Collaborative Assignment : A Multiagent Negotiation Approach Using BDI Concepts

Kiam Tian Seow
Centre for Decision Support
DSO National Laboratories
20 Science Park Drive
Republic of Singapore 118230
kiamtian-seow@dso.org.sg

Khee Yin How
Centre for Decision Support
DSO National Laboratories
20 Science Park Drive
Republic of Singapore 118230
hkheeyin@dso.org.sg

ABSTRACT

A multiagent collaboration algorithm using the concepts of belief (B), desire (D) and intention (I) for the classical assignment problem is presented. The problem can be viewed as seeking a concurrent allocation of one different resource for every task. Existing sequential algorithms use a single agent to operate on all assignment values; each value indicates the application quality-of-service (A-QoS) of one resource for one task. However, for many network applications, it may be practically more effective or desirable to deploy multiple communicating agents to solve the problem. The proposed algorithm for collaborative assignment is motivated by this potential and the availability of multiagent technologies to implement it.

Given an (N tasks and N resources) $N \times N$ assignment problem, the proposed algorithm has each different task represented by an agent possessing only (local) knowledge of A-QoS's of all resources for the task. The novelty of the algorithm lies in the generic BDI-based reasoning mechanism of task agents whose individual resource exchange intentions are arbitrated by a simple arbitration agent. The algorithm is examined both analytically and experimentally, and discussed, also in relation to existing multiagent work.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents, Multiagent Systems

General Terms

Algorithms, Design, Experimentation

Keywords

Agent Negotiation, Distributed Assignment Problem

1. INTRODUCTION

According to Nick R. Jennings [1], systems composing of multiple interacting problem solvers are becoming increasingly pervasive as a means of conceptualizing a diverse range of applications. This is being fuelled by rapid advances in communication and networking technologies, providing an ubiquitous networked environment in which it is becoming practically more effective or desirable to deploy multiple problem solvers as communicating agents for solving a variety of application problems. In this direction, this paper presents a distributed/parallel technique as a system of communicating agents for solving the classical assignment problem [2].

In the context of shared resource allocation, the goal of the assignment problem is to maximize the total A-QoS (application quality-of-service)[1] that a set of resources can offer when concurrently allocated to a set of tasks, under a basic constraint of allocating[2] one different resource for every task. In our opinion, being one of the simplest resource allocation problems, it serves as a useful test bed for new computational ideas, which can later be extended to more general resource allocation problems.

In the literature, a variety of exact algorithms for the *dual of the* classical assignment problem has been proposed, i.e., these algorithms are minimization techniques. They include, among others [3], the Munkres algorithm [2], the Hungarian method [4], the Jonker-Volgenant algorithm [5], the primal simplex algorithm [6] and the Ford-Fulkerson method [7]. What these existing sequential algorithms offer are centralized solutions which can only be implemented on a single agent operating on a complete table of negated A-QoS values[3]. Burkard and Cela [3] provides a good exposition of the current state-of-the-art on algorithms for the assignment problem and its extensions. In this paper, we refer to the sequential algorithms as Single Agent Assignment Algorithms (SA^3's). A distributed/parallel *agent* solution may offer a better alternative for the problem.

Distributed agent approaches use a network of processors to share the workload (i.e., number of tasks) in solving the assignment problem; hence these processors need not be more powerful. When the workload increases, new processors can simply be added to scale up the total processing capability. Single agent approaches running a centralized assignment algorithm, on the other hand, use a central processor to solve the problem completely. The central processor needs to scale up in power and speed when its workload increases, and this may not be practical or even feasible. Therefore, although some of the existing centralized algorithms are quite efficient, we believe that many network applications will potentially benefit more from a distributed/parallel agent approach to the problem.

Motivated by this potential and the availability of multiagent

AAMAS'02, July 15-19, 2002, Bologna, Italy.

[1] Just think of it as a benefit to be maximized.

[2] In this study, *allocating a resource to a task* is used interchangeably with *assigning a task to a resource.*

[3] As these are minimization algorithms, so operating on a table of *negated* A-QoS values will yield a maximal solution for our A-QoS maximization problem.

technologies [8] such as JADE[4] and FIPA-OS[5], among several others, to implement it, this paper proposes a new multiagent algorithm as a distributed/parallel technique for collaborative assignment. The proposed algorithm extends existing algorithms from centralized agent- to distributed/parallel agent-reasoning. It does so by distributing the tasks among the agents such that each different task is represented by a task agent possessing (local) knowledge of A-QoS's of all resources for the task only. Each task agent performs (Belief-Desire-Intention) BDI-based reasoning - a novel feature of the algorithm - and individually proposes resource exchange intentions which are arbitrated by a simple arbitration agent.

BDI architectures have their philosophical origins in the theory of human practical reasoning [9, Ch. 1] developed by M. Bratman [10] - that of deciding moment by moment which action to take in the furtherance of a goal or a set of goals. The basic architecture views the agent as having certain *mental attitudes* of belief, desire and intention representing, respectively, the information, motivation and decisive stance of the agent [11] in an arbitrary moment.

By embedding in a task agent - a distributed problem solver - a generic mechanism of BDI reasoning that is rooted in human practical reasoning, the proposed algorithm, in our opinion, provides a more natural way of solving the problem than any reasoning mechanism does in existing centralized algorithms. As the main body of this paper will detail, the concepts of BDI, used in a *metaphoric* sense, help to conceptualize the working mechanism of an arbitrary task agent - that of deciding collaboratively, round by round, which agent it should exchange its current resource selection with, in an attempt to achieve the agents' common goal of maximizing the total (allocated) A-QoS. It is appropriate to emphasize here that only the agent reasoning metaphor of BDI is applied, and our proposed algorithm does not specifically follow any BDI agent theory found in the literature (see references in [9, Ch. 1]). This should be clear in the algorithmic details presented in the main body of the paper.

Note that the multiagent approach has been described as distributed/parallel. By the former - *distributed* - we mean that each agent could run on a different workstation and communication would occur via a (high-speed) communication network. By the latter - *parallel* - we mean that each agent could run on a processor in a multiprocessor machine or more generally, on a thread in a multi-threading environment, and communication would occur via shared memory or message passing on a common bus.

Apparently related to our work are some efforts that *parallelize* the original sequential algorithms, including those referenced earlier, so that they become implementable on multiple processors to achieve computational speed-up (see [12] and the reports and references of [3, 13]). Our research motivation is, however, fundamentally different from these efforts', in that our aim is not to parallelize existing assignment algorithms, but to have a *negotiation* scheme that enables software agents to solve the assignment problem collaboratively. Such a scheme is needed in a network of autonomous platforms for distributed resource allocation.

Related and perhaps deserving special mention for applying a different metaphor of negotiation to the assignment problem is the auction algorithm proposed in the work of Bertsekas [14], as singled out from the extensive references in [3, 13]. We defer to Section *6.2.2* a discussion of this approach in relation to ours.

