Coordinated Problem Solving Through Resource Sharing in a Distributed Environment

Umesh Deshpande, Arobinda Gupta, and Anupam Basu

***Abstract*—An important feature in a distributed problem solving system is that the resources of different nodes can be shared through cooperation. In this paper, the generalized partial global planning (GPGP) approach used for multiagent systems is extended by providing a coordination mechanism for resource sharing across nodes. In our framework, multiple conflicting criteria (or objectives) like quality, cost, and duration may be associated with an input task. Preference ratings expressed subjectively may be assigned to each of the criteria. Task assignment in this system, which is a multiobjective decision making problem, is important for the satisfaction of the criteria. It has to be done with imprecise information since the system is dynamic and preference ratings are specified subjectively. A technique for task assignment using the fuzzy set approach is also presented in this paper. Simulation studies for the coordination mechanism and the task assignment have been performed to demonstrate their effectiveness.**

***Index Terms*—Contract net protocol, distributed problem solving, fuzzy logic, multiagent systems, multiobjective decision making.**

I. Introduction

In a distributed problem solving system, agents at different nodes (problem solvers) coordinate their individual problem solving activities to solve the complete problem [1]. One of the important features of these systems is that the resources of the different nodes can be shared through cooperation. Examples of such systems are a distributed health care system [2], which is a consortium of hospitals cooperating with one another, an e-University system [3] where different universities or colleges collaborate to offer courses to the students, supply chain management systems etc. Often such systems have to work under real-time constraints with deadlines (may be soft) associated with tasks.

Resource sharing at a *single node* has been dealt with in the generalized partial global planning (GPGP) ([4], [5]) framework of multiagent systems (MAS) in [6]. A coordination mechanism to handle mutually exclusive resources has been added in a framework for task analysis, environment modeling, and simulation (TAEMS) [4]. This has been demonstrated with an application to a hospital patient scheduling system. This framework, however, *does not* handle the case when different nodes (each a MAS in itself) have to coordinate for sharing resources. We extend the GPGP approach by providing a coordination mechanism to allow sharing of resources across nodes. This enables the realization of applications like a distributed hospital system where some hospitals may not have all the required resources.

Task migration is an important issue for resource sharing in a coordinated problem solving environment. If a subtask T' of a task T submitted to a node N_i requires a resource r that is not present at N_i then T' has to be migrated to some other node of the system where r is present. For example, in a distributed hospital system, a patient may need an MRI scanner which may not be present at a hospital, then the part of the therapy that requires the MRI scanner has to be done at some other hospital. The time required for the migration, the loads on the remote nodes, and the quality and cost of usage of the resources at the remote nodes are important in making the migration decision. In a practical situation, multiple conflicting criteria (or objectives) like minimizing cost and time required, maximizing quality, etc. could be associated with a task. In this paper, we first propose a coordination mechanism that makes the subtask migration decision based on only a single objective of minimizing the average waiting time for all the tasks. A distributed hospital environment is simulated using a multiagent framework and comparison studies of our coordination mechanism with other commonly used schemes has been carried out. The performance evaluation results show that our scheme performs better than the other approaches.

In our system, an user could assign a preference rating to each of the criteria. The preference ratings denote the relative importances for the criteria. Task assignment, which is a multiobjective decision making problem, is important for the satisfaction of the criteria. We address this important issue in our framework.

At the time of the task submission, the user normally has only an intuitive and imprecise notion of the preference ratings to be associated with the objectives. These could be expressed using subjective ordinal values or linguistic terms. For example, a patient in a hospital, may want a *fast* response with *good* quality at a *moderate* cost. Imprecision also arises since the system under consideration is dynamic and distributed. The global system state is not available at any single node and hence it has to be collected before the task assignment. Since the utilization of the resources changes continuously, it is not possible to get a precise system state. Hence, the decision making needs to be done allowing for tolerance for the imprecision. In this paper, we adapt the multiobjective decision making process using fuzzy logic for task assignment in a distributed system. The earlier proposed coordination mechanism is then extended to deal with multiple objectives and subjective preference ratings. Simulation studies of the task assignment demonstrate significant performance gains of our technique over the crisp techniques of task assignment.

In distributed problem solving (DPS) systems, a widely used scheme for task allocation is the contract net protocol (CNP) [7]. The objective function in this consists of a single parameter—the "utility" of the overall system. Market based approaches, like the one mentioned in [8] are also popular for task allocation, where a single measure "price" drives the allocation process. Auctions are often used for pricing as in [9].

In the present paper, we consider cooperative agents that strive to optimize an objective function consisting of several parameters for task allocation. Moreover, the relative preferences over the parameters may often be *subjective*. To the best of our knowledge, there is no previous work where multiple criteria based objective function governs task assignment as is presented here. The contributions of the paper are as follows.

1) A coordination mechanism for resource sharing across nodes is first proposed for a single objective of minimizing the average waiting time.
2) A technique for the assignment of a task to a node in the system in the presence of multiple criteria with subjective preference ratings is presented.
3) The earlier proposed coordination mechanism is then extended to incorporate multiple criteria with subjective preference ratings.

