*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*

# Distributed Lagrangian Relaxation Protocol for the Over-constrained Generalized Mutual Assignment Problem

Kenta Hanada and Katsutoshi Hirayama

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

**Abstract.** *The Generalized Mutual Assignment Problem* (GMAP) is a distributed combinatorial optimization problem in which, with no centralized control, multiple agents search for an optimal assignment of goods that satisfies their individual knapsack constraints. Previously, in the GMAP protocol, problem instances were assumed to be feasible, meaning that the total capacities of the agents were large enough to assign the goods. However, this assumption may not be realistic in some situations. In this paper, we present two methods for dealing with such "over-constrained" GMAP instances. First, we introduce a *disposal agent* who has an unlimited capacity and is in charge of the unassigned goods. With this method, we can use any off-the-shelf GMAP protocol since the disposal agent can make the instances feasible. Second, we formulate the GMAP instances as an Integer Programming (IP) problem, in which the assignment constraints are described with inequalities. With this method, we need to devise a new protocol for such a formulation. We experimentally compared these two methods on the variants of *Generalized Assignment Problem* (GAP) benchmark instances. Our results indicate that the first method finds a solution faster for fewer over-constrained instances, and the second finds a better solution faster for more over-constrained instances.

**Keywords:** generalized mutual assignment problem, distributed optimization, Lagrangian relaxation

## 1 Introduction

Obviously, in distributed AI, the distributed assignment, whose task is to assign something in a distributed context, has been a fundamental problem for decades. The *contract net protocol* [1] may be the oldest example that performs this task. More recently, *multi-robot task allocation* [2] and *distributed target tracking* [3] have attracted much attention as applications of this technology. For more formal treatment, *distributed constraint optimization* [4] and *distributed facility location* [5] seem popular as a basis for the reasoning or the optimization of distributed assignments.

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



To formally deal with complex assignment problems, Hirayama et al. proposed the *Generalized Mutual Assignment Problem* (GMAP) and the *Distributed Lagrangian Relaxation Protocols* (DisLRP) [6][7] [8]. GMAP is the distributed version of the *Generalized Assignment Problem* (GAP), whose goal is to find the most profitable assignment of goods to agents. In GMAP, the agents themselves cooperatively search for such an assignment. We regard this problem as a set partitioning problem by agents, each of whom has a resource constraint.

Here is an outline of agent behavior in DisLRP. First, the agents solve their individual 0-1 knapsack problems and announce their assignments of goods to their respective neighbors. Second, for all goods, the agents raise their *price* (Lagrange multiplier) if it is chosen by two or more agents, and they reduce their price if it is not chosen by any agent. Third, under the new prices, the agents solve their individual new 0-1 knapsack problems again. The agents repeat this procedure until all of the goods are chosen by exactly one agent, which means we get a proper set partition for the entire set of goods.

Previously, we assumed that, in GMAP, the total capacity of agents is large enough to assign the goods. However, we can easily imagine "over-constrained" situations, where the agents don't have enough resource capacities for the entire set of goods. We develop two methods to deal with such an over-constrained situation and experimentally compare them using the variants of GAP benchmark instances.

The basic idea of the first method is the introduction of an additional agent without a knapsack constraint, or equivalently, one with infinite capacity. We call this agent the *disposal agent*. The disposal agent assigns the goods that have not been chosen by any agent. By this method, since we do not need to change the existing GMAP formulation, we have an advantage because the off-the-shelf DisLRP can be used without any modifications. On the other hand, a basic idea of the second method is that we first formulate GMAP as an Integer Programming (IP) problem in which the assignment constraints are described with inequalities and relax them to decompose the problem. By this method, we do not need to introduce a special agent like the disposal agent, but we do need to adapt DisLRP to this new formulation.

The remainder of this paper is organized as follows. First, we define GMAP in Section 2. In Section 3, we present our two methods for dealing with over-constrained GMAP instances, each of which consists of the formulation of a problem and a solution. Next, we show the results of experiments on the variants of the GAP benchmark instances in Section 4 and conclude in Section 5.

