2004－ICS－138　(13)
2004／12／4

Distributed Lagrangean Relaxation Protocol for the Generalized Mutual Assignment Problem

Katsutoshi Hirayama†

The generalized assignment problem (GAP) is a typical NP-hard problem and has been studied for many years mainly in the operations research community. The goal of the GAP is to find an optimal assignment of jobs to agents such that the assignment satisfies all of the resource constraints imposed on individual agents. This work provides a distributed formulation of the GAP, the generalized mutual assignment problem (GMAP), to deal with the problems of task/job assignment in multi-agent systems. Then, we present a distributed lagrangean relaxation protocol that enables the agents to simultaneously solve a GMAP instance without any global control mechanism nor globally accessible communication medium like shared memory. In the protocol we introduce a parameter that controls the range of "noise" mixed in with the increment/decrement in a lagrangean multiplier. This parameter can be used to make the agents quickly agree on a feasible solution with reasonably good quality. Our experimental results imply that the parameter may also allow you to control a tradeoff between the quality of a solution and the cost of finding it.

1. Introduction

The *generalized assignment problem* (GAP) is a typical combinatorial optimization problem and has been studied for many years mainly in the operations research community. 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. Since the GAP is NP-hard, numerous attempts have been made to develop not only exact methods for finding an optimal solution but also heuristic methods for finding a near optimal solution[4].

This work provides a distributed formulation of the GAP, the *generalized mutual assignment problem* (GMAP), to deal with the problems of task/job assignment in multi-agent systems (MAS). We can view the GAP as a problem where a *coordinator* tries to optimally assigned his jobs to a set of agents. On the other hand, the GMAP can be considered to be a problem where a set of coordinators/agents, each of which having an individual set of jobs, try to optimally assign their jobs to each other.

The GMAP could be solved by a *centralized method* if all of the agents agreed to pass on their jobs and related information to the *super-coordinator* who is responsible for a solution of the entire problem. However, the centralized method, even though being efficient in many cases, is generally considered inappropriate for MAS problems, because it ends up forcing the agents to reveal their private information and gives the administrator at your site extra labor of building and maintaining the super-coordinator. Thus we seek for distributed solution to the GMAP in this work. We present a *distributed lagrangean relaxation protocol* that enables the agents to solve a GMAP instance without any global control mechanism nor globally accessible communication medium like shared memory.

In the literature of distributed problem solving, distributed task assignment protocols have been widely studied and many of them have their root in the well-known contract net protocol[6]. To the author's knowledge, no studies have ever tried to develop such a protocol under the distributed formulation of GAP. Furthermore, only few attempts have so far been made at the decentralized solution of combinatorial optimization problems, with notable exception of the work by Androulakis and Reklaitis on the decentralized solution of the optimization problem[1].

The paper is organized as follows. First, in Section 2 we formalize the GMAP as a collection of integer programs sharing some decision variables with each other and introduce a lagrangean relaxation obtained by dualizing

† Faculty of Maritime Sciences, Kobe University

the assignment constraints. Then, in Section 3 we provide key ideas of the distributed lagrangean relaxation protocol. We also introduce a parameter in order to make the agents quickly agree on a feasible solution with reasonably good quality. Next, in Section 4 we report our experiments being conducted to see the actual performance of the protocol, and finally conclude this work in Section 5.