The paper is organized as follows. The system model is presented in Section II. Section III explains the coordination mechanism. Simulation study of the coordination mechanism is discussed in Section IV.

Manuscript received December 12, 2002; revised April 10, 2003. This work was supported by a Research Grant received from the Ministry of Human Resources and Development, Government of India. This paper was recommended by Associate Editor F. Gomide.

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, West Bengal 721302, India (e-mail: uad@cse.iitkgp.ernet.in, agupta@cse.iitkgp.ernet.in, anupam@cse.iitkgp.ernet.in).

Digital Object Identifier 10.1109/TSMCB.2003.818535

1083-4419/04$20.00 © 2004 IEEE

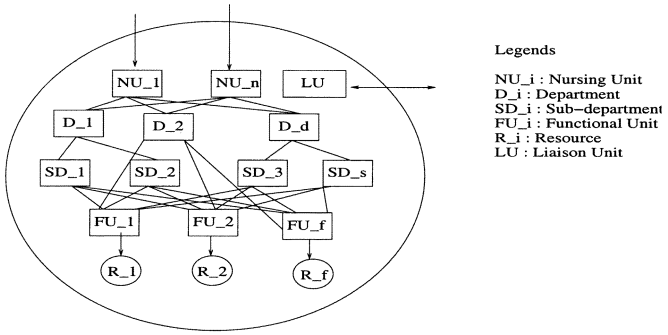

Fig. 1. Architecture of a node in a distributed hospital system.

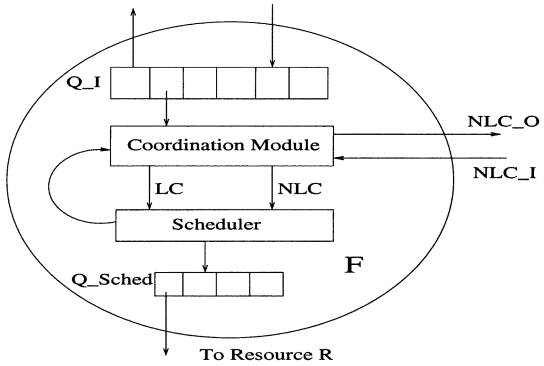

Fig. 2. Model of an agent F at a functional unit.

The task assignment problem and its solution is presented in Section V. The performance evaluation of the task assignment is presented in Section VI. Section VII concludes the paper.

II. System Model

There are n nodes $N_1, N_2, \ldots, N_n$ in a distributed system which communicate only through messages. Each node has a collection of agents controlling the resources present at that node. The nodes are logically completely connected. It is assumed that the communication delay for sending/receiving messages between any two nodes N_i and N_j can be estimated and is a constant Ψ_{ij}. A special unit called the *liaison unit* (LU), with an agent in it, is identified at every node that is responsible for coordination across nodes for resource sharing. All communication across nodes takes place through the LU. A task, specified in a manner similar to that of TAEMS, can be submitted to any node with an *appropriate deadline* and *preference ratings* associated with the different objectives.

As an example system, the architecture of a node (a hospital) in a distributed hospital system is shown in Fig. 1. Agents are present at each of the nursing units, departments, sub-departments, functional units, and the liaison unit. The figure shows a hierarchical organization with the resources controlled by the functional units. In the following subsection, the modeling of the agent at a functional unit is explained. The detailed explanation of the modeling of the other agents is presented in [2].

A. Modeling of the Agent at a Functional Unit

Assume that the agent F (as shown in Fig. 2) at a functional unit controls the resource R which has a duration of usage Δ. F consists of an input queue Q_I, a coordination module, and a scheduler that work together to prepare schedules for the input tasks. The schedule queue Q_{Sched} contains the dispatch times of the tasks input to F. The agents coordinate with one another using commitments [4]. The local commitments (LC) for F are those which the coordination module of F passes on to the local scheduler of F. The nonlocal commitments (NLCs) for F are those which the coordination module of F exchanges with the coordination module of some other agent. NLC_I is the set of commitments that are input from other agents to F and NLC_O are the set of output commitments passed on to other agents by F. The scheduler uses the LC and the NLC_I sets to generate schedules.

III. Mechanism for Coordination Across Nodes

The mechanism is based on modifications to the contract net protocol proposed in [10]. The basic idea of the proposed mechanism is based on the process of bidding. When a task at a node N requires a resource that is not present locally, then N finds out if there exists a node, called the *focused addressing node*, where the request for the service of that resource is most likely to be satisfied. In addition to this node, it also asks for bids from a subset of the nodes present in the network. If the request can be satisfied at the focused addressing node, all the bids by other nodes are ignored. Otherwise, the node with the best bid is chosen and the task is migrated to it.

A. Liaison Unit: Data Structures and Algorithms

The LU on each node N_i maintains the following two tables.

Local Resource Table (LRT)—The format and the contents are discussed below.