The rest of this paper is organized as follows : Section 2 reviews the classical assignment problem and proposes an agent approach to which it can be distributed/parallelized. Section 3 presents in detail a multiagent algorithm for collaborative assignment and illustrates its working mechanism with a simple example. Section 4 presents a solution analysis of the assignment problem via its reachability graph, on which the proposed algorithm traverses, and from which some important properties about the algorithm are derived. One drawback of the algorithm is that it does not guarantee an optimal solution. Section 5 presents and discusses some empirical evidence on the quality of the assignment solutions obtained using an experimental software prototype of the proposed algorithm. The quality is examined primarily in terms of the extent that the solutions produced deviate from the optimal one. Section 6 presents a discussion of the proposed algorithm and in relation to existing multiagent work. Section 7 summarizes the paper and points to some future work.

[4]From website http://sharon.cselt.it/projects/jade/

[5]From website http://fipa-os.sourceforge.net/

2. THE ASSIGNMENT PROBLEM

2.1 Problem Formulation

Let

$$T = \{t_0, t_1, \cdots, t_{|T|-1}\} \text{ and } R = \{r_0, r_1, \cdots, r_{|R|-1}\} \quad (1)$$

denote a set of tasks and resources respectively, and let

$$d_{ij} = d[t_i, r_j], \text{ for } t_i \in T, r_j \in R \quad (2)$$

be a measure of the A-QoS that a resource $r_j \in R$ can offer to a task $t_i \in T$ upon allocation.

Assume $|T| \leq |R|$. Then formally, the objective of the $|T| \times |R|$ assignment problem is to find the particular (total) assignment mapping

$$\Pi : T \mapsto R \qquad \text{such that for } t_i, t_j \in T, \\ i \neq j \text{ implies } \Pi(t_i) \neq \Pi(t_j) \quad (3)$$

and the total quality of service (total A-QoS)

$$S_{tot} = \sum_{i=0}^{|T|-1} d[t_i, \Pi(t_i)] \quad (4)$$

is maximized over all possible assignment (or allocation) sets induced by Π.

$\Pi(t) \in R$ is referred to as a resource selection by task $t \in T$ (under an arbitrary permutation of Π). Intuitively, Π (3) specifies that no two different tasks select the same resource, and every task in T selects only one resource in R. An assignment set corresponds to one permutation of Π (3); equivalently, we say that it contains *assignments* of the form $(t, \Pi(t)) \in T \times R$. The formulation follows closely the one given in [12], although the A-QoS maximization objective presented is *dual* of the cost minimization objective given therein.

Note that if $|T| = |R|$, then every resource in R is selected (by one task in T).

2.2 A Simple Example

Consider the following 3×3 assignment problem, with

$$T = \{t_0, t_1, t_2\} \text{ and } R = \{r_0, r_1, r_2\}.$$

The individual A-QoS values are tabulated in T-R Table 1.

Table 1: A 3×3 Assignment Problem

	r_0	r_1	r_2
t_0	{14}	5	8
t_1	2	6	{4}
t_2	8	{7}	3

For this problem, the optimal solution consists of the assignment set :

$$\{(t_0, r_0), (t_1, r_2), (t_2, r_1)\},$$

with a maximal total A-QoS of $14 + 4 + 7 = 25$.

Existing algorithms mentioned in the Introduction, such as the Jonker-Volgenant algorithm [5], can yield such optimal solutions. These are, however, minimization algorithms, so to solve our A-QoS maximization problem, they need to operate on a T-R table of *negated* A-QoS values.

The assignment problem is fundamental in a variety of resource allocation applications. For example, in a multiple sensor-multiple target tracking problem [15], set T represents a set of tracks and set R represents a set of sensors. In the example, the A-QoS value $d[t_i, r_j]$ represents the effectiveness of the resource $r_j \in R$ for task $t_i \in T$, and their goal is to find a set of allocations (i.e., a permutation of Π (3)) that maximizes the total A-QoS (4).

2.3 Distributing the Assignment Problem

The basic approach is to decompose the T-R table row-wise, such that each task agent represents (i.e., has the responsibility of selecting a resource $r \in R$ for) a task $t_i \in T$ and has A-QoS knowledge of task $t_i \in T$ only, namely, the individual A-QoS values $d[t_i, r]$ for all resources $r \in R$. For the example of Section 2.2, three task agents are needed; each agent represents a different $t_i \in T$ and only has A-QoS knowledge contained in the t_i-row of Table 1. A task agent representing task $t \in T$ is called a t-agent; for convenience, where it is clear in the context, we simply use the terms *task* $t \in T$ or *agent* $t \in T$ to refer to a *t-agent*.

The next section presents the main contribution of this paper that realizes the multiagent approach. It details an algorithmic but natural way these task agents reason and interact to solve the basic assignment problem. The work extends existing algorithms from centralized agent- to distributed/parallel agent-reasoning.

3. MULTIAGENT COLLABORATION

3.1 Overview of Approach

The multiagent algorithm proposed in this paper attempts to optimally solve an $N \times N$ assignment problem. In the algorithm, the task agents carry out *collaborative negotiations* among themselves in the presence of an arbitration agent. By negotiation, we mean that these task agents work out, interactively, an agreement that is acceptable by all agents, by way of individually selecting and re-selecting (more than once if necessary) a different resource through agent-proposed resource exchanges approved by the arbitration agent. The negotiation is collaborative because it involves agent reasoning and interaction in an attempt to achieve their common goal of maximizing the total (allocated) A-QoS (4). Collaborative negotiations allow task agents to work in parallel on the assignment problem. Details of the algorithm, namely, the generic reasoning mechanism of a task agent and the role of arbitration, are presented next.

3.2 Concepts of Belief, Desire and Intention

The essence of the algorithm lies in the task agent's reasoning mechanism which is based on the concepts of belief (B), desire (D) and intention (I).

To ground the BDI concepts in the problem context, the following domain-specific data structures are formally defined and interpreted as a task agent's beliefs, desires and intention computed in an arbitrary round of collaborative negotiation. In these definitions, the current resource selections of all agents refer to those made under an arbitrary permutation of Π (3).

DEFINITION 1 (BELIEF SET $\mathsf{B_i}$). *Given that an agent $t_i \in T$'s current resource selection is $r^i \in R$. Then its (current) belief set $\mathsf{B_i}$ is given by*

$$\mathsf{B_i} = \{r \in R \mid d[t_i, r] > d[t_i, r^i]\} \tag{5}$$

Intuitively, if $\mathsf{B_i} \neq \emptyset$, this means that agent $t_i \in T$ has at least one alternative resource selection $r \in \mathsf{B_i}$ that may lead to increase in total A-QoS (4) (when made in exchange with an agent whose current selection is $r \in R$).

