An Effective Approach for Distributed Meeting Scheduler

M. Sugumaran[1], P. Narayanasamy[2], and K. S. Easwarakumar[2]

[1]Department of Computer Science and Engineering,
Pondicherry Engineering College
Pondicherry -605 014, INDIA.
forsugu@yahoo.com

[2]Department of Computer Science and Engineering,
Anna University
Chennai - 600 025, INDIA.
sam@annauniv.edu, easwara@cs.annauniv.edu

Abstract

Meeting scheduling is generally considered as one of the most common activities that takes place in organizations. It is a distributed, time-consuming, iterative, and also a tedious task. Meeting scheduling is an example of a resource allocation problem. The principal resource considered for meeting scheduling is the persons' time. The resource requirement in a distributed environment is dynamic in nature and so the allocation of resources is difficult due to the lack of knowledge of the whole system. In this paper, three concepts such as equivalence classes of persons for delegation instead of a single person, A*-Algorithm as local search for finding common free slots for people, and multi-stage negotiation protocol for agent coordination are combined into a single approach and implemented to schedule both local and global meetings. This new approach aims a practical scheduler to schedule both local and global meetings, to have efficient schedule and flexibility to meet more meeting requests.

Keyword: Multi-Agent, Meeting Scheduling, A*-Algorithm, Artificial Intelligence, Negotiation Protocol.

I. Introduction

The basic problem in meeting scheduling is to find a common free time for all participants of a particular meeting. This problem becomes more complicated when there are various meetings to be scheduled concurrently with several constraints. Meeting scheduling is a distributed task, which is time-consuming, iterative, and also tedious. It can take place between two persons, or among several persons of the same organization or different organizations. Within organizations, substitute of persons for delegates become common as and when changes of assignments take place or due to some important assignments to be carried out. Meeting scheduling is the problem where agents can be used intelligently on behalf of users. A meeting scheduler is a system, which determines the date, time and location of a meeting for the associated participants with their constraints, and also manages various updates. Because of the inherent tedious, iterative and time-consuming features, meeting scheduling is considered for automation. The benefits of automated meeting schedulers are not only save time, effort on the part of human, but also gives more efficient schedules.

A number of scheduling problems such as multiprocessor scheduling, job-shop scheduling and timetable allocation have been investigated extensively in [1]. These problems are to construct a schedule for the given requirements. For the meeting scheduling problems, it is much more required to update the schedule whenever there is a change in the schedule or when a new meeting is to be scheduled. There are several commercial products available, but they are of limited functionalities. That is, they are just computational calendars with some special features like availability checkers, meeting reminders, etc. A review of several products can be found in [2]. None of these products is a truly autonomous agent capable of communicating and negotiating with other agents in order to schedule meetings in a distributed way taking into account of user's preferences and their calendar availability. A good work in distributed meeting scheduling [3-7] has been focused on solving meeting scheduling using a central host agent capable of communicating with all other agents using a negotiation based contracts [8]. The main purpose of the agent who hosts the meeting is to coordinate the search for a feasible schedule taking into consideration of attendees' calendars, but the user preferences are not taken into account. The work presented in [9] is an economic approach for distributed meeting scheduling. They used three centralized monetary based meeting scheduling systems and analyzed the tradeoffs between the mechanism complexity and information preferences and introduced the Clarke Tax Mechanism as a method for removing manipulability from them. Another work in [10] is based on modeling, communication constraints and preferences among agents. The agents exhibit intelligence that they are capable of negotiating and relaxing their constraints so as to reach an agreement on schedules with joint utility. Further agents also can react and revise the schedule due to dynamic changes. The scheduler presented in [11] considered equivalence classes of persons and used heuristics with common pruning technique to find the optimum schedule, is a centralized approach.