## 2 Generalized Mutual Assignment Problem

GAP has been studied for many years in operations research. Since it is a NP-hard problem, many exact algorithms [9] and heuristic approaches [10] have been proposed in centralized contexts. GMAP is a distributed version of GAP. In the entire system, agents on GMAP solve the following IP problem, denoted as $\mathcal{GAP}$:

$$\mathcal{GAP} \quad (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J) :$$

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



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

where $A = \{1, ..., m\}$ is a set of agents, $J = \{1, ..., n\}$ is a set of goods, and $p_{kj}$ and $w_{kj}$ are the profit and the amount of resources required when agent $k$ selects goods $j$. $c_k$ is the capacity, i.e., the amount of available resources, of agent $k$. $x_{kj}$ is a decision variable whose value is set to 1 when agent $k$ selects goods $j$ and 0 otherwise. The goal of the problem is to maximize the summation of profits under the assignment constraints (1), which means each good is assigned to exactly one agent and the knapsack constraints (2), which means no agent can use more resources than its capacity.

To solve this problem by using a distributed method, we have to divide the problem while keeping its structure. The *Lagrangian decomposition* [11] provides such decomposition of the problem. The Lagrangian relaxation problem is obtained by dualizing the assignment constraints (1) of $\mathcal{GAP}$ as follows:

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

where $\mu_j$ is a real-valued parameter called a *Lagrange multiplier* for goods $j$ and vector $\mu = (\mu_1, \mu_2, ..., \mu_n)$ is called a *Lagrange multiplier vector*. For any value of $\mu$, $L(\mu)$ provides an upper bound on the optimal value of $\mathcal{GAP}$[12].

Since an upper bound should be the lowest, we have another minimization problem on $\mu$:

$$\min.\ \ L(\mu).$$

We usually call this the *Lagrangian dual problem*. In $L(\mu)$, the objective function is additive over the agents and the constraints are separable over them; this maximization can be achieved by solving the following subproblems: for each agent $k$,

$$L_k(\mu) = \max.\ \sum_{j\in J}(p_{kj} - \mu_j)x_{kj}$$

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



$$\text{s. t.} \sum_{j \in J} w_{kj} x_{kj} \leq c_k,$$
$$x_{kj} \in \{0, 1\},\ \forall j \in J,$$

and for the remaining terms,

$$L_{const}(\mu) = \sum_{j \in J} \mu_j,$$

We can thus describe the Lagrangian dual problem as follows:

$$\text{min.} \sum_{k \in A} L_k(\mu) + L_{const}(\mu),$$

Our distributed solution method solves this problem using only local communications among agents.

## 3 Solutions for the over-constrained problem

Basically, GAP and GMAP are supposed to be feasible, meaning a proper set partition of the goods exists that does not violate the knapsack constraints. However, in reality, we may face over-constrained situations, where the agents do not have enough capacity for the entire set of goods. In this paper, we present two methods for dealing with such over-constrained situations.

### 3.1 DisLRP with a disposal agent

The first method introduces an additional agent, called a disposal agent, who has no knapsack constraint or is equivalently equipped with infinite capacity. The disposal agent does not get any profit even if he has some goods. Among the regular agents and the disposal agent, all goods must be assigned to exactly one agent.