DEFINITION 2 (DESIRE SET $\mathsf{D_i}$). *Given that an agent $t_i \in T$'s current resource selection is $r^i \in R$ and its belief set is $\mathsf{B_i}$, $\mathsf{B_i} \neq \emptyset$. An arbitrary agent $t_j \in T$ whose current resource selection is $r^j \in R$ is said to accept agent $t_i \in T$'s beliefs $\mathsf{B_i}$ if $r^j \in \mathsf{B_i}$. To generate the desired exchange options or desires $\mathsf{D_i}$, agent $t_i \in T$ broadcasts its beliefs $\mathsf{B_i}$ and current selection $r^i \in R$, and an arbitrary agent $t_j \in T$ who accepts the beliefs would respond with a pair of A-QoS values $d[t_j, r^j]$ and $d[t_j, r^i]$, so that for each of the $|\mathsf{B_i}|$ responses received, the corresponding resource exchange option $[(t_i, r^j), (t_j, r^i), \rho] \in \mathsf{D_i}$ (i.e., is agent $t_i \in T$'s desire) if $\rho > 0$, where ρ is defined by*

$$\rho = -d[t_i, r^i] + d[t_i, r^j] - d[t_j, r^j] + d[t_j, r^i] \tag{6}$$

If $\rho > 0$, it means that there is a net gain if agent $t_i \in T$ gives up its current selection $r^i \in R$ and selects $r^j \in R$, and in exchange, agent $t_j \in T$ gives up its current selection $r^j \in R$ and selects $r^i \in R$. Thus, any desire $d \in \mathsf{D_i}$, when carried out, will definitely lead to an increase in total A-QoS without violating Π (3). Quite naturally, it provides the motivation for agent $t_i \in T$ to want to exchange its current resource selection.

DEFINITION 3 (INTENTION I_i). *Given that an agent $t_i \in T$'s desire set is $\mathsf{D_i}$, $\mathsf{D_i} \neq \emptyset$. Then, agent $t_i \in T$'s intention I_i is given by*

$$\begin{aligned} I_i &= [(t_i, r^j), (t_j, r^i), \rho] \in \mathsf{D_i}, \text{ for which} \\ \rho &= max\{\rho' \mid [-, -, \rho'] \in \mathsf{D_i}\} \end{aligned} \tag{7}$$

Agent $t_i \in T$'s decisive stance or intention has to be I_i since it is the best exchange option that the agent can propose. It is said to have no intention if either $\mathsf{B_i} = \emptyset$ or $\mathsf{D_i} = \emptyset$.

Finally, in the role of arbitration, an intention that contributes to the highest increase (in total A-QoS) if carried out is selected from all the agents' intentions $I_i \in \mathsf{I}$ gathered. In the following algorithm, this simple role of arbitration is handled by a dedicated agent.

3.3 Distributed Agent Algorithm

The proposed algorithm MA^3 assumes that $|T| = |R| = N$, and consists of an arbitration agent (or arbiter) and a team of t-agents, $t \in T$. Agent $t \in T$ only has A-QoS knowledge of the task it represents, i.e., $d[t, r]$ for all $r \in R$. Each task agent initially selects a resource $r \in R$ according to (a permutation of) $\Pi : T \mapsto R$ (3). The arbiter then initiates the negotiation process which is described in the next section.

3.3.1 Algorithmic Details

The generic BDI reasoning mechanism of a task agent and the simple role of the arbitration agent in an arbitrary round of collaborative negotiation can now be described as follows :

Algorithm MA^3 : Collaborative (Task) Agent

---

1. If agent believes that there are alternative resource selections which may lead to increase in total A-QoS, it would, based on its (local) beliefs, generate the desired exchange options or desires, from which the best option will be chosen as its intention.
2. Agent submits its intention (or the lack thereof) to the arbitration agent.
3. Concurrent with Step 1 and Step 2, it responds to any requesting task agent whose beliefs it accepts, by sending to the requesting agent the A-QoS values as required for computing the requesting agent's desire.
4. Agent changes its resource selection (and then acknowledges it), proceeds to next round of negotiation or quit, as decided by the arbitration agent.

Algorithm MA^3 : Arbitration Agent

---

1. Agent first receives the intentions (or the lack thereof) of all the task agents.

2. If agent sees that all task agents have no intention to exchange, it terminates the negotiation by telling all task agents to quit.
3. Otherwise, it
   (a) selects an intention with the highest increase (in total A-QoS) and instructs the two agents concerned to proceed with the resource exchange.
   (b) receives acknowledgement of resource exchange made as instructed (from the two agents concerned), before telling all task agents to proceed to next round of negotiation.

3.3.2 An Example

To illustrate the working mechanisms of the proposed algorithm MA^3, consider the earlier example problem presented in Section 2.2. Figure 1 shows two assignment tables for the problem, in which the resource selection of each agent $t_i \in \{t_0, t_1, t_2\}$ is represented by enclosing the corresponding A-QoS value within curly brackets {}.

	r_0	r_1	r_2
t_0	{14}	5	8
t_1	2	{ 6}	4
t_2	8	7	{3}

Table (a)

$\xrightarrow{\text{MA}^3}$

	r_0	r_1	r_2
t_0	{14}	5	8
t_1	2	6	{4}
t_2	8	{7}	3

Table (b)

Figure 1: Example to illustrate Algorithm MA^3

Referring to Figure 1, Table (a) represents a randomly selected (initial) assignment set and Table (b) represents a solution assignment set. The following illustrates how the solution can be obtained by collaborative negotiations.

- Round 1
  - Agent t_0 selects r_0 and agent t_1 selects r_1 (as initialized). Both agents believe that they have the best selection because their belief sets are empty, hence no desire, and therefore no intention.
  - Agent t_2 selects r_2 (as initialized) but believes that there are alternative resource selections that may increase the total A-QoS, namely resources r_0 and r_1. It therefore generates its desired exchange options as follows :
    * for exchange with agent t_0, the exchange gain is $-d[t_2,r_2] + d[t_2,r_0] - d[t_0,r_0] + d[t_0,r_2]$ which is equal to $-3+8-14+8 = -1 \leq 0$.
    * for exchange with agent t_1, the exchange gain is $-d[t_2,r_2] + d[t_2,r_1] - d[t_1,r_1] + d[t_1,r_2]$ which is equal to $-3+7-6+4 = 2 > 0$.
    * Hence, its only desire is $[(t_2,r_1),(t_1,r_2),2]$ and is therefore also its intention.
  - To get the required pairs of A-QoS values $\{d[t_0,r_0], d[t_0,r_2]\}$ and $\{d[t_1,r_1], d[t_1,r_2]\}$ for computing its desire set as done above, agent t_2 broadcasts its belief set $\{r_0, r_1\}$ and current selection $r_2 \in R$. The respective agents whose current resource selection is in agent t_2's belief set respond with those values. In subsequent rounds, such broadcasts and responses are deemed understood and will not be mentioned again.
  - Agents t_0, t_1 and t_2 send their intentions (or the lack thereof) to the arbitration agent.
  - The arbitration agent tells agent t_1 to change its resource selection to r_2 and agent t_2 to change it to r_1.
  - Once both agents t_1 and t_2 inform the arbitration agent that they have changed the selections as instructed, the arbitration agent tells all agents to proceed to next round of negotiation.
- Round 2
  - Agent t_0 selects r_0 and believes that it has the best selection because its belief set is empty, hence no desire, and therefore no intention.
  - Agent t_1 selects r_2 but believes that an alternative resource selection r_1 may increase the total A-QoS. It therefore generates its desired exchange options as follows :
    * for exchange with agent t_2, the exchange gain is $-d[t_1,r_2] + d[t_1,r_1] - d[t_2,r_1] + d[t_2,r_2]$ which is equal to $-4+6-7+3 = -2 \leq 0$.
    * Hence, it has no desire, and therefore no intention.
  - Agent t_2 selects r_1 but believes that an alternative resource selection r_0 may increase the total A-QoS. It therefore generates its desired exchange options as follows :
    * for exchange with agent t_0, the exchange gain is $-d[t_2,r_1] + d[t_2,r_0] - d[t_0,r_0] + d[t_0,r_1]$ which is equal to $-7+8-14+5 = -8 \leq 0$.
    * Hence, it has no desire, and therefore no intention.
  - Agents t_0, t_1 and t_2 send their lack of intentions to the arbitration agent.
  - The arbitration agent tells all agents t_0, t_1 and t_2 to quit. The final resource selections of the agents yield the solution as shown in Table (b) of Figure 1.