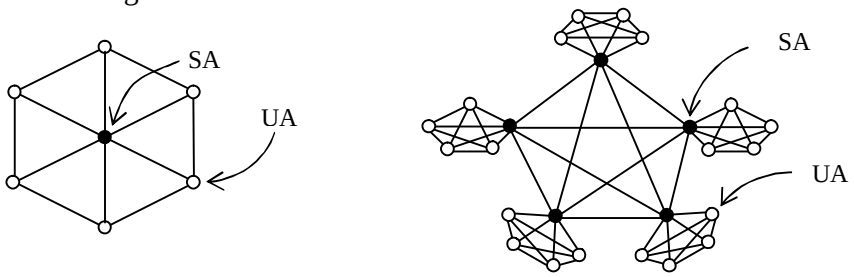
Crawford and Veloso [20] stated that there are a number of problems in the Microsoft Outlook approach. Microsoft Outlook largely ignores the negotiation step and issues of uncertainty about other users' calendars. The scheduling features in Microsoft Outlook rely largely on an open calendar systems (where users are required to make their calendars publicly viewable within the organization). A major limitation of Microsoft Outlook is that it will not consider moving existing meetings on behalf of the user. In many cases, it may only be possible to schedule very large meetings when many users move smaller meetings.

Most of the processing of Distributed Meeting Scheduler (DMS) takes place in the local schedulers. Sugihara et al (1989) presented a heuristic algorithm for Timetable Rearrangement (TR) in which an exhaustive search algorithm is used with common pruning techniques. The time complexity of their heuristic algorithm is exponential in the worst case. Since TR is NP-hard and Sugihara's approach time complexity is exponential, we use A*-Algorithm with heuristic functions to reduce the processing time. Further, the processing which takes place in the local schedulers may affect globally. Since we are using equivalence classes of persons for delegation to provide flexibility in the local schedulers, this will certainly reduce the number of messages sent and number of iterations executed.

In this paper, a combined approach comprises of equivalence classes of persons, A*-Algorithm, and multi-stage negotiation protocol is proposed. A group of persons of any category is considered equivalent. That is, the persons in a group are treated equivalent for any meeting or task. This makes the organizations to have flexibility in scheduling meetings and service more requests. We combined the A*-Algorithm approach [15] and multi-stage negotiation protocol for agent coordination to schedule meetings in a distributed paradigm. A* facilitates the search process of the local scheduler to provide efficient schedules.

II. The Multi-Agent Meeting Scheduler

The basic model of the distributed meeting scheduler consists of different organizations connected by a communication network. Each organization (node) is assigned a set of agents, one agent per person. In an organization, there is one agent called ***scheduling agent*** (**SA**)**,** and the other agents are called ***user agents*** (**UAs**), shown in Figure 1. These agents act and negotiate with others on behalf of their associated users and their organizations autonomously. The SA acts on behalf the hosts of the meetings to schedule local meetings and also on behalf of its organization for global meetings. Local meetings are conducted within an organization, whereas global meetings are conducted between two or more organizations.



(a) Agents at a single node (b) SAs are connected by a network

Figure 1. The Meeting scheduler model

The user who proposes a meeting is called ***initiator*** and the corresponding agent is called ***host***. The users who are invited to attend for that meeting are called ***participants*** and their corresponding agents are called ***invitees***. The invitees have been divided into two groups as ***executives*** and ***regular invitees***. This division depends on the problem domain. That is, for some problems, the invitees are divided into executives and invitees; and for some other problems there is only invitees. The executives are not generally considered for change of persons because of their unique features, whereas the regular invitees are considered for change of persons for most of the applications. In case, for a particular invitee no substitute is required that also could be represented in the proposals to invitees. At any point of time an agent may act as a host in a meeting, an invitee in some other meeting or an executive yet some other meeting.