**Formulation** We can formulate GMAP including the disposal agent, denoted by $d \notin A$, as follows:

$$\begin{aligned}
&\mathcal{GAP}' \ \ (\text{decide } x_{kj},\ \forall k \in A \cup \{d\},\ \forall j \in J) : \\
&\text{max.} \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} \\
&\text{s. t.} \sum_{k \in A \cup \{d\}} x_{kj} = 1, \ \ \forall j \in J, \\
&\sum_{j \in J} w_{kj} x_{kj} \leq c_k, \ \ \forall k \in A, \\
&x_{kj} \in \{0, 1\},\ \forall k \in A \cup \{d\},\ \forall j \in J.
\end{aligned}$$

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



For $\mathcal{GAP}'$, the Lagrangian relaxation problem that dualizes the assignment constraints is described as follows:

$$\begin{aligned} L'(\mu) = \max. \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj} \\ + \sum_{j\in J}\mu_j\left(1 - \sum_{k\in A\cup\{d\}} x_{kj}\right) \\ \text{s. t.} \sum_{j\in J} w_{kj}x_{kj} \le c_k,\ \forall k \in A, \\ x_{kj} \in \{0,1\},\ \forall k \in A\cup\{d\},\ \forall j \in J, \end{aligned} \tag{3}$$

For any value of $\mu$, $L'(\mu)$ provides an upper bound on the optimal value of $\mathcal{GAP}'$.

Since an upper bound should be the lowest, we have another minimization problem on $\mu$:

$$\text{min. } L'(\mu).$$

We usually call this the *Lagrangian dual problem*. Since, in $L'(\mu)$, the objective function is additive over the agents and the constraints (3) are separable over the agents, this maximization can be achieved by solving the following subproblems: for each regular agent $k$,

$$\begin{aligned} L'_k(\mu) = \max. \sum_{j\in J}(p_{kj} - \mu_j)x_{kj} \\ \text{s. t.} \sum_{j\in J} w_{kj}x_{kj} \le c_k, \\ x_{kj} \in \{0,1\},\ \forall j \in J, \end{aligned}$$

for disposal agent $d$,

$$\begin{aligned} L'_d(\mu) = \max. \sum_{j\in J}(-\mu_j)x_{dj} \\ \text{s. t. } x_{dj} \in \{0,1\},\ \forall j \in J, \end{aligned}$$

and for the remaining terms,

$$L'_{const}(\mu) = \sum_{j\in J}\mu_j.$$

We can thus describe the Lagrangian dual problem as follows:

$$\text{min. } \sum_{k\in A} L'_k(\mu) + L'_d(\mu) + L'_{const}(\mu).$$

Our distributed solution method solves this problem using only local communications among agents including the disposal agent.

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



**Virtualization of the disposal agent** Since we added the disposal agent in this method, communication costs would increase because the total number of agents increases by one. To avoid this scenario, we can virtualize the disposal agent in our real implementation. Looking at the subproblem of the disposal agent $L'_d(\mu)$, we realize that the disposal agent assigns goods $j$ if $\mu_j$ becomes lower than zero. Therefore, the regular agents should know what goods are assigned to the disposal agent by using Lagrange multiplier $\mu$. Consequently, we need not to add the *real* disposal agent. Instead, the regular agents can simulate the behavior of the disposal agent. We will describe this method more concretely below.

**Solution** The basic procedure of this protocol is as follows:

**(Step 1)** All agents initialize their Lagrange multiplier vectors as 0.

**(Step 2)** Under a current value of $\mu$, each agent $k$ solves his knapsack problem to compute $L'_k(\mu)$. Then the agents send those results to their respective neighbors. At the same time, each agent treats every good $j$ whose $\mu_j$ is lower than zero as if it is assigned to the disposal agent.

**(Step 3)** If all assignment constraints of the original problem are satisfied, the agents can quit the procedure to provide an optimal solution.

**(Step 4)** Each agent $k$ finds an upper bound and a lower bound. Agent $k$ also finds the smallest upper bound, *BestUB*, and the largest lower bound, *BestLB*, among those found so far. If both *BestUB* and *BestLB* have the same value, the agents can quit the procedure to provide an optimal solution.

**(Step 5)** Each agent $k$ updates the Lagrange multiplier vector from $\mu^{(t)}$ to $\mu^{(t+1)}$ by the subgradient method [12] and returns to Step 2.

After initialization at Step 1, this procedure iterates Steps 2 through 5. We refer to one iteration as a *round*.