Name	Duration	Surplus

Name is the name of the resource present locally at the node N_i. Duration is the time required for the usage of the resource. Surplus is the ratio of the time the resource was not used in a past window of length WL_k. WL_k is chosen appropriately depending on the duration for the usage of the resource r_k. The surplus is an indication of the load on a resource. Each agent of the node sends the surplus to the LU with a period of the window length for that resource. The surplus information is sent across nodes periodically and it is also piggybacked with message exchanges between nodes.

Remote Resource Table (RRT)—This table keeps information about the resources at other nodes in the network as known by N_i. The format and the contents are as follows.

Name	Id	Duration	Surplus	Delay Estimate

Id is the identifier for the node where the mentioned resource with name Name is present. Duration is the time for the usage of the resource at that node. Surplus is the one that was most recently obtained from the remote node. For each resource, the entries are ordered in the decreasing order of the surplus. Delay Estimate at the node N_i is computed for every $\langle N_j, r_k \rangle$ pair where the resource r_k is not present at N_i but is present at a remote node N_j. Let us say that N_j promised a bid for time t_0 but the actual finish time of the service of r_k was t_1. The difference $t_1 - t_0$ is referred to as the delay. This delay can be estimated for every resource present at each remote node based on the previous history of the $\langle \mathrm{node},\ \mathrm{resource} \rangle$ pair. This is called the $delay\ estimate$. It is calculated every time a remote node finishes servicing a request for a resource. The delay estimate for the $nth(n > 0)$ request for a resource r_k requested by a source node s and serviced at a destination node d is calculated using exponential smoothening described as

$$\tau_n = \alpha_{kd} * t_{n-1} + (1 - \alpha_{kd}) * \tau_{n-1}$$

where τ_{n-1} is the estimated delay for the $(n-1)^{th}$ request ($\tau_0 = 0$), t_{n-1} is the difference between the actual finish time (the task result arrival time) and the bid promised for the $(n-1)th$ request, and α_{kd} is a value ($0 \leq \alpha_{kd} \leq 1$) chosen appropriately for the $\langle d, r_k \rangle$ pair.

B. Bid Process

If a resource r_k for a subtask T is not present at a node N_i, a request is made to the LU of N_i. Alongwith the request, the *expected worst case start time* (EWCST) is also sent. Requests that are likely to be started after EWCST are not good since the deadlines would be missed.

Request for bids (RFBs) are sent to those nodes from the remote resource table (RRT) for r_k such that each such node N_j satisfies the condition that $4^* \Psi_{ij}$ + delay estimate+current time > EWCST. The multiplication by four is used since there would be four communications between N_i and N_j (first for requesting the bid by N_i, second for sending the bid by N_j, third for sending the task by N_i, and the last for sending the results by N_j). RFBs are sent at the maximum to the first ρ nodes of the RRT where ρ is a system-wide parameter used to minimize the communication overhead. If the first node of the RRT satisfies the additional condition that its $\mathrm{surplus} > \Theta_k$, then it is chosen for focused addressing. Θ_k is a threshold and is a constant chosen depending on the resource r_k.

When a node, say N_h, receives a RFB message, it checks if it can guarantee the request. If the request can be guaranteed, it sends the *earliest possible finish time* (EFT). If the focused addressing node replies that it can guarantee T, N_i will ignore bids by other nodes. Otherwise, N_i would wait for bids from other nodes and would select the best bidder. The best bidder is the one that returns the least value of $2^* \Psi_{ih}$ + EFT. The multiplication factor is two here, since only the last two communications of the four mentioned above would be required after a bid is obtained.

N_i will wait for bids until it gets a favorable one or till $(EWCST - 2 * \Psi_{\max})$ whichever is earlier. The worst case delay $\Psi_{\max}$ is the maximum time required to communicate between any two nodes. Still, if no favorable bids are available, the subtask T could be migrated randomly to any node.

The coordination mechanism requires various parameters like EWCST, EFT, and the guarantee routine which are provided by a real-time scheduler. The design of the scheduler and the computation of these parameters is presented in detail in [2].

IV. Evaluation of the Coordination Mechanism

A network of six nodes is simulated for carrying out the experiments. In the simulation model of this environment, the focus is on the number and distribution of the resources. A total of 43 resources are distributed randomly across the nodes. The duration of the usage of the resources is randomly chosen. The task arrival is modeled as a Poisson process with different arrival rates at each node. Let X denote the *system wide arrival rate*. The performance measures are the *average task waiting time* (W) and the *guarantee ratio* (G = number of deadlines guaranteed/total number of tasks).

Experiments are conducted for the following three cases of agents present at the LU.

- *random Node Case*, where an agent selects a node randomly when a required resource is not present locally.
- *nearest Node Case*, where an agent selects a closest node in the network where the resource is available.
- *contract Net Protocol (cnp) case*, where an agent uses the coordination mechanism proposed in Section III.