4. ANALYSIS AND IMPLEMENTATION

In this section, we present an analysis of the assignment problem via a formulation of an assignment reachability graph for the problem. Using the properties of the reachability graph, basic properties about the proposed algorithm MA^3 are formalized. To conclude this section, a brief note on an implementation of the proposed algorithm is given.

4.1 The Assignment Reachability Graph

Given an assignment problem, we model the possible sequential execution of desires in a reachability graph as follows :

For a set of tasks T and a set of resources R, for which $|T| = |R| = N \geq 2$, let

$$\mathcal{G} \stackrel{\text{def}}{=} (V, D, \delta, V_o) \tag{8}$$

represent an assignment reachability graph (ARG) in which :

1. V denotes a (nonempty) finite set of states uniquely characterizing the permutations of Π (3), and we write $\Pi(t)|_v$ to denote the resource selection of task $t \in T$ in state $v \in V$. $|V| = N!$. The total A-QoS (4) in a state $v \in V$ (i.e., a permutation of Π) is denoted by $|v|$ and given by
$$|v| = \sum_{i=0}^{(N-1)} d[t_i, \Pi(t_i)|_v].$$
2. $D \subseteq V \times V$ denotes a finite set of desires.
3. $\delta : D \times V \mapsto V$ is a state transition function (due to resource exchange between two arbitrary tasks $t_i, t_j \in T$), such that $\delta(e_{ij}, v) = v' \in V$ iff $\Pi(t_i)|_{v'} = \Pi(t_j)|_v$ and $\Pi(t_j)|_{v'} = \Pi(t_i)|_v$ and the magnitude of $e_{ij} \in D$, $\triangle e_{ij}|_v > 0$, is defined by :
$$\begin{aligned} \triangle e_{ij}|_v &= \{-d[t_i, \Pi(t_i)|_v] + d[t_i, \Pi(t_j)|_v]\} + \\ &\quad \{-d[t_j, \Pi(t_j)|_v] + d[t_j, \Pi(t_i)|_v]\} > 0. \end{aligned}$$
We can interpret $\triangle e_{ij}|_v$ as the increase in total A-QoS if task t_i and task t_j exchange their resource selections held in state $v \in V$, i.e., $\Pi(t_i)|_v$ and $\Pi(t_j)|_v$, respectively.

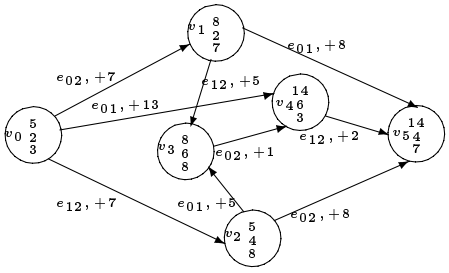

Figure 2: Assignment Reachability Graph : An Example

4. $V_o \subseteq V$ denotes a finite set of terminal states such that for $v_o \in V_o$, $\delta(e, v_o)$ is not defined for any $e \in D$.

Let D^* contain all possible finite sequences, or strings, over D, plus the null string ε. Then, definition of δ can be extended to D^* as follows :

$$\delta(\varepsilon, v) = v,$$

$$(\forall e \in D)(\forall w \in D^*), \delta(we, v) = \delta(e, \delta(w, v)).$$

For a string $s \in D^*$, $|s|$ denotes the length of the string, i.e., the number of elements of set D in string s. $|s| = 0$ if $s = \varepsilon$.

We conclude this section with the following definition.

DEFINITION 4 (MAX-TRANSITION). *At an arbitrary state $v \in V$, a max-transition is a desire $e_{max} \in D$ for which :*

$$\triangle e_{max}|_v = max\{\triangle e|_v \mid \delta(e, v) \in V\}. \quad (9)$$

A max-transition $e_{max} \in D$ is unique if $\triangle e_{max}|_v > \triangle e|_v$ for all $e \in D$, $\delta(e, v) \in V$ for which $e \neq e_{max}$.

In effect, a max-transition represents the *best* intention that the arbitration agent selects to conclude a round of negotiation.

4.1.1 An Example

Note that an ARG $\mathcal{G}$ can be naturally described by a directed-transition graph. Figure 2 shows an ARG for the simple example presented in Section 2.2.

To interpret Figure 2 correctly, it is necessary to take note of the following : For this example, as seen in T-R Table 1, there are no equal A-QoS values in each row. Thus, for simplicity, in each state $v_i \in V$, we list only the A-QoS values such that the top most value is due to a resource selection by agent t_0, the next is due to that by agent t_1 and so on. In a state $v_i \in V$, one can easily determine from T-R Table 1 which resource has been selected by which agent. For instance, in state $v_0 \in V$, the top most value listed is 5 which is due to the resource selection by task t_0, thus we know that task t_0 selects resource r_1 in state $v_0 \in V$, since $d[t_0, r_1] = 5$ as given in Table 1. Notationally, for $\delta(e_{ij}, v) = v'$, the information associated with e_{ij} at state $v \in V$ is represented as : e_{ij}, $\triangle e_{ij}|_v$.

In the illustration of the proposed algorithm MA^3 in Section 3.3.2 that uses the same example problem, we note that the negotiation starts from state $v_4 \in V$ of the ARG in Figure 2, since it is an equivalent (and unique) representation of Table (a) in Figure 1. In the illustration, after round 1, the arbitration agent approves the only and best intention proposed, as represented by transition e_{12} that leads state $v_4 \in V$ to state $v_5 \in V_o$ representing Table (b) of Figure 1. But although these agents have reached a terminal state of the ARG, the task agents only know about this in another round (i.e., round 2) of negotiation, when the arbitration agent receives the lack of intentions by all agents and informs them to terminate negotiation.