2. Formalization

2.1 Generalized Mutual Assignment Problem

The GAP can be formulated as the following integer program

$$\text{GAP}: \quad \max. \sum_{i\in A}\sum_{j\in J} p_{ij}x_{ij} \tag{1}$$

$$\text{s. t.} \sum_{i\in A} x_{ij} = 1, \quad \forall j \in J, \tag{2}$$

$$\sum_{j\in J} w_{ij}x_{ij} \le c_i, \quad \forall i \in A, \tag{3}$$

$$x_{ij} \in \{0,1\}, \forall i \in A, \forall j \in J, \tag{4}$$

where $A = \{1,\ldots,m\}$ is a set of agents, $J = \{1,\ldots,n\}$ is a set of jobs, p_{ij} is the profit of assigning job j to agent i, w_{ij} is the resource requirement for agent i to perform job j, and c_i is the available resource capacity of agent i. Decision variable x_{ij} takes 1 if agent i is to perform job j and 0 otherwise. The objective is to maximize the profit of the assignment such that each job is assigned to exactly one agent (the *assignment constraints* (2) are satisfied), the total resource requirement of each agent does not exceed the available resource capacity of it (the *knapsack constraints* (3) are satisfied), and each job is assigned or not assigned to an agent (the *01 constraints* (4) are satisfied). The maximal value of the profit is called the *optimal value* and an assignment that gives the optimal value is called an *optimal solution*. The problem of finding an optimal solution to the GAP is known to be NP-hard.

The GMAP consists of a set of agents, each of which (say agent $k \in \{1,\ldots,m\}$) has its own GAP including a set of agents A_k and a set of jobs J_k. Obviously, agent k wants all of the jobs in J_k to be assigned to the agents in A_k. If we view $\bigcup_{k=1}^m A_k$ as A and $\bigcup_{k=1}^m J_k$ as J, we can easily formulate this problem as the GAP.

However, since we seek for distributed solution to the GMAP, this formulation must be decomposed into pieces, each of which is uniquely mapped to an agent. We can obtain one natural decomposition based on the idea that the value of decision variable x_{kj} should be determined by agent k ($\in A$) (in other words agent k has the right to decide whether it will undertake job j or not), where agent k ($\in A$) has

$$\begin{aligned}\text{GMP}_k: \quad \max. & \sum_{j\in R_k} p_{kj}x_{kj} \\ \text{s. t.} & \sum_{i\in S_j} x_{ij} = 1, \quad \forall j \in R_k, \\ & \sum_{j\in R_k} w_{kj}x_{kj} \le c_k, \\ & x_{kj} \in \{0,1\}, \quad \forall j \in R_k,\end{aligned}$$

and tries to determine the values of x_{kj} ($j \in R_k$). In this formulation, R_k is a set of all jobs that may be assigned to agent k and S_j is a set of all agents that job j may be assigned to. The other elements are same as the ones in the GAP formulation. Note that, in the assignment constraints, GMP_k includes the decision variables whose values are to be determined by other agents. Thus, agent k is unable to solve GMP_k without referring to the values of these variables.

Lemma 1 All of the optimal solutions to $\{GMP_k \mid k \in A\}$ constitute an optimal solution to the GAP.

Proof: This obviously follows from the fact that the feasible region of the GAP is equal to the intersection of all of the feasible regions of $\{\text{GMP}_k \mid k \in A\}$ and the fact that the objective function of the GAP is equal to the sum of the objective functions of $\{\text{GMP}_k \mid k \in A\}$. □

2.2 Lagrangean Relaxation

By dualizing the assignment constraints (2) of the GAP, we obtain the following lagrangean relaxation problem:

$$\begin{aligned}\text{LGAP}(\mu): \quad \max. & \sum_{i\in A}\sum_{j\in J} p_{ij}x_{ij} \\ & + \sum_{j\in J} \mu_j(1 - \sum_{i\in A} x_{ij}) \\ \text{s. t.} & \sum_{j\in J} w_{ij}x_{ij} \le c_i, \forall i \in A, \\ & x_{ij} \in \{0,1\}, \forall i \in A, \forall j \in J,\end{aligned}$$

where $\mu = (\mu_1, ..., \mu_n)$ is the *lagrangean multiplier vector* whose elements (called *lagrangean multipliers*) take real numbers. It is well known

that the optimal value of the LGAP(μ) provides an upper bound for the optimal value of the GAP.