The average task execution time is 300 min of simulated time. Each of the experiments is executed for a total simulated time of 30 days. The

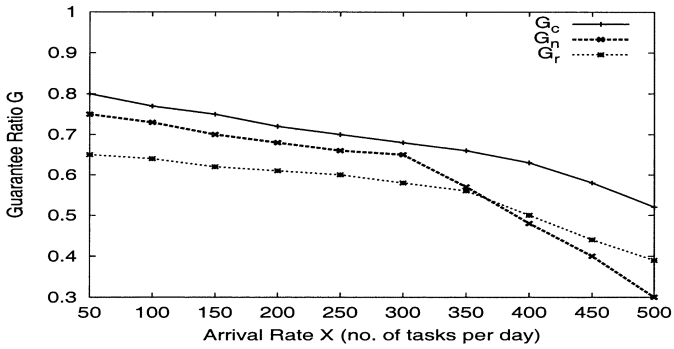

Fig. 3. Guarantee ratio with four heavily and two lightly loaded nodes.

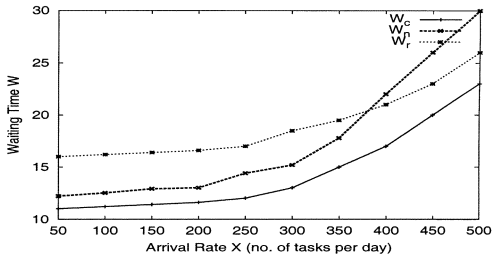

Fig. 4. Waiting time with four heavily and two lightly loaded nodes.

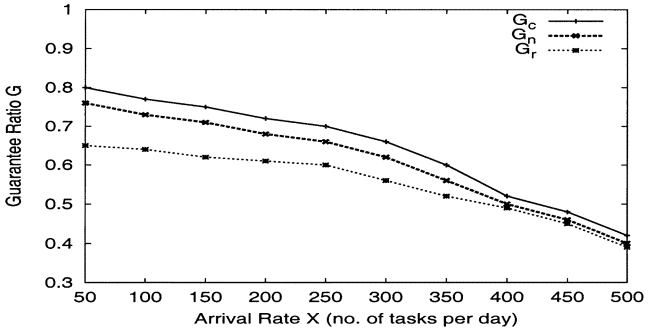

Fig. 5. Guarantee ratio with three heavily and three lightly loaded nodes.

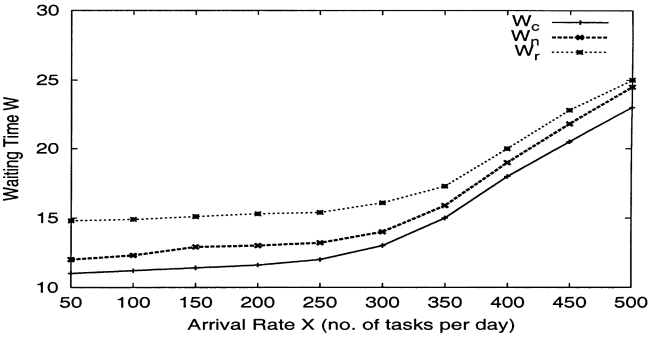

Fig. 6. Waiting time with three heavily and three lightly loaded nodes.

G and W values in all the three cases with X varying from 50–500 tasks per day are noted. Let G_c, G_r, and G_n be the guarantee ratios in the *cnp*, random node, and nearest node cases respectively and W_c, W_r, and W_n be the average waiting times in the *cnp*, random node, and nearest node cases, respectively. Figs. 3–6 show the results.

The graphs indicate that the *cnp* case performs consistently better than both the random node and the nearest node cases in all the scenarios. This is because the coordination mechanism in Section III takes

into account both the loads at the different nodes and the travel time required for migration. In the best possible case, the *cnp* case performs about 18% better than the nearest node case and about 20% better than the random node case for the guarantee ratio. It performs about 38% better than the nearest node case and about 45% better than the random node case for the average waiting time. Thus we are justified in spending more time in the *cnp* case for making a good decision for task migration.

As observed, the nearest node case performs better, in general, than the random node case. Specially, when the arrival rates are low, its performance is close to the *cnp* case. This is because the nearest node case is a more informed decision than the random node case since it takes into account the travel times. When the load is low, the travel times play an important role, but as is seen from Figs. 3 and 4, when there are more highly loaded nodes in the system and the arrival rates are high, the nearest node case performs worse than the random node case. The reason is that the nearest node case would make many bad decisions at high loads since its decision is solely based on the travel time without giving any consideration to the node loads. The random node case would spread over the migrated tasks across the system and hence, would perform better.

When the system load is almost equally distributed across all the nodes and when the arrival rate is high (above 350 in Figs. 5 and 6), all the three cases perform almost equally well with the *cnp* case being slightly better. In such a case there is very little to choose between the nodes. The coordination mechanism does not have much gains since it spends time in the bidding process itself. Still, the *cnp* case performs about 6% better than the nearest node case and about 8% better than the random node case for the guarantee ratio. It performs about 5% better than the nearest node case and about 6% better than the random node case for the average waiting time.