4.1.2 Properties of ARG

Below, we establish some basic properties of an ARG $\mathcal{G}$ (8).

PROPERTY 1. *If $e \in D$ and $\delta(e, v_1) = v_2 \in V$, then $|v_2| > |v_1|$.*

Proof : If $\delta(e, v_1) = v_2$, and arbitrarily $e = e_{ij} \in D$, then

$$|v_1| = \sum_{all\ k \in \{1,\cdots N\} - \{i,j\}} (d[t_k, \Pi(t_k)|_{v_1}] + \{d[t_i, \Pi(t_i)|_{v_1}] + d[t_j, \Pi(t_j)|_{v_1}]\}) \quad (10)$$

and

$$|v_2| = \sum_{all\ k \in \{1,\cdots N\} - \{i,j\}} (d[t_k, \Pi(t_k)|_{v_2}] + \{d[t_i, \Pi(t_i)|_{v_2}] + d[t_j, \Pi(t_j)|_{v_2}]\}) \quad (11)$$

But there is no change in resource selections among tasks in $T - \{t_i, t_j\}$ under transition $e_{ij} \in D$, thus we can let

$$c = \sum_{all\ k \in \{1,\cdots N\} - \{i,j\}} d[t_k, \Pi(t_k)|_{v_1}] = \sum_{all\ k \in \{1,\cdots N\} - \{i,j\}} d[t_k, \Pi(t_k)|_{v_2}]. \quad (12)$$

Therefore, we have :

$$\begin{aligned}
|v_2| &= c + \{d[t_i, \Pi(t_i)|_{v_2}] + d[t_j, \Pi(t_j)|_{v_2}]\} \\
&\quad \text{By combining Eqn (11) and Eqn (12)} \\
&= c + \{d[t_i, \Pi(t_j)|_{v_1}] + d[t_j, \Pi(t_i)|_{v_1}]\} \\
&\quad \text{Because } \Pi(t_i)|_{v_2} = \Pi(t_j)|_{v_1} \text{ and } \Pi(t_j)|_{v_2} = \\
&\quad \Pi(t_i)|_{v_1} \text{ by definition of } \delta(e_{ij}, v_1) = v_2 \\
&= c + \{d[t_i, \Pi(t_i)|_{v_1}] + d[t_j, \Pi(t_j)|_{v_1}]\} + \\
&\quad \{-d[t_i, \Pi(t_i)|_{v_1}] + d[t_i, \Pi(t_j)|_{v_1}]\} + \\
&\quad \{-d[t_j, \Pi(t_j)|_{v_1}] + d[t_j, \Pi(t_i)|_{v_1}]\} \\
&= |v_1| + \{-d[t_i, \Pi(t_i)|_{v_1}] + d[t_i, \Pi(t_j)|_{v_1}]\} + \\
&\quad \{-d[t_j, \Pi(t_j)|_{v_1}] + d[t_j, \Pi(t_i)|_{v_1}]\} \\
&\quad \text{By combining Eqn (10) and Eqn (12)} \\
&= |v_1| + \triangle e|_{v_1} \\
&\quad \text{By definition of } e \in D \\
&> |v_1| \\
&\quad \text{Because } \triangle e|_{v_1} > 0 \text{ by definition.}
\end{aligned}$$

Hence the property. ■

PROPERTY 2. *ARG $\mathcal{G}$ is acyclic.*

Proof : If $\delta(w, v) = v'$ for $w \in D^* - \{\varepsilon\}$, then by applying Property 1 recursively over $w \in D^*$, $|v'| > |v|$. Clearly, there is no string $x \in D^*$ such that $\delta(x, v') = v$, for it would contradict the fact that $|v'| > |v|$. Hence the property. ■

PROPERTY 3. *$V_o \neq \emptyset$ (i.e., given an arbitrary $v \in V$, $\exists w \in D^* : \delta(w, v) \in V_o$).*

Proof : The property follows from Property 2 and the finiteness of $|V|$. ■

4.2 Properties of Algorithm

We now present some basic properties of Algorithm MA^3.

THEOREM 1. *Algorithm MA^3 always terminates in a finite number of negotiation rounds.*

Proof : Starting from an arbitrary state $v \in V$ of an ARG $\mathcal{G}$ that is not a terminal state, Algorithm MA^3 traverses one max-transition $e \in D$ (as defined by Definition 4) in one round of negotiation such that by Property 3, it will eventually enter a terminal state $v_o \in V_o$. In a terminal state, a final round of negotiation proceeds during which the arbitration agent receives the lack of intentions by all agents and informs them to terminate negotiation. Hence the result. ■

(a) A Simple 3×3 Assignment Problem

	r_0	r_1	r_2
t_0	9	8	7
t_1	1	3	4
t_2	6	3	5

(b) Problem's ARG

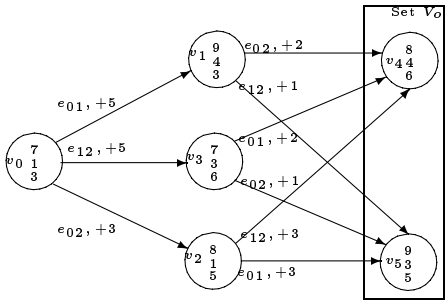

Figure 3: An Example That Shows Algorithm MA^3's No Guarantee of Optimality of Solution

THEOREM 2. *Given a random initial assignment set, algorithm* MA^3 *does not guarantee an optimal solution.*

Proof : This can be established by an example. Consider the example problem in Figure 3 (a). As shown in Figure 3 (b), this problem's ARG indicates that $V_o = \{v_4, v_5\}$, with $|v_4| = 18$ (optimal value) and $|v_5| = 17$. This means that if algorithm MA^3 should start at state $v_5 \in V$, it will end in the same state because it is a terminal state, but $v_5 \in V_o$ does not represent an optimal solution because $|v_5| < |v_4|$. Hence the result. ■

4.3 Implementation Using JADE

An experimental version of the proposed multiagent algorithm MA^3 for collaborative assignment has been implemented in a networked environment. The implementation is in Java™ 2 SDK[6] (Standard Edition, Version 1.2.2_006) and uses the JADE Agent Platform[7] (Version 2.1). JADE is an experimental JAva DEvelopment tool which is an open source, FIPA specification-compliant framework that simplifies the development of multiagent applications [16].

5. EXPERIMENT

In this section, we present an empirical study of the proposed algorithm to assess the quality of its assignment solutions. The quality is examined primarily in terms of the extent that the solutions produced deviate from the optimal one.

5.1 Results

To conduct the study, we ran the experimental prototype of Algorithm MA^3 for a number of randomly generated 10×10 test problems. An implementation[8] of the Jonker-Volgenant algorithm [5] was used to produce an optimal total value as a reference solution. For each test problem, extensive runs of the prototype were carried out based on different initial assignment sets. The findings for some of the test problems (given in Appendix A for reference) are tabulated in Table 2.