As with the decomposition of the GAP into $\{\mathrm{GMP}_k \mid k \in A\}$, the LGAP($\mu$) can be decomposed into a set of problems, where each agent k ($\in A$) has

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

Lemma 2 All of the optimal solutions to $\{LGMP_k(\mu) \mid k \in A\}$ constitute an optimal solution to the $LGAP(\mu)$.

Proof: This also follows from the fact that the feasible region of the LGAP(μ) is equal to the intersection of all of the feasible regions of $\{\mathrm{LGMP}_k(\mu) \mid k \in A\}$ and the fact that the objective function of the LGAP(μ) is equal to the sum of the objective functions of $\{\mathrm{LGMP}_k(\mu) \mid k \in A\}$. □

Theorem 1 The sum of the optimal values of $\{\mathrm{LGMP}_k(\mu) \mid k \in A\}$ provides an upper bound for the optimal value of the GAP.

Proof: This follows from Lemma 2 and the fact that the optimal value of the LGAP(μ) provides an upper bound for the optimal value of the GAP. □

Note that, in the above, different upper bounds result from different values for μ. An upper bound should be lower (closer to the optimal) and hopefully being minimized by setting appropriate values for μ. This minimization problem, whose goal is to minimize an upper bound for the optimal value of the GAP, is called the *lagrangean dual problem*.

Theorem 2 If all of the optimal solutions to $\{LGMP_k(\mu) \mid k \in A\}$ satisfy the assignment constraints, $\sum_{i \in A} x_{ij} = 1$, $\forall j \in J$, for some values of μ, then these optimal solutions constitute an optimal solution to the GAP.

Proof: Theorem 1 proclaims that these optimal solutions constitute a solution that provides an upper bound for the optimal value of the GAP. On the other hand, since these optimal solutions satisfy the assignment constraints, each of which is an equality constraint relaxed in $\mathrm{LGMP}_k(\mu)$ for any agent k, they also constitute a feasible solution that provides a lower bound for the optimal value of the GAP. Hence, these optimal solutions constitute an optimal solution to the GAP. □

3. Protocol

3.1 Neighborhood

To solve the primal problem, agent k needs to be informed of the values for the variables in the second term of the objective function of $\mathrm{LGMP}_k(\mu)$: for each job j in R_k (a set of all jobs that may be assigned to k), (a) lagrangean multiplier μ_j, (b) the size of set S_j (a set of all agents that j may be assigned to), and (c) decision variables $\{x_{ij} \mid i \in S_j, i \neq k\}$. The values of these variables can be obtained through communication with a set of agents denoted as $\bigcup_{j \in R_k} S_j$. On the other hand, agent k also needs to communicate with the same set of agents in order to solve the dual problem. We therefore refer to the set as agent k's *neighbors* and make an agent communicate only with its neighbors in the protocol.

3.2 Primal Problem

When knowing the values of the associated lagrangean multipliers, the size of S_j for any of its jobs, and the decision variables of its neighboring agents, agent k is to search for an optimal solution to $\mathrm{LGMP}_k(\mu)$ to determine the values of $\{x_{kj} \mid j \in R_k\}$. This search problem is equivalent to the *knapsack problem* whose goal is to pick up the most profitable subset of R_k such that the total resource requirement of the subset does not exceed c_k. Note that, for each job j in R_k, the profit is $p_{kj} - \frac{\mu_j}{|S_j|}$ and the resource requirement is w_{kj}. Since the knapsack problem belongs to the NP-hard problems, an efficient exact solution method virtually does not exist. However, the recent progress in exact solution methods for the knapsack problem is so remarkable that the problem instances with around 10,000 jobs are now readily solved[3)].

Then, agent k is to adopt the optimal solution as its current job assignment and inform its neighbors of the assignment. The assignment (along with other information related to the convergence detection process) is sent via a *assign* message in the protocol.