A meeting schedule consists of a set of n meetings scheduled among a set of m persons. A schedule of n meetings is represented as an 11-tuple:

$$S(n) = (P, M_n, <, h, t, p, w, a, d, \tau, \rho),$$

where $P = \{p_1, p_2, \ldots, p_m\}$ is a set of m persons spread across the network; $M_n = \{m_1, m_2, \ldots, m_n\}$ is a set of n meetings to be scheduled among these m persons of P; $<$ is a partial order on M_n; $h(m_i)$ is the host of the meeting m_i; $t(m_i)$ is the time duration of meeting m_i; $p(m_i)$ is a set of groups of persons taken from P such that exactly one person in each group is required to attend the meeting m_i; $w(m_i)$ is a set of time instances at which meeting m_i can start for local meeting, whereas it represents the weight (or priority) of the meeting m_i for global meeting; $a(m_i)$ is the arrival time of meeting request m_i or the time at which $h(m_i)$ becomes aware of the need to schedule m_i; $d(m_i)$ is the deadline by which the $h(m_i)$ needs to schedule the meeting m_i; $\tau(m_i)$ is the start time of meeting m_i, i.e., at which the meeting m_i is finally scheduled; and $\rho(m_i)$ is the set of attendants of meeting m_i.

The parameters P, M_n, $<$, h, t, p, w, τ, and ρ are used in both local and global meetings, whereas a and d used for only global meetings. The local and global meeting requests are identified by referring the content of w. If the content of w is a single item it is a global meeting, or if it is a set of items the request is a local meeting.

Further the schedule satisfies the following five conditions for both local and global meetings. Let m_i and m_j be any two meetings.

C1: No person attends more than one meeting simultaneously.
C2: If $m_i < m_j$, then m_j starts after m_i ends.
C3: For each group $g \in p(m_i)$, exactly one person can attend the meeting m_i.
C4: m_i can start at one of the time instances in $w(m_i)$ for local, or in $(a(m_i), d(m_i))$ for global meeting.
C5: m_i should be scheduled within the deadline $d(m_i)$.
C6: The time interval of a meeting is contiguous and cannot be split across days.

In a schedule $S(n)$, the parameters $<$, t, p h, a, d and w represent the input requirements of meetings, whereas τ and ρ represent a schedule of meetings that satisfies all the requirements. The assignment of rooms to meetings can easily be incorporated into the schedule by considering rooms as pseudo-persons. For example, selection of a room or venue for m_i from rooms R_1, R_2,…, R_k is represented by considering a group of corresponding pseudo-persons in $p(m_i)$ [11].

The architecture of our agent is very similar to the systems used in [12, 13, 18]. Each agent has six main components: user interface, user preference, negotiation, location manger, and message constructor/decoder and calendar manager. Each component does some specific tasks, whereas the negotiation module plays the important role. It gathers information from all other five modules and computes the possible meeting schedule using negotiation protocol.

All participants of a particular meeting may not have a common free time. So, whenever scheduling a meeting among a group of persons, some kind of coordination is required. Negotiation is a key coordination technique used to address several DAI (Distributed Artificial Intelligence) issues [14]. A significant part of the coordination work is under negotiation. Negotiation is the communication process of a group of agents in order to reach a mutually accepted agreement on some matter. We use a multi-stage negotiation protocol, based on the contract net protocol [8], for agents' negotiation.

III. A*-Algorithm for the Local Meeting Scheduler

A*-Algorithm, is a very useful algorithm, provides a best-first search through a graph or tree representing a problem space. We use a tree for representing the problem space, called ***search tree***. Each node in the search tree represents the description of a state of the problem. The set of operators used in the A* describes a way of changing state description. Each node contains, in addition to a description of the problem state it represents, an indication of how promising it is, a parent link that points back to the node from which it came, a list of nodes that were generated from it. The parent link is used to find the path to the goal node once the goal is found. We use two data structures in the A*-Algorithm, called ***open-heap*** and ***node-set***. *Open* heap having nodes that have been generated but not yet been expanded. A node of the *open-heap* is mainly used to keep the node's heuristic value and its node-id. The *open-heap* is a min/max heap in which min/max heuristic value node is at the root, so that it always gets expanded before all other nodes. *Node-set* having nodes that have already been generated. It is mainly used to check whether a node is already in the search tree or not. The main purpose of using this set is to reduce the complexity of the algorithm in search of nodes. Whenever a new node is generated in the search tree, it is verified with the *node-set*. If it is not available in the node-set, heuristic function is applied to it. Then its heuristic value with node-id is inserted into the heap and the set.