Note that the global information of the entire system is required for computing in Steps 3, 4, and 5. In this work, we use a *spanning tree* to collect this global information, as proposed in [8].

In Step 4, we need to compute both an upper and a lower bound. The upper bound can be computed, at each round, as the total sum of the optimal values of agents (including the disposal agent) plus the total sum of the elements of $\mu$. On the other hand, the lower bound can be computed, at each round, by building a feasible solution for $\mathcal{GAP}'$ out of a solution for $L'(\mu)$. More specifically, a feasible solution is built as follows:

- If a good is assigned to exactly one agent, it will be assigned to the agent.
- If a good is assigned to two or more agents, it will be assigned to the agent among those agents having the largest profit.
- If a good is assigned to no agent, it will be assigned to the disposal agent.

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



In Step 5, we use the subgradient method to update Lagrange multiplier $\mu_j$. In this method, agent $k$ computes subgradient $g_j^{(t)}$ for all goods $j$ by

$$g_j^{(t)} \leftarrow 1 - \sum_{i \in A \cup \{d\}} x_{ij}$$

and updates Lagrange multiplier $\mu_j$ as follows:

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - \frac{\pi^{(t)}(BestUB^{(t)} - BestLB^{(t)})g_j^{(t)}}{\sum_{j \in J}(g_j^{(t)})^2}.$$

In this rule, agent $k$ needs to know *BestUB* and *BestLB* at this point, but they are obviously global information. As we mentioned before, we use the same *spanning tree* as [8] to collect such global information. $\pi$ is a control parameter, whose initial value is two, that halves itself if neither *BestUB* nor *BestLB* is updated through 30 consecutive rounds.

### 3.2 DisLRP with Inequality-based Formulation

The second method describes assignment constraints with inequalities instead of equalities.

**Formulation** We can formulate the entire problem as the following IP problem:

$$\begin{aligned}
\mathcal{GAP}'' \quad & (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J) : \\
\text{max.} \quad & \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} \\
\text{s. t.} \quad & \sum_{k \in A} x_{kj} \leq 1, \quad \forall j \in J, \\
& \sum_{j \in J} w_{kj} x_{kj} \leq c_k, \quad \forall k \in A, \\
& x_{kj} \in \{0, 1\},\ \forall k \in A,\ \forall j \in J.
\end{aligned} \tag{4}$$

The difference between $\mathcal{GAP}$ and $\mathcal{GAP}''$ is that assignment constraints are described with equalities in $\mathcal{GAP}$ and inequalities in $\mathcal{GAP}''$. Clearly, with this "relaxation," we allow all goods to be assigned to no more than one agent.

The Lagrangian relaxation problem is obtained by dualizing the assignment constraints (4) of $\mathcal{GAP}''$ as follows:

$$\begin{aligned}
L''(\mu) = \text{max.} & \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} \\
& + \sum_{j \in J} \mu_j \left( 1 - \sum_{k \in A} x_{kj} \right)
\end{aligned}$$

Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186



$$\begin{aligned} &\text{s. t. } \sum_{j \in J} w_{kj} x_{kj} \leq c_k, \ \forall k \in A, \\ &x_{kj} \in \{0,1\}, \ \forall k \in A, \ \forall j \in J, \\ &\mu_j \geq 0, \ \forall j \in J. \end{aligned}$$

Note that in the above Lagrange multiplier vector $\mu_j$ has a nonnegative constraint, since the assignment constraint on all goods $j$ is described with inequality. Similar to the previous section, the Lagrangian dual problem is the following minimization problem on non-negative real vector space and gives an upper bound on the optimal value of $\mathcal{GAP}''$,

$$\text{min. } L''(\mu) \qquad \text{s. t. } \quad \mu_j \geq 0, \ \forall j \in J.$$