3.3 Dual Problem

When knowing the current job assignments of its neighbors, agent k solve the lagrangean dual problem to update the values of $\{\mu_j \mid j \in R_k\}$. In this work we adopt the *subgradient optimization method*, which is a well known technique for systematically updating a lagrangean multiplier vector[5]. In our method, under the current values of decision variables, agent k first calculates *subgradient* G_j for the assignment constraint on each job $j \in R_k$ as follows:

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

Then, using *step length* l_t which decays at rate r $(0 < r \leq 1)$ as time t, the round in this case, passes (i.e., $l_{t+1} = rl_t$), agent k updates μ_j as

$$\mu_j = \mu_j - l_t G_j. \quad (5)$$

We should notice that since μ_j is attached to job j all of the agents in S_j must agree on a common value to μ_j. If the agents in S_j assign different values to μ_j, then neither theorem 1 nor 2 holds any more. To set a common value to μ_j, 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, and prohibit each of the agents from working at round $t+1$ until it receives all of the assign messages issued from its neighbors at round t. By doing this, without performing explicit communication among S_j, we can make agents automatically set a common value to a lagrangean multiplier.

3.4 Convergence Detection

We can terminate the protocol when all of the optimal solutions to $\{LGMP_k(\mu) \mid k \in A\}$ satisfy the assignment constraints, $\sum_{i \in A} x_{ij} = 1,\ \forall j \in J$, for some values to a lagrangean multiplier vector μ, because theorem 2 suggests that these optimal solutions constitute an optimal solution to the GAP. The problem of detecting this fact is virtually the same with the solution detection problem which typically arises in the *distributed constraint satisfaction*[7]. The solution detection process is easily implemented for a locally-synchronized type of distributed constraint satisfaction solver such as the *distributed breakout algorithms*[2]. Thus, we incorporate a similar detection process into the protocol.

3.5 Convergence to Feasible Solution

In the protocol, a feasible solution, which is also an optimal solution according to theorem 2, is found for the first time when the termination condition is met. That is, until the termination condition is met, the respective assignment at each round is not feasible (although, according to theorem 1, an upper bound for the optimal value of the GAP may be calculated using the assignment). The question is whether the protocol always converges to a feasible solution that is also an optimal solution. An answer is unfortunately no, and thus you should have forced the protocol to terminate at a certain number of rounds without obtaining any feasible solution.

For the combinatorial optimization problems in a centralized context, even if being terminated on the way to an optimal solution, the lagrangean relaxation method can usually find a feasible solution because it equips a domain-specific technique, called the *lagrangean heuristics*, to transform the "best" infeasible solution into a feasible one. However, it may be difficult to devise such a heuristic in our protocol, because such the "best" infeasible solution, which we think belongs to global information, is inaccessible in a distributed context.

Therefore, we introduce a simple technique to make the agents quickly agree on a feasible solution with reasonably good quality. This is realized simply by replacing the multiplier updating rule (5) to another. As mentioned in Section 3.3, for any job j, all of the agents in S_j must agree on a common value to the respective multiplier μ_j. This is necessary for theorems 1 and 2 to be true. On the other hand, we observe that a common value to μ_j sometimes yields an oscillation, where some of the agents in S_j repeat to "cluster and disperse" around job j, and makes the protocol fail to find an optimal solution. Thus, we relax this requirement by letting the agents in S_j assign slightly different values to μ_j. In this work, we introduce a parameter δ $(0 \leq \delta \leq 1)$ that controls the range of "noise" mixed in with the increment/decrement in a lagrangean multiplier. The noise, denoted as N_δ, is a random variable whose value is uniformly distributed over $[-\delta, \delta]$. The multiplier updating rule (5) is thus replaced by

$$\mu_j = \mu_j - (1 + N_\delta) l_t G_j. \quad (6)$$

This rule diversifies agents' views on the value of μ_j, which is sometimes interpreted as the

price of job j, and thus being able to break an oscillation. On the other hand, since this rule violates the theorems, the protocol may converge to a feasible, but not optimal, solution. Note that the rule (6) is equal to the rule (5) if δ is set to zero.