In this model, six operations are used such as SL (Shift Left), SR (Shift Right), CP (Change of Person), XP (eXchange of Person) [15], CSL (Continuous SL) and CSR (Continuous SR) to rearrange the scheduled meetings such that a new meeting request could be fixed in a particular duration. Based on these operations, heuristic value is determined and with this value the node is inserted into the *open* heap. This heuristic value will be the deciding factor for selecting the next node in the search tree expansion. We define a heuristic function that counts the number of persons available (or not available) of those slots to be considered for the new meeting in a particular duration [15]. The heuristic function enables the algorithm to search the more promising path first by choosing the node or branch using the *open-heap*. We call this heuristic function f', is an approximation to a function f that gives the true evaluation of the node. For many applications, the function f' is defined as

$$f'(n) = g(n) + h'(n)$$

where the function $g(n)$ is a measure of the actual cost of getting from the initial state to the current node n and $h'(n)$ is an estimate of the additional cost of getting the goal node from the current node n [16, 17, 19]. $h'(n)$ is the place where knowledge about the problem domain is to be exploited. The combined function $f'(n)$ represents an estimate of the cost of getting from the initial node to a goal node along the path that generated the current node n.

The actual operation of the algorithm is very simple. It proceeds in steps, expanding one node at each step, until it generates a node that corresponds to a goal state. At each step, it selects the most promising node that has been generated but not expanded, called ***current-node***. It generates the successors of the *current-node* and checks if any of them have been generated before. For the new nodes, it applies the heuristic functions then adds them to the *open-heap* and the *node-set*. By doing this check, it is guaranteed that each node appears only once in the tree. Then the next step begins. This process continues till it gets the goal node, or the open-heap is empty. The six operations used in the A*-Algorithm, shown in Figure 2, is briefly given below.

Let m_{n+1} be the current meeting to be scheduled and $p(m_{n+1}) = \{g_1, g_2, \ldots, g_r\}$ be the set of groups of persons to be considered for scheduling m_{n+1}. Let $\rho(m_{n+1})$ be the set of attendees for m_{n+1}, chosen from $p(m_{n+1})$ one person per group. Further, let $(x, x + t(m_{n+1}))$ be the time interval to be considered for scheduling the meeting m_{n+1}. These assumptions are used in the following six operations.

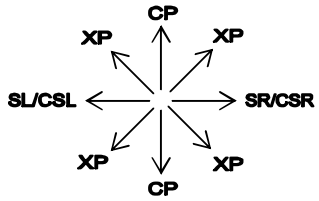


Figure 2. Operations used in the A*

SL: Shifts meetings horizontally to the left. For each person $j \in \rho(m_{n+1})$ and each meeting $m_i \in (x, x + t(m_{n+1}))$, if there is a time interval $(s, s + t(m_i))$ to the left and disjoint with $(x, x + t(m_{n+1}))$ such that $s \in w(m_i)$ and no person in $\rho(m_i)$ attends any meeting during $(s, s + t(m_i))$, then change the start time of m_i to s.

SR: Shifts meetings horizontally to the right. For each person $j \in \rho(m_{n+1})$ and each meeting $m_i \in (x, x + t(m_{n+1}))$, if there is a time interval $(s, s + t(m_i))$ to the right and disjoint with $(x, x + t(m_{n+1}))$ such that $s \in w(m_i)$ and no person in $\rho(m_i)$ attends any meeting during $(s, s + t(m_i))$, then change the start time of m_i to s.