In $L''(\mu)$, the objective function is additive over the agents and the constraints are separable over them; this maximization can be achieved by solving the following subproblems: for each agent $k$,

$$\begin{aligned} L''_k(\mu) = \text{max. } &\sum_{j \in J} (p_{kj} - \mu_j) x_{kj} \\ \text{s. t. } &\sum_{j \in J} w_{kj} x_{kj} \leq c_k, \\ &x_{kj} \in \{0,1\}, \ \forall j \in J, \end{aligned}$$

and for the remaining terms,

$$L''_{const}(\mu) = \sum_{j \in J} \mu_j.$$

We can thus describe the Lagrangian dual problem as follows:

$$\begin{aligned} \text{min. } &\sum_{k \in A} L''_k(\mu) + L''_{const}(\mu) \\ \text{s. t. } &\mu_j \geq 0, \ \forall j \in J. \end{aligned}$$

Our distributed solution method solves this problem by using only local communications among agents.

**Solution** The basic procedure of this protocol is as follows:

**(Step 1)** All agents initialize their Lagrange multiplier vectors as 0.

**(Step 2)** Under a current value of $\mu$, each agent $k$ solves his knapsack problem to compute $L''_k(\mu)$. Then the agents send these results to their respective neighbors.

**(Step 3)** If all assignment constraints of the original problem are satisfied, and on any good $j$,

$$\mu_j (1 - \sum_{k \in A} x_{kj}) = 0 \tag{5}$$

holds, then the agents can quit the procedure to provide an optimal solution.

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



**(Step 4)** Each agent $k$ finds an upper bound and a lower bound. Agent $k$ also finds the smallest upper bound, *BestUB*, and the largest lower bound, *BestLB*, among those found so far. If both *BestUB* and *BestLB* have the same value, the agents can quit the procedure to provide an optimal solution.

**(Step 5)** Each agent $k$ updates the Lagrange multiplier vector from $\mu^{(t)}$ to $\mu^{(t+1)}$ by the subgradient method [12] while keeping $\mu_j \geq 0$ and goes back to Step 2.

In this procedure, note that the following differences are derived from relaxing inequalities instead of equalities.

First, in Step 3, the termination condition of this protocol becomes more difficult than that of the previous one. Second, in Step 4, to build a feasible solution, we can simply ignore the goods that have not been chosen by any agent. Third, in Step 5, we need to replace the updating rule for a Lagrange multiplier by the following rule:

$$g_j^{(t)} \leftarrow 1 - \sum_{i \in A} x_{ij}$$

$$temp \leftarrow \mu_j^{(t)} - \frac{\pi^{(t)}(BestUB^{(t)} - BestLB^{(t)})g_j^{(t)}}{\sum_{j \in J}(g_j^{(t)})^2},$$

$$\mu_j^{(t+1)} \leftarrow \max\{temp, 0\}.$$

Clearly, by this rule, a Lagrange multiplier never gets lower than zero.

Obviously, both $\mathcal{GAP}'$ and $\mathcal{GAP}''$ provide the same optimal value for the over- constrained GMAP because $\mathcal{GAP}''$ turns into $\mathcal{GAP}'$ by introducing slack variables for the inequality constraints. It must be pointed out that the difference between $\mathcal{GAP}'$ and $\mathcal{GAP}''$ may be slight, but their Lagrangian duals, which our methods try to solve, differs significantly in that

- The Lagrangian dual of $\mathcal{GAP}'$ has the problem of the disposal agent while that of $\mathcal{GAP}''$ does not.
- The Lagrangian dual of $\mathcal{GAP}''$ has non-negativity constraints on variables while that of $\mathcal{GAP}'$ does not.

Furthermore, as mentioned before, the method for solving the Lagrangian dual of $\mathcal{GAP}''$ has a more complex termination condition.

## 4 Experiments

We experimentally compared the performance of the two methods on the variants of GAP benchmark instances from the OR-Library[13]. Clearly, the instances in our experiments should be over-constrained, but none of the GAP benchmark instances have that property; in other words, in each benchmark instance, the total capacities of the agents are large enough for the goods to be assigned.