Table 2: Experimental Results For Randomly Generated 10×10 Test Problems

Problem No.	SA^3 [5] Optimal α_{opt}	Proposed MA^3 Random-I (Worst-Case) α_{wc}	ρ (%)	n_{max}
10-1	179.6	166.2	7.5	10
10-2	2086.0	2032.0	2.6	10
10-3	5160.0	5160.0	0.0	11
10-4	194.2	177.0	8.9	12
10-5	225.3	216.9	3.7	11
10-6	380.0	346.0	8.9	12
10-7	152.2	142.4	6.4	12

Legend :
10-X : label of a 10×10 assignment problem given in Appendix A
n : number of negotiation rounds
n_{max} : maximum number of negotiation rounds
α_{opt} : optimal Total A-QoS value
α_{wc} : total A-QoS value
ρ : error, given by $|\frac{\alpha_{wc} - \alpha_{opt}}{\alpha_{opt}}| \times 100\%$

From the table, we see that for these test problems, the worst-case solutions produced were good enough in the sense that they were either equal to or within 10% of the optimal, as verified against the solutions produced by an existing centralized algorithm [5]. Besides, we notice that the ratio of the maximum number of negotiation rounds n_{max} (required to produce a solution) and the problem size N approximated to 1 (to the nearest integer).

6. DISCUSSION

6.1 Multiagent Assignment Algorithm

6.1.1 Role of Arbitration & Total Decentralization

In the proposed algorithm MA^3, for conceptual clarity, we have assigned the simple role of arbitration to a dedicated agent. In practice, a task agent can actually be appointed to handle this role. In fact, it is also possible to endow every task agent with this role, but with slight (code) modifications so that the task agent broadcasts its intention and changes its resource selection accordingly if the best intention as arbitrated suggests it, instead of originally sending its intention to, and waiting to receive an exchange decision from the arbitration agent. While the latter variant of the algorithm brings about a *totally decentralized* algorithm in some sense, the significant communication overheads due to massive broadcasting of agents' intentions could pose a serious performance issue for real-time applications.

6.1.2 Online Applications

In a network, an agent using a centralized algorithm for assignment must first gather all A-QoS data from the task agents concerned. But in many real-world applications of the assignment problem, one should expect that the online A-QoS values may not be constant. Whenever some A-QoS values significantly change, an existing centralized algorithm would have to respond, recomputing a valid solution from scratch by gathering and regathering the A-QoS data, thereby possibly incurring high communication and processing overheads. For the proposed algorithm MA^3, no such centralized gathering of data is needed; the BDI-based reasoning mechanism divides the processing into negotiation rounds, and in each round, a task agent locally accesses and directly acts on its *row* of A-QoS data. Because of this, it is even possible

[6] From website http://www.java.sun.com
[7] From website http://sharon.cselt.it/projects/jade/
[8] From website http://www.magiclogic.com/assignment.html

to trivially modify a task agent to respond *on-the-fly* (i.e., during the collaboration process) by independently updating its local A-QoS data before the start of each negotiation round.

By Property 1, we know that after every round of negotiation that leads to a resource exchange, the total A-QoS value will increase. Therefore, in another variant of algorithm MA^3, the algorithm can terminate negotiation within a specified number of negotiation rounds. This variant requires simply modifying only the condition for termination in the role of arbitration, which clearly does not violate Property 1. Although this variant can possibly terminate before it reaches a terminal state, it attempts to produce the best possible solution within the specified number of rounds. Such a variant should prove useful in time critical applications.

6.2 Related Multiagent Work

6.2.1 Distributed Constraint Satisfaction Algorithm

The work of M. Yokoo et al [17] presents a general distributed constraint satisfaction algorithm DCS in which the qualitative constraints to be satisfied are distributed among the automated agents. In the context of shared resource allocation, each agent can be defined to handle a variable that represents a task in a shared-resource environment, where the domain (i.e., possible values) of the task variables are the resources to be shared. The qualitative constraints specify how the resources are to be allocated among the tasks. The goal of algorithm DCS is to find a constraint-conforming combination of variable values that enables all tasks to be executed concurrently.

To see how our work relates to M.Yokoo et al [17]'s, consider a reformulation of the assignment problem as a simple constraint satisfaction problem with optimization as follows.

Redefine set $T = \{t_0, t_1, \cdots, t_{|T|-1}\}$ as a set of task variables whose domain is given by set $R = \{r_0, r_1, \cdots, r_{|R|-1}\}$[9]. Then

$$maximize \sum_{i=0}^{|T|-1} d[t_i, \Pi(t_i)] \quad (13)$$

subject to the qualitative constraints that

$$\text{for all } i, k \in \{0, 1, 2, \cdots |T| - 1\},\ i \neq k,\ t_i \neq t_k. \quad (14)$$

In the example of Section 2.2 with $T = \{t_0, t_1, t_2\}$, $t_0 \neq t_1$, $t_1 \neq t_2$ and $t_2 \neq t_0$ constitute constraints (14).

Starting from an assignment set that satisfies constraints (14), the proposed algorithm MA^3 enables a series of 'A-QoS incrementing' resource exchanges that preserve these constraints. In contrast, given the same problem, algorithm DCS will find an assignment set, out of many possible sets, that conforms to constraints (14) only, with no consideration for optimization (13). The extension of distributed constraint satisfaction problem (DCSP) to distributed constraint satisfaction optimization problem (DCSOP) is an important research direction as optimal solutions to the latter - or at least those verifiably good ones - are often preferred. Intuitively speaking, one conceptually neat way to address a DCSOP is to find[10] an initial solution satisfying the given constraints, followed by a series of constraint-preserving collaborative negotiations that can lead to an optimal solution, or at least one that is good enough in attending to the tasks. The proposed algorithm MA^3 for the classical assignment problem - a specialized DCSOP - may be viewed as a small step in this direction.

6.2.2 The Auction Algorithm

Deserving special mention for applying a different metaphor of negotiation to the assignment problem is the auction algorithm that originated in the work of Bertsekas [14]. The spirit of negotiation via auctioning is evident by the algorithm doing the following in each iteration.

- Bid on behalf of the persons (or tasks in our context) for objects (or resources in our context). The bid of each person is the *object with the highest net value*. The net value is the magnitude of the difference between the benefit (or A-QoS value in our context) of assigning the object to this person and the object's latest price.
- Assign each object to a *person with the highest bid for it*, after unassigning the object from a different person (assigned with it at the start of the iteration), and adjust its price accordingly.

The algorithm terminates once every object has been bidded for at least once; for actual details, please refer to any of the references [18, 14].

There have been several multiprocessor- and shared memory-based implementations [3] of the algorithm to exploit the inherently parallelizable phases of bidding and assigning. But, to the best of our knowledge, there is no reported implementation of the algorithm as a system of software agents performing bidding and assigning to solve an $N \times N$ assignment problem. A multiagent implementation, if it exists, is at best a system of N task agents negotiating with N resource agents, possibly with a simple agent keeping track of the bid receipt of each resource agent and informing all agents when the termination condition holds. This contrasts with our proposed algorithm of N task agents negotiating among themselves for N resources in the presence of an arbitration agent.