CP: Changes a person vertically within a group. This operation makes free slots for the persons $\rho(m_{n+1})$ in $(x, x + t(m_{n+1}))$ within groups. For each meeting $m_i \in (x, x + t(m_{n+1}))$, if no person in $\rho(m_i)$ except some person $j \in \rho(m_i)$ attends m_i and if there is a person k in $\{a \mid a, j \in g_r$ and $g_r \in p(m_i), k \neq j\}$ who does not attend any meeting during $(\tau(m_i), \tau(m_i) + t(m_i))$, then change the attendant j of m_i to k.

XP: Exchanges persons within a group on different meetings. This makes free slots for the current meeting by making changes diagonally. For any two persons $j, k \in g_p$ in $p(m_i)$ and $j, k \in g_q$ in $p(m_n)$ such that $j \in \rho(m_i)$ and $k \in \rho(m_n)$, then swap j and k between $\rho(m_i)$ and $\rho(m_n)$ if j is free in $(x, x + t(m_n))$ and k is free in $(x, x + t(m_i))$.

CSL: Shifts the meetings in $(x, x + t(m_{n+1}))$ as well as the meetings to the left of this interval further to the left in order so that each person $j \in \rho(m_{n+1})$ is free in $(x, x + t(m_{n+1}))$.

CSR: Shifts the meetings in $(x, x + t(m_{n+1}))$ as well as the meetings to the right of this interval further to the right in order so that each person $j \in \rho(m_{n+1})$ is free in $(x, x + t(m_{n+1}))$.

The A*-Algorithm using the above operations in the local meeting scheduler is given below.

Algorithm1

Schedule($T(n)$, $I(m_{n+1})$)

// $T(n)$ – schedule of n meetings.
// $I(m_{n+1})$ – inputs of the requested meeting, m_{n+1}.
// start-time(m_{n+1}) – a start time at which m_{n+1} could be scheduled.
// deadline(m_{n+1}) – before which m_{n+1} should be scheduled.
// $p(m_{n+1})$ – a set of group of persons considered for $\rho(m_{n+1})$.

1. Initialize the ***open-heap*** and the ***node-set***.
2. Create a search tree, G, starting with the root node s of the schedule $T(n)$.
3. Insert s into the ***open-heap*** and the ***node-set***.
4. **While**(answer is not found)
5. **Begin**
6. If(***open-heap*** is empty)
7. Print the error message and exit.
8. $n \leftarrow$ delete the root of the ***open-heap***.
9. If(n is a goal node)
10. Schedule the meeting m_{n+1} and exit.
11. Generate a set M of successors which are satisfying the distance factor and are not in the ***node-set*** by applying the operations: **CP**, **XP**, **SL**, **SR**, **CSL**, and **CSR** on n; insert into the ***node-set***.
12. For each node in M, apply the heuristic function and establish a pointer to n.
13. Insert the nodes of M into the ***open-heap*** according to their heuristic values.
14. **End**
15. **End Schedule**

Theorem 1: *If the operations CP, XP, SL, SR, CSL, and CSR are applied to Algorithm1, it finds an optimal solution if any exists*.

Proof:
Let us consider Figure 2. The operations CP, XP, SL, SR, CSL and CSR are used to enumerate all possible states for the current meeting, m_{n+1}. As the algorithm starts with the root node and the operations are applied in each level of the search tree, it generates a finite number of nodes on each level. The heap always has the minimum value or the most probable node at the root for expansion and removed each time of expansion.

Case (i): If slots are free for $\rho(m_{n+1})$ in $(x, x + t(m_{n+1}))$, Algorithm1 finds the solution and that is the optimal one.

Case (ii): If slots are not free for $\rho(m_{n+1})$ in $(x, x + t(m_{n+1}))$, then the six operations CP, XP, SL, SR, CSL and CSR are applied to enumerate all possible states for the current meeting, $m_{n+1,}$ from the current node.

