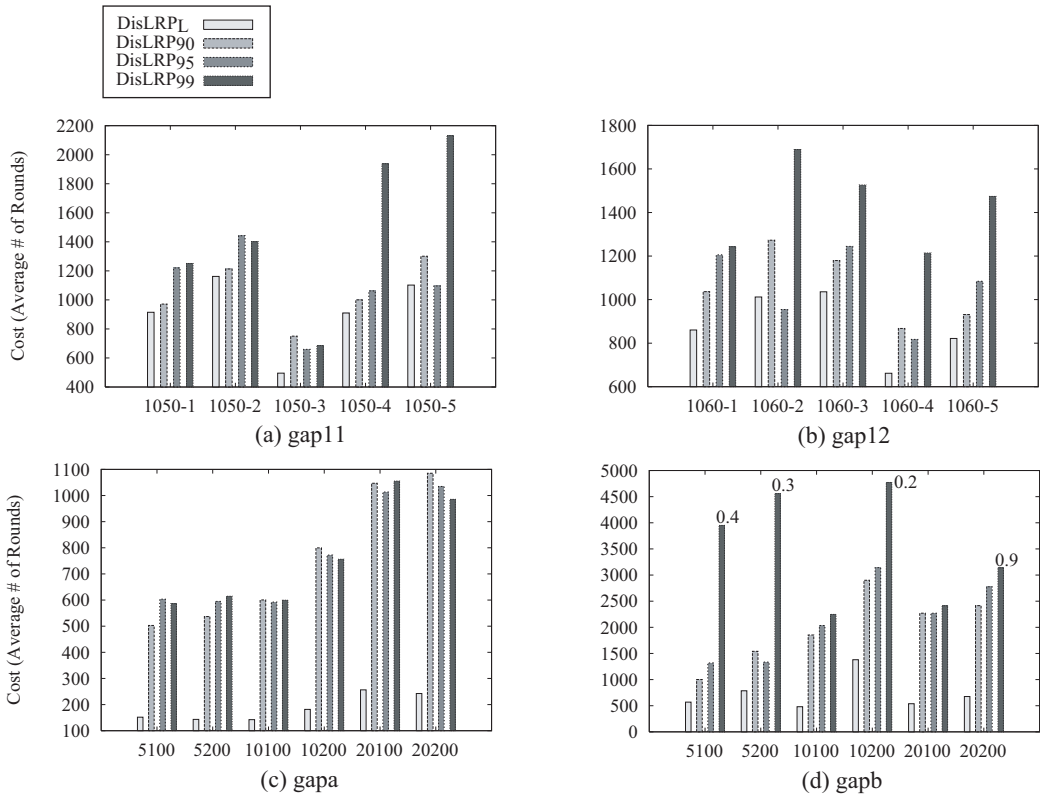

Figure 2: Solution cost

gapa and the gapb. We should point out that when δ takes an extremely large value, DisLRP$_\alpha$, particularly with α be very close to one, may fail to find a solution, because with that setting the noise becomes so large that it is frequently canceled out. In this case DisLRP$_\alpha$ behaves like the plain DisLRP with rule (6). This implies that the performance of DisLRP$_\alpha$ is not very robust for a combination of δ and α.

Conclusions

This paper presented DisLRP$_\alpha$ for the GMAP. Unlike DisLRP$_\mathrm{L}$, DisLRP$_\alpha$ can provide a theoretical guarantee on global solution quality. Our experimental results showed that DisLRP$_\alpha$ can certainly find a solution whose global objective value is higher than that theoretically guaranteed and, while spending extra communication and computation costs, DisLRP$_\alpha$ can produce a significantly better solution than DisLRP$_\mathrm{L}$ if we set α appropriately.

References

Androulakis, I. P., and Reklaitis, G. V. 1999. Approaches to asynchronous decentralized decision making. *Computers and Chemical Engineering* 23:341–355.

Dias, M. B.; Zlot, R.; Kalra, N.; and Stentz, A. 2006. Market-based multirobot coordination: a survey and analysis. *Proceedings of the IEEE* 94(7):1257–1270.

Fisher, M. L. 1981. The Lagrangian relaxation method for solving integer programming problems. *Management Science* 27(1):1–18.

Hirayama, K., and Yokoo, M. 2005. The distributed breakout algorithms. *Artificial Intelligence* 161(1–2):89–115.

Hirayama, K. 2006a. A distributed solution protocol that computes an upper bound for the generalized mutual assignment problem. In *7th Intl. Ws. on Distributed Constraint Reasoning*, 102–116.

Hirayama, K. 2006b. A new approach to distributed task assignment using Lagrangian decomposition and distributed constraint satisfaction. In *AAAI-2006*, 660–665.

Kutanoglu, E., and Wu, S. D. 1999. On combinatorial auction and Lagrangian relaxation for distributed resource scheduling. *IIE Transactions* 31(9):813–826.

Lasdon, L. S. 2002. *Optimization Theory for Large Systems*. Dover.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2003. An asynchronous complete method for distributed constraint optimization. In *AAMAS-2003*, 161–168.

Nauss, R. M. 2006. The generalized assignment problem. In Karlof, J. K., ed., *Integer Programming: Theory and Practice*, 39–55. CRC Press.

Yagiura, M., and Ibaraki, T. 2006. Generalized assignment problem. In Gonzalez, T. F., ed., *Handbook of Approximation Algorithms and Metaheuristics*, Computer & Information Science Series. Chapman & Hall/CRC.