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



**Table 1.** DisLRP with disposal agent on c1040-1

| $x$ | *Round* | *BestUB* | *BestLB* | *Quality* |
|---|---|---|---|---|
| 0.1 | 1 | 0.0000 | 0 | - |
| 0.2 | 190 | 244.0000 | 244 | 1.0000 |
| 0.3 | 2115 | 456.0000 | 455 | 1.0000 |
| 0.4 | 1002 | 601.0000 | 601 | 1.0000 |
| 0.5 | 10000 | 705.7143 | 692 | 0.9806 |
| 0.6 | 819 | 828.0000 | 828 | 1.0000 |
| 0.7 | 10000 | 900.4581 | 887 | 0.9851 |
| 0.8 | 10000 | 934.9447 | 933 | 0.9979 |
| 0.9 | 10000 | 950.0000 | 948 | 0.9979 |

**Table 2.** DisLRP with inequality-based formulation on c1040-1

| $x$ | *Round* | *BestUB* | *BestLB* | *Quality* |
|---|---|---|---|---|
| 0.1 | 1 | 0.0000 | 0 | - |
| 0.2 | 2 | 244.0000 | 244 | 1.0000 |
| 0.3 | 1189 | 455.0000 | 455 | 1.0000 |
| 0.4 | 1264 | 601.0000 | 601 | 1.0000 |
| 0.5 | 10000 | 705.7146 | 690 | 0.9777 |
| 0.6 | 491 | 828.0000 | 828 | 1.0000 |
| 0.7 | 10000 | 900.4357 | 887 | 0.9851 |
| 0.8 | 10000 | 935.0000 | 933 | 0.9979 |
| 0.9 | 10000 | 950.0000 | 931 | 0.9800 |

Thus, to make an instance over-constrained, we simply reduce the capacity of each agent by a constant factor. More specifically, we multiplied capacity $c_k$ of each agent $k$ by $x \in \{0.1, 0.2, \ldots, 0.9\}$. Consequently, we generated 540 instances in total, most of which can be expected to be over-constrained. We call this $x$ a *capacity coefficient*.

We conducted our experiments on a simulator written in JAVA and used lp_solve 5.5.2.0[14] for each agent to solve a local knapsack problem. Lp_solve is a freely available Linear/Integer programming solver with many easy-to-use application program interfaces (APIs).

Tables 1 and 2 show the results for the DisLRP with the disposal agent and the DisLRP with the inequality-based formulation, respectively, on a benchmark instance called c1040-1 that consists of 40 goods to be assigned to 10 agents. Tables 3 and 4 also show the results on a benchmark instance called c1060-1 that consists of 60 goods to be assigned to 10 agents. In these tables, $x$ is the capacity coefficient, *Round* is the number of rounds spent until a procedure is terminated, *BestUB* is the lowest upper bound, and *BestLB* is the highest lower bound. Also, *Quality*, which means the ratio of *BestLB* on *BestUB*, denotes the quality lower bound of the obtained feasible solutions. Obviously, the closer this quality lower bound is to one, the better the performance. In our experiments we stopped a run at 10000 rounds if the procedure failed to find an optimal solution. In that case, *BestUB* and *BestLB* do not reach the same value.

To see whether the differences on *Round* and *Quality* are statistically significant, we applied the Wilcoxon signed-rank test to series of data obtained by our two methods. The results are summarized in Table 5. The null hypothesis on *Quality* is not rejected, which means that our two methods may not be different in terms of *Quality*. On the other hand, the null hypothesis on *Round* is rejected, which suggests that our two methods are different in terms of *Round*.