4. Experiments

We observed the relation between the values of δ, which controls the degree of noise, and the performance of the protocol. In the experiments we used problem instances, where there exist $m \in \{3, 5, 7\}$ agents that are numbered from 1 to m, each of which has 5 jobs, meaning that the total number of jobs, n, was $5m$, and tries to assign each of its jobs to some other agents (including myself) using one of the following *assignment topologies*.

chain The i^{th} agent ($i \in \{2, \ldots, m-1\}$) tries to assign each of its jobs to the $(i-1)^{th}$ agent, the $(i+1)^{th}$ agent, or myself. On the other hand, the 1^{st} agent does to either the 2^{nd} agent or myself; the m^{th} agent to either the $(m-1)^{th}$ agent or myself.

ring The i^{th} agent ($i \in \{2, \ldots, m-1\}$) tries to assign each of its jobs to the $(i-1)^{th}$ agent, the $(i+1)^{th}$ agent, or myself. On the other hand, the 1^{st} agent does to the m^{th} agent, the 2^{nd} agent, or myself; the m^{th} agent to the $(m-1)^{th}$ agent, the 1^{st} agent, or myself.

cmplt Each agent tries to assign each of its jobs to any of the agents (including myself).

rndm3 Each agent tries to assign each of its jobs to any of the three agents, two of which are randomly selected from the other agents for each job and the other of which is myself.

We generated a random instance consisting of m agents with one of the above assignment topologies by randomly selecting an integer value from $[1, 10]$ for both w_{ij} and p_{ij}. On the other hand, we fixed c_i to 20 for any agent i in every instance. To ensure the feasibility of a problem instance, we checked if a generated instance is feasible by using a centralized exact solver to screen out infeasible ones.

The protocol was implemented in Java. The agents in the protocol can exchange messages using TCP/IP socket communication on a specific port. In the experiments, we put m agents in one machine and made them communicate using its local port. The parameters for the protocol were fixed as follows: *cutOffRound* $= 100n$, $l_0 = 1.0$, and $r = 1.0$, where *cutOffRound* is the upper bound of rounds at which a run is forced to terminate, l_0 is an initial step length for the agents, and r is a decay rate of the step length for the agents. The parameter δ controlling the degree of noise is ranged over $\{0.0, 0.3, 0.5, 1.0\}$. For each problem instance, 20 runs were made at each value of δ (except for 0.0) and the following data were measured:

Opt.R the ratio of the runs where optimal solutions were found;

Fes.R the ratio of the runs where feasible solutions were found;

Avg/Bst.Q the average/best value of the solution qualities;

Avg.C the average value of the numbers of rounds at which feasible solutions were found;

Note that, in the Avg/Bst.Q, the solution quality was measured as the ratio of the profit of an obtained feasible solution to the optimal value. Note also that, when a run finished with no feasible solution, we did not count the run for Avg/Bst.Q, but did count using the value of *cutOffRound* for Avg.C. On the other hand, when the value of δ is 0.0, we made only one run because there is no randomness in the protocol with $\delta = 0.0$.

The results are shown in Table 1. The Pr.ID column shows a label of a problem instance meaning, from left to right, assignment topology, the number of agents, the total number of jobs, an available resource capacity, and an instance identifier.

As we mentioned in Section 3.5, the rule (6) is equal to the rule (5) when the value of δ is 0.0. The protocol with this setting, therefore, is to terminate only when an optimal solution is found (otherwise, it is forced to be terminated when the cutoff round, $100n$, is reached). However, in the experiments, we observed that the protocol with that setting failed to find optimal solutions within the cutoff rounds for almost all of the problem instances (for 19 out of 20 instances).