7. CONCLUSION

This paper has proposed a multiagent algorithm MA^3 to address the classical $N \times N$ assignment problem. To the best of our knowledge, it is perhaps the first effort to develop a distributed/parallel problem solving technique using the agent reasoning metaphor of Belief-Desire-Intention for the problem.

This research should be of theoretical interest to multiagent researchers since the assignment problem is shown to be a specialized DCSOP, and the solution algorithm MA^3 represents a simple but interesting application of the concepts of belief (B), desire (D) and intention (I). In the proposed multiagent algorithm MA^3, these concepts have been metaphorically embedded in a task agent in a way that mimics human practical reasoning in some restricted sense. The research should also be of practical interest to application researchers as the original algorithm MA^3 proposed extends existing algorithms from centralized agent- to distributed/parallel agent-reasoning. Being agent-oriented, it can be easily modified to address several application concerns as discussed in this paper.

The apparent limitation is that it may not guarantee an optimal solution. But empirically, we have shown that in randomly selecting an initial assignment set, the worst-case solution is good enough. Also, the ratio of the maximum number of negotiation rounds required to produce a solution and the problem size N often approximates to 1 (to the nearest integer), providing a coarse indicator of the worst-case complexity bound of the proposed algorithm.

It is apparent that the proposed algorithm MA^3 may not guarantee an optimal solution unless the initial assignment set is *properly selected*. Notice that in the example of Figure 3, if the algorithm starts from any other state $v \in V - \{v_5\}$, it will end in state $v_4 \in V_o$ which represents the optimal solution. In general, whether an optimal solution can be obtained or not depends on the selection of the initial state (i.e., the initial assignment set). In future work, heuristics will be developed to select an initial assignment set, on which algorithm MA^3 can produce an optimal solution (or at least one that is *good enough*) in not more than $0.5N$ rounds of negotiation. Extensive experimentation will be carried out further to trap any significant deviation from the encouraging empirical evidence obtained thus far. A detailed study on the time complexity of the algorithm is also needed to obtain insights into the kind of distributed/parallel computer and communication architecture on which the algorithm should be implemented to solve the assignment problem most efficiently.

[9] That the resource set R is the domain of task set T means that every task variable $t \in T$ is assigned one domain value $r \in R$.

[10] Note that for the classical assignment problem, randomly finding an initial assignment set that satisfies constraints (14) is quite trivial, and hence assumed given for algorithm MA^3.

8. REFERENCES

[1] N. R. Jennings, "Specification and implementation of a Belief-Desire-Joint-Intention architecture for collaborative problem solving," *International Journal Of Intelligent and Cooperative Information Systems*, vol. 2, no. 3, pp. 289–318, 1993.

[2] F. Burgeios and J. C. Lassalle, "An extension of the Munkres algorithm for the assignment problem to rectangular matrices," *Communications of the ACM*, vol. 14, no. -, pp. 802–806, 1971.

[3] R. E. Burkard and E. Cela, "Linear assignment problems and extensions," *Handbook of Combinatorial Optimization, Vol.4* (P. M. Pardalos and D. Z. Du, eds.), pp. 75–149, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[4] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. -, pp. 83–97, 1955.

[5] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. -, pp. 325–340, 1987.

[6] M. Akgul, "A genuinely polynomial primal simplex algorithm for the assignment problem," *Discrete Applied Mathematics*, vol. 45, no. –, pp. 93–115, 1993.

[7] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, Princeton, N.J, 1962.

[8] B. Burg, J. Dale, and S. Willmott, "Open standards and open source for agent-based systems," *AgentLink News*, no. 6, pp. 2–5, January 2001. Available at website http://www.agentlink.org/newsletter/6/.

[9] G. Weiss, ed., *Multiagent System : A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, London, U.K, 1999.

[10] M. E. Bratman, ed., *Intentions, Plans and Practical Reason*. Harvard University Press, Cambridge, M.A, 1987.

[11] A. S. Rao and M. P. Georgeff, "BDI agents : From theory to practice," *Proceedings of The First International Conference on MultiAgent Systems*, (San Francisco, CA, U.S.A), pp. 312–319, - 1995.

[12] T. D. Gottschalk, "Concurrent implementation of Munkres algorithm," *Proceedings of the Fifth IEEE International Conference on Distributed Memory Computing*, pp. 52–57, 1990.

[13] S. Støroy and T. Sørevik, "Massively parallel augmenting path algorithms for the assignment problem," *Computing*, vol. 59, no. 1, pp. 1–16, 1997.

[14] D. P. Bertsekas, "The auction algorithm : A distributed relaxation method for the assignment problem," *Annals of Operations Research*, vol. 14, pp. 105–123, 1988.

[15] S. S. Blackman, *Multiple-Target Tracking with Radar Applications*, ch. 14 : Special Topics. ARTECH HOUSE, INC., Dedham, MA, 1986.

[16] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with JADE," *Pre-Proceedings of The Seventh International Workshop on Agent Theories, Architectures and Languages*, (Boston, MA, U.S.A), pp. 85–99, July 2000.

[17] 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*, vol. 10, no. 5, pp. 673–685, September/October 1998.

[18] D. P. Bertsekas, *Linear Network Optimization : Algorithms and Codes*. MIT Press, Cambridge, MA, USA, 1991.

APPENDIX

A. TEST PROBLEMS

Tabulated below are some randomly generated $N \times N$ test problems, where Ti$\in T$, Rj$\in R$, $i, j \in \{1, 2, \cdots, N\}$ and $N = 10$. There are two additional columns 'WC' and 'Nmax' which contains the initial assignment set for algorithm MA^3 that leads to α_{wc} and n_{max} (see Table 2), respectively. Under column 'WC' or 'Nmax', each Rj is a selection of Ti $\in T$ at row i, according to Rj $= \Pi$(Ti)$\in R$. Where either such column is absent, we mean that an arbitrary initial assignment set for algorithm MA^3 will yield the corresponding α_{wc} or n_{max}.