We further analyzed by dividing the instances into two groups: one was comprised of instances whose capacity coefficients ranged from 0.2 to 0.5 and the other was comprised of instances whose capacity coefficients ranged from 0.6 to

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



**Table 3.** DisLRP with disposal agent on c1060-1

| $x$ | *Round* | *BestUB* | *BestLB* | *Quality* |
|---|---|---|---|---|
| 0.1 | 1 | 239.0000 | 239 | 1.0000 |
| 0.2 | 23 | 474.0000 | 474 | 1.0000 |
| 0.3 | 301 | 784.0000 | 784 | 1.0000 |
| 0.4 | 181 | 1010.0000 | 1010 | 1.0000 |
| 0.5 | 10000 | 1167.4033 | 1159 | 0.9928 |
| 0.6 | 10000 | 1316.7786 | 1295 | 0.9835 |
| 0.7 | 10000 | 1397.3609 | 1373 | 0.9826 |
| 0.8 | 10000 | 1426.0116 | 1406 | 0.9860 |
| 0.9 | 10000 | 1442.0958 | 1419 | 0.9840 |

**Table 4.** DisLRP with inequality-based formulation on 1060-1

| $x$ | *Round* | *BestUB* | *BestLB* | *Quality* |
|---|---|---|---|---|
| 0.1 | 1 | 239.0000 | 239 | 1.0000 |
| 0.2 | 13 | 474.0000 | 474 | 1.0000 |
| 0.3 | 53 | 784.0000 | 784 | 1.0000 |
| 0.4 | 198 | 1010.0000 | 1010 | 1.0000 |
| 0.5 | 10000 | 1167.4036 | 1159 | 0.9928 |
| 0.6 | 10000 | 1316.7501 | 1310 | 0.9949 |
| 0.7 | 10000 | 1397.3601 | 1378 | 0.9861 |
| 0.8 | 10000 | 1426.0126 | 1406 | 0.9860 |
| 0.9 | 10000 | 1442.0978 | 1419 | 0.9840 |

0.9[1]. For each of these two groups, we also applied the Wilcoxon signed-rank test again to see whether the differences on *Round* and *Quality* are statistically significant. The results are summarized in Tables 6 and 7. According to Table 6, both null hypotheses on *Round* and *Quality* are rejected, which means that our two methods have different performances. Looking at their medians, DisLRP with inequality-based formulation seems to find a better solution faster for more over-constrained instances. On the other hand, according to Table 7, only the null hypothesis on *Round* is rejected. Looking at its median, DisLRP with the disposal agent can find a solution faster for fewer over-constrained instances.

To summarize, we found

1. DisLRP with inequality-based formulation finds a better solution faster for more over-constrained instances,
2. DisLRP with the disposal agent finds a solution faster for fewer over-constrained instances.

We are a bit surprised by the first finding in our experiments. For more over-constrained instances, we sometimes observed that the agents in the DisLRP

**Table 5.** Wilcoxon signed-rank test for all instances

| | *Quality* | *Round* |
|---|---|---|
| Hypothesis | No difference between two data series. | |
| Test statistic $T$ | 67704.5 | 15658.5 |
| $\|Z\|$-value | 0.3087 | 3.2182 |
| Conclusion | Not rejected at significance level 5% | Rejected at significance level 1% |
| Median | | |
| (disposal) | 0.9998 | 285 |
| (inequality) | 0.9990 | 174 |

[1] We ignored the instances whose capacity coefficients are 0.1 since most have zero as their optimal values.

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



**Table 6.** Wilcoxon signed-rank test for instances whose capacity coefficients range from 0.2 to 0.5

| | *Quality* | *Round* |
|---|---|---|
| Hypothesis | No difference between two data series. | |
| Test statistic $T$ | 250 | 3122 |
| $\|Z\|$-value | 3.17479 | 7.3127 |
| Conclusion | Rejected at significance level 1% | Rejected at significance level 1% |
| Median | | |
| (disposal) | 0.9980 | 171.5 |
| (inequality) | 1.0000 | 64 |