In this section, the simulation results of the coordination mechanism proposed in Section III were discussed. In the next section, a technique for assigning tasks to a node of the system in the presence of multiple objectives and subjective preference ratings is presented.

V. Task Assignment With Imprecision

Task assignment in the system could be done by any of the nodes of the system or by a special node used solely for that purpose. The node where the decision is taken has to collect the global state first. As discussed in Section I, there is imprecision in the state collection because of the dynamic nature of the system.

This section is organized as follows. The multiobjective decision making process using fuzzy logic is reviewed in Section V-A. In Section V-B, the computation of the fuzzy sets for each of the objectives is explained. The coordination mechanism is extended in Section V-C to incorporate the multiple objectives and the preference ratings.

A. Multiobjective Decision Making Using Fuzzy Logic

The typical multiobjective decision problem involves the selection of one alternative a_i from a universe of alternatives containing n elements, $A = \{a_1, a_2, \ldots, a_n\}$ given a set of p elements, $O = \{O_1, O_2, \ldots, O_p\}$, of criteria or objectives that are important to the user. In [11], each objective has been represented as a fuzzy set over the set of alternatives $\{A\}$. Thus, the grade of membership, $\mu_{O_i}(a)$ of the alternative a in the ith objective O_i denotes the degree to which a satisfies the criteria specified by O_i. For a collection of objectives, the decision function $D = \bigwedge_{i=1}^{p}(O_i)$. The optimum solution a^* will be the alternative that maximizes D. According to [12], a set of linear and ordinal preferences $\{P\}$ can be introduced along with the objectives. These preferences can be linguistic values specifying the attribution of the user for the importances of each of the objectives. If $P = \{b_1, b_2, \ldots, b_p\}$, then the decision function will be a joint intersection as follows.

$$D = \bigwedge_{i=1}^{p}(b_i \rightarrow O_i) = \bigwedge_{i=1}^{p}\left(\bar{b}_i \bigvee O_i\right).$$

The optimum solution a^* is the alternative that maximizes D. Let us define $C_i = \bar{b}_i \bigvee O_i$ and hence, $\mu_{C_i}(a) = \text{Max}[\mu_{\bar{b}_i}(a), \mu_{O_i}(a)]$, then the optimum solution expressed in membership form, is given by the following.

$$\mu_D(a^*) = \text{Max}_{a \in A}\left[\min\left\{\mu_{C_1}(a), \mu_{C_2}(a), \ldots, \mu_{C_p}(a)\right\}\right].$$

A special procedure, as explained in [12], needs to be followed in the event of a tie between two or more alternatives.

B. Computation of the Fuzzy Sets

For task assignment, the alternatives, A, are the different nodes of the system, i.e., $A = \{N_1, N_2, \ldots, N_n\}$. There are four objectives defined in this setup—*time criticality* (TC), *quality* (Q), *cost* (C), and *migration overhead* (MO). The time criticality, quality, and cost objectives have the obvious meanings but the migration overhead is not so obvious. When a task T is input to a node of the system, all the resources required by T may not be present at that node. The migration overhead, both in terms of time and cost, is incurred for the migration of subtasks of T to the remote nodes since some resources required for T may not be locally present. For example, in a distributed hospital system, the migration overhead corresponds to the time and cost required for shifting from one hospital to another. The user can give a rating for the overhead depending on the number of migrations he can tolerate.

The objectives $O = \{O_1, O_2, O_3, O_4\} = \{TC, Q, C, MO\}$ are defined for every node of the system. When a task T is input into the system, the preference ratings P associated with T are also input where $P = \{b_1, b_2, b_3, b_4\}$. The four objectives $O = \{TC, Q, C, MO\}$ are defined using fuzzy sets as follows.

$$TC = \left\{\frac{h_{11}}{N_1}, \frac{h_{21}}{N_2}, \ldots, \frac{h_{n1}}{N_n}\right\} \quad Q = \left\{\frac{h_{12}}{N_1}, \frac{h_{22}}{N_2}, \ldots, \frac{h_{n2}}{N_n}\right\}$$
$$C = \left\{\frac{h_{13}}{N_1}, \frac{h_{23}}{N_2}, \ldots, \frac{h_{n3}}{N_n}\right\} MO = \left\{\frac{h_{14}}{N_1}, \frac{h_{24}}{N_2}, \ldots, \frac{h_{n4}}{N_n}\right\}.$$

In the fuzzy sets, a value h_{ij}, which is the membership value of a node N_i for an objective O_j, indicates how good N_i is with respect to the criteria O_j. We refer to the h_{ij} value as the *satisfaction measure*. In our case, all the *satisfaction measures* and the preference ratings are from the unit interval [0, 1]. Other cases of specifying the preference ratings like linguistic terms and ordinal values can also be mapped onto the unit interval. The fuzzy sets of the objectives are computed when a task T is input taking into account the current available state of the system and coordination mechanism of Section III. The local state at every node is available in the Local and Remote Resource Tables (LRT and RRT) of the LU.