Problem 10-1

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	Nmax	WC
T0	10.5	6.4	5.1	4.0	9.1	12.6	14.0	10.0	8.8	17.0	R4	R6
T1	7.2	9.9	16.8	8.8	10.1	20.3	15.8	6.1	6.0	5.9	R8	R7
T2	17.5	6.8	5.1	14.0	8.1	19.6	14.0	15.0	18.8	18.0	R7	R8
T3	7.3	5.5	7.4	4.3	19.0	9.6	12.1	13.8	10.2	24.4	R2	R9
T4	2.7	3.1	7.4	17.2	1.6	16.6	21.1	14.7	19.9	7.8	R6	R0
T5	7.7	6.1	5.4	7.2	7.6	14.6	11.1	18.7	19.0	8.8	R5	R1
T6	6.6	7.8	7.2	5.1	4.9	5.2	9.2	6.6	7.6	10.6	R3	R2
T7	5.3	2.5	1.4	8.3	9.0	6.6	2.7	23.8	11.2	14.4	R1	R3
T8	8.6	14.8	7.2	9.1	2.9	5.2	19.2	16.6	17.6	20.6	R9	R4
T9	2.2	4.9	6.8	8.8	10.1	22.3	18.8	16.7	9.0	2.9	R0	R5

Problem 10-2

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	Nmax	WC
T0	3	9	12	3	32	5	54	4	25	78	R3	R6
T1	45	1	3	7	7	78	99	6	98	1	R4	R7
T2	7	34	34	43	5	687	0	95	21	4	R2	R5
T3	2	32	23	56	598	4	98	0	32	57	R1	R4
T4	123	2	67	34	855	0	0	7	45	9	R9	R2
T5	53	4	3	67	4	4	7	9	54	8	R5	R8
T6	12	13	14	23	13	12	15	34	32	12	R6	R9
T7	35	2	32	2	34	87	8	95	4	7	R8	R1
T8	88	88	88	99	99	99	88	88	88	88	R7	R0
T9	3	6	1	2	2	2	4	35	12	44	R0	R3

Problem 10-3

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	Nmax
T0	98	7	99	427	75	18	7	987	87	71	R4
T1	52	98	74	421	32	75	4	7	687	74	R6
T2	49	98	78	12	2	4	32	82	2	42	R7
T3	9	856	685	95	22	10	2	12	2	7	R8
T4	1	5	125	74	7	54	21	42	98	21	R9
T5	9	212	2	954	85	4	9	98	8	4	R0
T6	8	7	78	45	68	84	87	768	87	42	R1
T7	98	78	95	69	98	865	9	5	682	2	R2
T8	24	24	94	45	5	78	487	42	20	78	R3
T9	4	24	9	9	45	41	6	48	9	82	R5

Problem 10-4

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	WC	Nmax
T0	19.8	7.8	9.5	6.9	9.8	6.5	9.0	5.0	8.2	2.0	R9	R1
T1	9.9	21.2	2.0	15.4	8.5	16.6	9.2	9.8	8.2	4.6	R3	R2
T2	19.8	7.0	10.9	17.3	7.5	18.7	7.9	18.7	8.7	7.1	R4	R3
T3	18.5	17.7	7.8	4.5	6.8	8.4	8.7	5.8	8.7	4.2	R5	R4
T4	11.3	5.5	12.5	7.4	7.4	5.4	21.1	4.2	9.8	21.3	R6	R5
T5	25.2	19.8	17.4	21.2	3.2	7.5	4.5	7.6	8.7	7.4	R7	R6
T6	24.4	24.8	9.4	4.5	7.9	8.2	4.7	4.1	12.0	17.8	R8	R7
T7	4.7	2.4	9.1	4.9	4.5	4.1	6.7	8.3	8.1	8.2	R0	R8
T8	9.7	8.5	8.0	9.5	22.8	10.1	2.6	12.9	8.9	4.7	R1	R9
T9	14.6	19.8	7.8	12.0	20.5	14.3	31.2	18.2	4.4	10.8	R2	R0

Problem 10-5

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	WC	Nmax
T0	13.7	24.4	26.1	15.9	14.5	4.1	6.7	8.3	15.1	22.2	R1	R3
T1	7.3	15.5	21.5	7.4	11.4	5.4	21.1	4.2	9.8	10.3	R5	R7
T2	8.7	21.5	12.0	14.5	17.8	10.1	2.6	12.9	8.9	30.7	R0	R2
T3	11.9	21.2	9.0	15.4	18.5	16.6	9.2	9.8	19.2	25.6	R9	R1
T4	12.2	19.8	15.8	19.2	13.2	7.5	8.5	7.6	8.7	7.4	R5	R9
T5	14.4	24.8	5.9	14.5	9.9	8.2	4.7	4.1	10.0	7.8	R3	R5
T6	25.8	17.0	10.9	17.3	30.5	28.7	17.9	18.7	18.0	17.1	R7	R9
T7	10.5	7.7	7.8	14.5	16.8	8.4	4.7	6.8	2.7	4.2	R6	R8
T8	20.6	19.8	9.8	12.0	15.5	14.3	31.2	18.2	14.4	10.8	R8	R0
T9	6.8	17.8	29.5	8.9	26.8	6.5	9.0	5.0	20.2	12.0	R2	R4

Problem 10-6

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	Nmax	WC
T0	26.0	14.0	29.0	6.0	26.0	6.0	19.0	27.0	20.0	12.0	R3	R9
T1	11.0	20.0	9.0	15.0	18.0	15.0	9.0	23.0	19.0	45.0	R7	R6
T2	20.0	39.0	19.0	25.0	5.0	14.0	16.0	38.0	24.0	40.0	R2	R2
T3	25.0	17.0	5.0	15.0	30.0	18.0	7.0	8.0	8.0	27.0	R0	R0
T4	10.0	7.0	4.0	1.0	36.0	51.0	48.0	61.0	22.0	14.0	R1	R8
T5	11.0	28.0	4.0	41.0	9.0	18.0	20.0	19.0	10.0	35.0	R9	R1
T6	8.0	16.0	12.0	14.0	17.0	30.0	2.0	15.0	26.0	30.0	R6	R5
T7	13.0	13.0	15.0	19.0	34.0	7.0	18.0	7.0	14.0	7.0	R8	R7
T8	10.0	40.0	15.0	20.0	25.0	30.0	35.0	5.0	23.0	33.0	R4	R3
T9	1.0	10.0	50.0	60.0	18.0	24.0	31.0	44.0	20.0	23.0	R5	R4

Problem 10-7

	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	WC	Nmax
T0	11.0	12.0	13.0	14.0	5.1	6.9	7.0	8.0	9.0	5.5	R2	R9
T1	11.0	12.1	13.6	24.2	25.0	26.1	17.0	2.9	19.0	17.3	R3	R0
T2	3.4	12.2	13.7	15.8	18.8	16.0	17.0	8.0	7.7	16.4	R4	R1
T3	11.0	6.3	10.4	8.8	6.4	4.0	9.5	3.0	5.3	10.0	R5	R2
T4	15.2	18.3	7.7	16.2	15.0	8.1	17.0	9.0	12.3	11.4	R6	R3
T5	1.0	3.0	9.9	7.1	5.8	6.7	4.8	8.0	9.0	6.3	R7	R4
T6	6.9	12.7	6.2	14.0	15.0	16.0	20.0	18.0	19.0	5.0	R8	R5
T7	1.8	6.3	3.3	4.1	5.7	4.4	7.8	8.0	5.5	7.2	R9	R6
T8	10.0	2.9	3.8	9.4	9.7	5.7	7.2	9.3	16.1	7.7	R0	R7
T9	9.1	1.8	2.2	4.2	5.2	6.2	7.7	8.8	9.9	10.0	R1	R8