**Table 7.** Wilcoxon signed-rank test for instances whose capacity coefficients range from 0.6 to 0.9

| | *Quality* | *Round* |
|---|---|---|
| Hypothesis | No difference between two data series. | |
| Test statistic $T$ | 2937 | 980 |
| $\|Z\|$-value | 1.9490 | 2.6479 |
| Conclusion | Not rejected at significance level 5% | Rejected at significance level 1% |
| Median | | |
| (disposal) | 0.9859 | 455 |
| (inequality) | 0.9831 | 1220 |

with inequality-based formulation are likely to select the sets of goods that are mutually exclusive in the very early rounds and, as a result, seem to have many chances to satisfy the termination condition (5). This suggests that such instances may be solved (nearly) optimally even with local knapsack optimization. On the other hand, in the DisLRP with the disposal agent, who has poor knowledge about the system, especially in the very early rounds, the states of the system may be disturbed in those early rounds. We expect this explains our first finding. For less over-constrained instances, both methods require much effort to coordinate the selection of goods among the agents. The DisLRP with inequality-based formulation requires more rounds to coordinate since its Lagrangian dual problem is more restrictive.

## 5 Conclusion

We presented two methods for over-constrained GMAP instances. The first is DisLRP with a disposal agent, which performs better for fewer over-constrained instances. The second is DisLRP with inequality-based formulation, which performs better for more over-constrained instances.

It is important to keep in mind that these two methods are for over-constrained GMAP instances, where the total capacities of agents are not sufficient for the goods. On the other hand, for an under-constrained instance with sufficient overall capacities, these methods may yield the optimal value that is different

*Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174-186*



from the one obtained by the conventional DisLRP with equality-based formulation. This is obvious because, by relaxing assignment constraints for an under-constrained instance, a feasible region gets larger and as a result the optimal value may change. Currently, it seems reasonable to suggest that the proposed methods be used for over-constrained instances while the previous DisLRP with equality-based formulation be used for under-constrained instances. In our future work, we would like to develop an unified framework for handling both under- and over-constrained GMAP instances.

## References

1. Smith, R.G.: The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Transactions on Computers, 29(2):1104–1113, 1990.
2. Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: a survey and analysis. Proceedings of the IEEE, 94(7):1257–1270, 2006.
3. Bhatti, S., Xu, J.: Survey of target tracking protocols using wireless sensor network. In: Proceedings of the 5th International Conference on Wireless and Mobile Communications, pages 110–115, 2009.
4. Modi, P.J., Shen, W-M., Tambe, M., Yokoo, M.: An asynchronous complete method for distributed constraint optimization. In: Proceedings of the Second International Joint Conference on Autonomous Agents Multi-Agent Systems (AAMAS-2003), pages 161–168, 2003.
5. Frank, C., Römer, K.: Distributed facility location algorithms for flexible configuration of wireless sensor networks. In: Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS-2007), pages 124–141, 2007.
6. Hirayama, K.: A new approach to distributed task assignment using Lagrangian decomposition and distributed constraint satisfaction. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-2006), pages 660–665, 2006.
7. Hirayama, K.: An $\alpha$-approximation protocol for the generalized mutual assignment problem. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI-2007), pages 744–749, 2007.
8. Hirayama, K., Matsui, T., Yokoo, M.: Adaptive price update in distributed Lagrangian relaxation protocol. In: Proceedings of the 8th International Joint Conference on Autonomous Agents Multi-Agent Systems (AAMAS-2009), pages 1033–1040, 2009.
9. Nauss, R.M.: The generalized assignment problem. In: J. K. Karlof, editor, Integer Programming: Theory and Practice, pages 39–55. CRC Press, 2006.
10. Yagiura, M., Ibaraki, T.: Generalized assignment problem. In: T. F. Gonzalez, editor, Handbook of Approximation Algorithms and Metaheuristics, Computer Information Science Series. Chapman Hall/CRC, 2006.
11. Guignard, M., Kim, S.: Lagrangean decomposition: A model yielding stronger Lagrangean bounds. Mathematical Programming, 39:215-228, 1987.
12. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, second edition, 1999.
13. OR-Library, `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`
14. lpsolve, `http://sourceforge.net/projects/lpsolve/`