The computation of the *satisfaction measures* cannot be precise since the parameters required for the computation are not available *a-priori*. For example, the completion time for a task T at a node $N_i$cannot be known *a-priori*. Also, those subtasks of T that require a resource r not present at N_i would have to be migrated. The destination node of the migration is not known *a-priori* and has to be found out using the coordination mechanism of Section III. In all the following computations,

the destination node is assumed to be the best node N_b. The best node is the one having the maximum surplus (as known by N_i from its RRT).

Using the *satisfaction measures*, the assignment of T to one of the nodes N_A of the system has to be done to optimize the *average performance measure*. The performance measure, P_T, for a task T is the weighted average of the values obtained for each of the objectives after the execution of T with the weights being the preference ratings. The *average performance measure* is the average of P_T over all the incoming tasks. The multiobjective decision process discussed in Section V-A is executed and T is assigned to the node identified by the process. In the following subsections, the computation of the satisfaction measures for the various objectives is explained.

1) Time Criticality Satisfaction Measures: The values h_{i1} for all values of i between 1 and n indicate the goodness of a node N_i with respect to the time criticality factor. First, an estimate of the time required for T at N_i, TR_i, is computed. To obtain this, first consider a resource r present at N_i and required by T. The duration of r at N_i is divided by the surplus of r at N_i (found from the LRT of N_i). This gives an approximate measure of the time required for r. When the surplus is very low, a bound on this measure is made. Summation Sum_1 of this computation for all such resources is done.

Next, consider a resource required by T but not present at N_i. In this case, the duration of r at the best node N_b is divided by the surplus of r at N_b. Moreover, we also have to consider the time for migrating subtasks of T to N_b. Hence the term $2 * \Psi_{ib}$ is added. The multiplication by 2 is there since two communications between N_i and N_b would be required—one for sending the task by N_i and other for sending the results by N_b. Summation Sum_2 of this computation for all such resources is done. The time required by T at N_i, TR_i, is estimated to be the addition of the two summations Sum_1 and Sum_2.

In the following expression for TR_i, the term $\forall r \in T \wedge r \in N_i$ denotes a resource r that is required by T and is present at N_i and the term $\forall r \in T \wedge r \notin N_i$ denotes a resource r that is required by T but is not present at N_i. TR_i for all values of i between 1 and n is calculated as follows.

$$\text{TR}_i = \sum_{\forall r \in T \wedge r \in N_i} \frac{\Delta_i^r}{S_i^r} + \sum_{\forall r \in T \wedge r \notin N_i} \left(\frac{\Delta_b^r}{S_b^r} + 2 * \Psi_{ib} \right)$$

where Δ_i^r is the duration of the usage for r at N_i and Δ_b^r is the duration of usage of r at the best node N_b. Similarly S_i^r is the surplus at the node N_i and S_b^r is the surplus at the best node N_b. Let $TR_{\min} = MIN_i(TR_i)$. Then, $h_{i1} = \text{TR}_{\min}/TR_i$.

2) Quality Satisfaction Measures: For finding h_{i2}, first the values of Q_i are found using the following expression.

$$Q_i = \sum_{\forall r \in T \wedge r \in N_i} (Q_i^r) + \sum_{\forall r \in T \wedge r \notin N_i} (Q_b^r)$$

where Q_i^r is the quality of r at N_i and Q_b^r is the quality of r at the best node N_b. For resources that are absent, the quality of the best node is chosen as before. Let $Q_{\max} = \text{MAX}_i(Q_i)$. Then $h_{i2} = Q_i/Q_{\max}$.

3) Cost Satisfaction Measures: For finding h_{i3}, first the values of C_i are found using the following expression.

$$C_i = \sum_{\forall r \in T \wedge r \in N_i} (C_i^r) + \sum_{\forall r \in T \wedge r \notin N_i} (C_b^r)$$

where C_i^r is the cost of r at N_i and C_b^r is the cost of r at the best node N_b. Let $C_{\min} = \text{MIN}_i(C_i)$. Then $h_{i3} = C_{\min}/C_i$.

4) Migration Overhead Satisfaction Measures: The values h_{i4} are computed as follows. First the estimation of the migration overhead, MO_i, for an input task T at a node N_i is carried out. The time required for the actual migration and the cost due to the migration are included in the calculation of MO_i.

$$MO_i = \sum_{r \in T \wedge r \notin N_i} \left((2 * \Psi_{ib}) + \left(MAX_{\forall j \; r \in N_j} \left(C_b^r - C_j^r \right) \right) \right).$$

The first term in the summation, $2 * \Psi_{ib}$, represents the actual migration time of the task. The second term indicates the cost overhead. C_j^r is the cost of the resource r at a node N_j and C_b^r is the cost of r at N_b. The maximum of the difference is chosen as a pessimistic estimate. MO_i is computed for all values of i. Let $MO_{\min} = MIN_i(MO_i)$. Then $h_{i4} = MO_{\min}/MO_i$.