On the other hand, the performance of the protocol was dramatically changed when δ was set to one of the non-zero values. The results show that Opt.R, Fes.R, and Avg.C are ob-

Table 1 Experimental results

Pr.ID	δ	Opt.R	Fes.R	Avg.Q	Bst.Q	Avg.C	Pr.ID	δ	Opt.R	Fes.R	Avg.Q	Bst.Q	Avg.C
chain-3-15-20-002	0.0	0/1	0/1	N/A	N/A	1500	rndm3-5-25-20-000	0.0	0/1	0/1	N/A	N/A	2500
	0.3	13/20	20/20	0.990	1.000	82.0		0.3	8/20	20/20	0.977	1.000	527.3
	0.5	8/20	20/20	0.984	1.000	93.2		0.5	6/20	19/20	0.965	1.000	428.8
	1.0	6/20	20/20	0.962	1.000	69.8		1.0	2/20	20/20	0.951	1.000	288.6
chain-3-15-20-004	0.0	1/1	1/1	1.000	1.000	22	rndm3-5-25-20-001	0.0	0/1	0/1	N/A	N/A	2500
	0.3	16/20	20/20	0.988	1.000	70.5		0.3	19/20	20/20	0.998	1.000	40.6
	0.5	14/20	20/20	0.993	1.000	32.7		0.5	17/20	20/20	0.994	1.000	66.6
	1.0	6/20	20/20	0.943	1.000	74.1		1.0	10/20	20/20	0.957	1.000	53.8
cmplt-3-15-20-000	0.0	0/1	0/1	N/A	N/A	1500	chain-7-35-20-000	0.0	0/1	0/1	N/A	N/A	3500
	0.3	4/20	20/20	0.974	1.000	224.3		0.3	3/20	19/20	0.969	1.000	810.7
	0.5	1/20	20/20	0.977	1.000	93.4		0.5	1/20	19/20	0.965	1.000	505.3
	1.0	1/20	20/20	0.952	1.000	57.2		1.0	0/20	19/20	0.959	0.986	473.8
cmplt-3-15-20-001	0.0	0/1	0/1	N/A	N/A	1500	chain-7-35-20-001	0.0	0/1	0/1	N/A	N/A	3500
	0.3	6/20	20/20	0.972	1.000	216.9		0.3	5/20	20/20	0.983	1.000	359.0
	0.5	8/20	20/20	0.978	1.000	93.4		0.5	2/20	20/20	0.977	1.000	223.1
	1.0	0/20	20/20	0.933	0.980	91.7		1.0	3/20	20/20	0.966	1.000	283.5
chain-5-25-20-000	0.0	0/1	0/1	N/A	N/A	2500	ring-7-35-20-000	0.0	0/1	0/1	N/A	N/A	3500
	0.3	5/20	20/20	0.989	1.000	269.6		0.3	3/20	18/20	0.959	1.000	993.7
	0.5	2/20	20/20	0.978	1.000	177.9		0.5	2/20	20/20	0.955	1.000	359.5
	1.0	1/20	20/20	0.964	1.000	148.0		1.0	0/20	20/20	0.946	0.996	164.0
chain-5-25-20-001	0.0	0/1	0/1	N/A	N/A	2500	ring-7-35-20-001	0.0	0/1	0/1	N/A	N/A	3500
	0.3	11/20	20/20	0.995	1.000	84.2		0.3	1/20	20/20	0.949	1.000	1013.0
	0.5	13/20	20/20	0.993	1.000	146.3		0.5	0/20	20/20	0.939	0.992	479.1
	1.0	3/20	20/20	0.985	1.000	72.4		1.0	0/20	20/20	0.935	0.983	278.3
ring-5-25-20-000	0.0	0/1	0/1	N/A	N/A	2500	cmplt-7-35-20-000	0.0	0/1	0/1	N/A	N/A	3500
	0.3	1/20	19/20	0.966	1.000	832.6		0.3	0/20	20/20	0.929	0.977	559.3
	0.5	1/20	20/20	0.966	1.000	478.9		0.5	0/20	20/20	0.919	0.984	298.8
	1.0	1/20	20/20	0.957	1.000	362.9		1.0	0/20	20/20	0.865	0.928	151.4
ring-5-25-20-001	0.0	0/1	0/1	N/A	N/A	2500	cmplt-7-35-20-001	0.0	0/1	0/1	N/A	N/A	3500
	0.3	5/20	20/20	0.970	1.000	373.6		0.3	0/20	20/20	0.938	0.987	488.4
	0.5	2/20	20/20	0.968	1.000	176.8		0.5	0/20	20/20	0.911	0.970	288.1
	1.0	1/20	20/20	0.939	1.000	161.6		1.0	0/20	20/20	0.883	0.960	179.9
cmplt-5-25-20-000	0.0	0/1	0/1	N/A	N/A	2500	rndm3-7-35-20-001	0.0	0/1	0/1	N/A	N/A	3500
	0.3	0/20	20/20	0.934	0.980	423.9		0.3	1/20	18/20	0.971	1.000	1213.0
	0.5	0/20	20/20	0.923	0.980	208.6		0.5	0/20	18/20	0.962	0.996	735.2
	1.0	0/20	20/20	0.881	0.959	182.5		1.0	1/20	18/20	0.951	1.000	966.3
cmplt-5-25-20-001	0.0	0/1	0/1	N/A	N/A	2500	rndm3-7-35-20-002	0.0	0/1	0/1	N/A	N/A	3500
	0.3	1/20	20/20	0.954	1.000	245.7		0.3	3/20	16/20	0.967	1.000	1507.0
	0.5	0/20	20/20	0.944	0.981	121.3		0.5	0/20	20/20	0.939	0.996	735.2
	1.0	0/20	20/20	0.908	0.953	111.4		1.0	1/20	20/20	0.916	1.000	507.9