C. Extending the Coordination Mechanism

The coordination mechanism proposed in Section III needs to be extended to incorporate multiple objectives and preference ratings. The multiobjective decision making process has to be applied when a subtask T' of a task T has to be migrated since a resource r required for T is not present locally. The preference ratings which the user had assigned for the various objectives for T should be used at the time of of making this decision. The basic coordination mechanism, as explained in Section III, remains the same but at the end of it all the nodes to whom the request for bid (RFB) was sent have to be evaluated against the preference ratings of the various objectives.

As explained in Section III-B, the bid returned by a destination node is the earliest finish time (EFT). Hence, $h_{i1} = \text{EFT}_{\min}/\text{EFT}_i$ where EFT_i is the EFT returned by a remote node N_i and $\text{EFT}_{\min}$ is the minimum of all the EFT's returned. $h_{i2} = Q_i^r/Q_{\max}^r$ where Q_i^r is the quality of r at N_i and $Q_{\max}^r$ is the maximum of the qualities of r across all the nodes where the RFB was sent. Similarly, $h_{i3} = C_{\min}^r/C_i^r$ where C_i^r is the cost of r at N_i and $C_{\min}^r$ is the minimum of the costs of r across all the nodes where the RFB was sent. $h_{i4} = \Psi_{\min}/\Psi_{si}$ where s is the source node and N_i is a node where the RFB was sent. $\Psi_{\min}$ is the minimum of Ψ_{si} among all the remote nodes where RFB was sent. h_{i1}, h_{i2}, h_{i3} and h_{i4} values are computed for all values of i between 1 and n. As before, the approach in Section V-A is used to find the target node.

VI. Performance Evaluation of the Task Assignment Algorithm

The experiments are performed on the same setup as explained in Section IV. The preference ratings for every task are randomly generated alongwith a task. When the task execution is over, a performance measure—which is the weighted average of the duration required, the quality accumulated, the cost, and the migration overhead incurred for the task execution with the weights being the preference ratings—is calculated. The performance measures for all tasks are summed up and the average performance measure P=(sum ofperformance measures for all tasks/total number of tasks) is calculated.

The *fuzzy* task assignment approach is compared with the two baselines of *random* and *round-robin* task assignment algorithms. Let P_{fuz}, P_{ran}, and P_{rr} be the performance measures using the *fuzzy*, the *random*, and the *round-robin* approaches, respectively. Graphs for $P_1 = P_{\text{fuz}}/P_{ran}$ and $P_2 = P_{\text{fuz}}/P_{rr}$ (which are the ratios of the performance measure of the *fuzzy* technique against the *random* and the *round-robin* techniques respectively) are plotted against the values of the arrival rate X varying from 50 tasks/day to 500 tasks/day. Each such experiment is performed for three different values of R, the number of resources required by a task; the values being $R = 5$, $R = 10$, and $R = 15$. Figs. 7 and 8 show the results. The graphs have to be interpreted as follows. Any point on a curve is a ratio comparing the fuzzy performance measure versus the random or round-robin

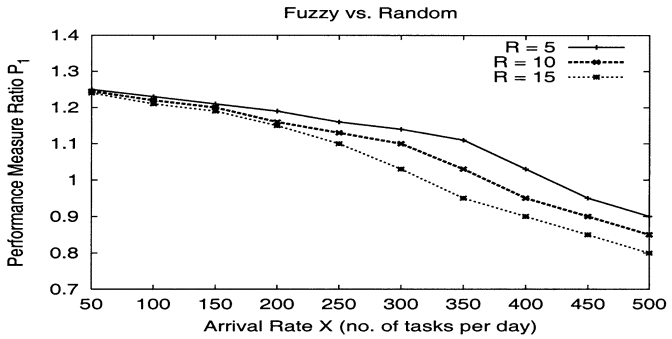

Fig. 7. Comparison of fuzzy and random task assignment methods.

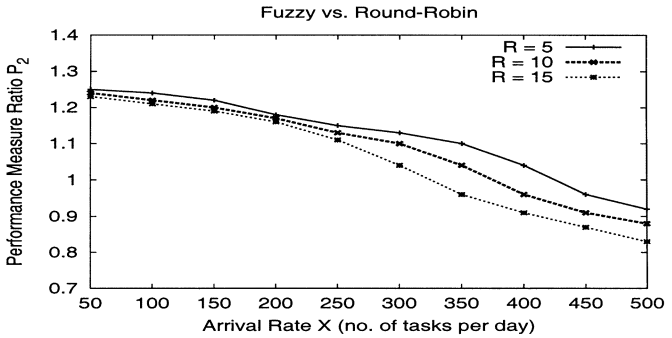

Fig. 8. Comparison of fuzzy and round-robin task assignment methods.

performance. For example, if the point is 1.2 then it means that the fuzzy performance is 20% better than the random or round-robin performance. If the point is 0.9 then it means that the fuzzy performance is 10% worse than the random or round-robin performance. From the observations the following conclusions can be drawn.

For a large range of the arrival rates (till 350 tasks/day for $R = 5$, till 300 for $R = 10$, and till 250 for $R = 15$), the fuzzy approach gives consistent high gains (between 25% and 10%). The is because of the fact that the random and round-robin algorithms assign tasks without consideration of the goodness of a node and the preference ratings of the tasks.

When the arrival rate is high (between 350 and 400 for $R = 5$, between 300 and 350 for $R = 10$, and between 250 and 300 for $R = 15$), the fuzzy technique has small gains (less than 10%). The reason is that the system has reached the saturation point. Each node is highly loaded in this case and the fuzzy performance is only slightly better.

For very high arrival rates (above 400 for $R = 5$, above 350 for $R = 10$, and above 300 for $R = 15$), the fuzzy technique performance is worse than others. At this point, the system is overloaded. The surplus values required by the fuzzy technique are updated at a much less rate than the very high task arrival rate. Hence it cannot make a good decision even after spending a longer time. Recall that the graphs in Section IV indicate that on overload, a large number of tasks miss their deadlines.

It is observed from the graphs that the performance with a lower R value is better than that with a higher R value. This can be explained as follows. In the computation of the fuzzy sets for the objectives, whenever a resource is not present at a node, we predict the target node for the subtask migration as the one which has the maximum amount of the surplus. This prediction could be erroneous for some of the tasks and hence the performance goes down. For lower R values, the probability of the absence of a resource at a node is lower, and hence the error would be introduced lesser number of times.

Based on the same argument as above, the *overload point*, i.e. the point from which the fuzzy technique performs worse than the random or the round-robin techniques also comes later, i.e. with higher arrival rates, for lower values of R. This gives an indication that for larger tasks requiring higher number of resources, the arrival rate of the system should be such that the system is not overloaded. The task mix and the complexity of tasks should be used to control the arrival rate.

VII. Conclusions

In this paper, the GPGP approach for multiagent systems is extended to support resource sharing across nodes in a distributed environment. A coordination mechanism based on the contract net protocol is proposed that would decide where a task should be migrated if the resources required for the accomplishment of the task are not available locally. Simulation studies have been performed to compare this mechanism with other commonly used techniques. It is observed that the mechanism performs better under all circumstances.

In the system under consideration, multiple criteria or objectives could be associated with a task and preference ratings may be assigned to each of the criteria. The preference ratings may be imprecise since they could be described in a subjective manner. The important problem of the assignment of a task to a particular node in the distributed system is addressed. A technique for the same using the fuzzy set approach is presented. It is felt that this approach is appropriate since there is imprecision in not only the specification of the preference ratings but also in the global system state collected for decision making because of the dynamic nature of the system. The effectiveness of the technique is demonstrated through simulation studies.

Acknowledgment

The authors would like to thank the anonymous reviewers whose insightful comments and suggestions have been incorporated in this revised manuscript.

References

[1] E. Durfee, "Distributed problem solving and planning," in *Multiagent Sytems: A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Ed. Cambridge, MA: MIT Press, 1999, ch. 3, pp. 121–164.

[2] U. Deshpande, A. Gupta, and A. Basu, "Collaboration in a Distributed Hospital Environment," Dept. Computer Science Engineering, IIT-Kharagpur, India, Tech. Rep. IITKGP/CSE/AB/2002/I, July 2002.

[3] e-University Task Force, "The University of Texas at Austin e-University Initiative,", [Online]. Available: http://www/utexas.edu/e-University/.

[4] K. Decker, "Environment Centered Analysis and Design of Coordination Mechanisms," Ph.D. dissertation, Univ. Massachusetts, Amherst, 1995.

[5] K. Decker and V. Lesser, "Designing a family of coordination algorithms," in *Proc. 1st Intl. Conf. Multi-Agent Systems* San Francisco, 1995, pp. 73–80.

[6] K. Decker and J. Li, "Coordinating mutually exclusive resources using GPGP," *Auton. Agents Multi-Agent Syst.*, vol. 3, no. 2, pp. 133–157, June 2000.

[7] R. Davis and R. Smith, "Negotiations as a metaphor for distributed problem solving," *Artif. Intell.*, vol. 20, no. 1, pp. 63–109, Jan. 1983.

[8] W. Walsh and M. Wellman, "A market protocol for decentralized task allocation," in *Proc. 3rd Int. Conf. Multi-Agent Syst.* Paris, France, 1998, pp. 325–332.

[9] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and W. Stornetta, "Spawn: A distributed computational economy," *IEEE Trans. Software Eng.*, vol. SE–18, pp. 103–117, 1982.

[10] K. Ramamritham, J. Stankovic, and W. Zhao, "Distributed scheduling of tasks with deadlines and resource requirements," *IEEE Trans. Comput.*, vol. 38, pp. 1110–1123, Aug. 1989.

[11] R. Bellman and L. Zadeh, "Decision making in a fuzzy environment," *Manage. Sci.*, vol. 17, pp. 141–165, 1970.

[12] R. Yager, "A new methodology for ordinal multiobjective decisions based on fuzzy sets," *Decision Sci.*, vol. 12, pp. 589–600, 1981.