viously improved while Avg/Bst.Q is kept at a reasonable level. This suggests that using the protocol with those settings the agents can quickly agree on a feasible solution with reasonably good quality.

It is also the fact that the protocol with $\delta \neq 0.0$ 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 instances (cmplt-5-25-20-000, cmplt-7-35-20-000, and cmplt-7-35-20-001). However, for each of the other instances, an optimal solution was found in at least one run at some value of δ.

For many instances (12 out of 20 instances), when the value of δ increses we can see that both Avg.C and Avg.Q get lower. In other words, increasing the value of δ may generally have an effect to rush the agents into reaching a compromise, i.e., a lower-quality solution. This implies that the parameter δ may allow you to control a tradeoff between the quality of a solution and the cost of finding it.

5. Conclusion

We have presented the generalized mutual assignment problem (GMAP) and a solution protocol for solving the GMAP. In the protocol, we have also introduced the parameter controlling the degree of noise mixed in with the increment/decrement in a lagrangean multiplier. In our future work, we would like to pursue more sophisticated techniques to update a lagrangean multiplier vector and the method that would realize distributed calculation of the upper bound for the optimal value.

References

1) I. P. Androulakis and G. V. Reklaitis. Approaches to asynchronous decentralized decision making. Computers and Chemical Engineering 23, pp.341–355, 1999.
2) K. Hirayama and M. Yokoo. The distributed breakout algorithms. Artificial Intelligence (to appear).
3) S. Martello, D. Pisinger, and P. Toth. New trends in exact algorithms for the 0-1 knapsack problem. European Journal of Operational Research 123, pp.325–332, 2000.
4) I. H. Osman. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. OR Spektrum 17, pp. 211–225, 1995.
5) C. R. Reeves. Modern heuristic techniques for combinatorial problems. Blackwell, 1993.
6) R. G. Smith. The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Transaction on Computers 29(2), pp. 1104–1113, 1990.
7) M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara The Distributed constraint satisfaction problem: formalization and algorithms. IEEE Transactions on Knowledge and Data Engineering 10(5), pp. 673–685, 1998.