The algorithm starts with the root node and the operations are applied at each level of the search tree. As the operations correctly capture all possible states from the root and the nodes are stored without duplicates into the heap, it generates a finite number of nodes on each level. Since the heap is used to get the most probable node for expansion and removed, it leads an optimal solution in a fewer levels of expansion, otherwise ends up with the empty heap.

Theorem 2. *The heuristic function used in the Algorithm1 gives the optimal solution.*

Proof:
Let $L(n)$ be the minimum cost incurred on applying the operations to arrive n from the root. When this is proved, the optimality of the algorithm follows since when z (solution node) is chosen at line 8, $L(z)$ will give the minimum cost operations required to arrive z from the root.

Basis Step (i = 1): The first time we arrive at line 8, the root is chosen. Since L(root) is zero, L(root) is the minimum cost operations from root to root.
Inductive Step: Assume that for all $k < i$, the k^{th} time we arrive at line 8, $L(n)$ is the minimum cost operations from root to n.

Suppose that we are at line 8 for the i^{th} time and we remove n from the top of the heap with minimum value $L(n)$.

First we show that if there is a sequence of operations applied from the root to a node w whose cost is less than $L(n)$, then w is not in the heap (that is, w was previously removed from the heap and expanded at line 8).

Suppose that, by contradiction, w is in the heap. Let P be a minimum path from the root to w, x be the node nearest to the root on P that is in the heap, and u be the predecessor of x on P. Then u is not in the heap, so u was chosen at line 8 during a previous iteration in the while loop. By the inductive assumption, $L(u)$ is the minimum cost operations from the root to u.

$$\begin{aligned} \text{Now } L(x) &\leq L(u) + h(u, x), \text{ where } h(u, x) \text{ is the heuristic value from } u \text{ to } x \\ &\leq \text{cost of } P \\ &< L(n). \end{aligned}$$

But this inequality shows that n is not the node in the heap with minimum $L(n)$. This contradiction completes the proof that if there is a path from the root to a node w whose cost is less than $L(n)$, then w is not in the heap.

A. Negotiation Protocol for Global Meeting Scheduler

A negotiation strategy refers to a decision function used by an agent to make a proposal or counter proposal. Different negotiation strategies have been used to solve negotiable problems. But the problem of choosing the correct negotiation strategy has significant impact on convergence. The negotiation protocol used here is based on [8] and works in multi-stages. Depending upon the problem requirement, the number of stages may vary (two or three). The negotiation protocol presented here uses three phases, shown in Figure 3. The host negotiates with executives in the first phase, negotiates with regular invitees in the second phase, and then it takes decision about confirmation or cancellation of scheduling the current meeting in the third phase.

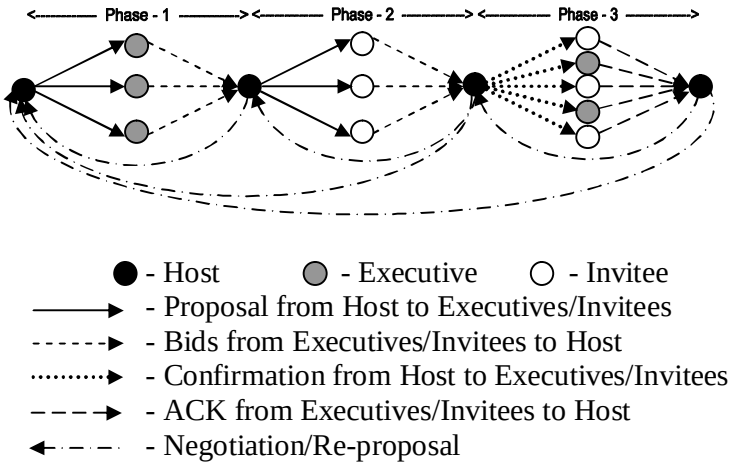


Figure 3. Multi-stage negotiation protocol

Agents communicate through messages by exchanging information at higher levels of abstraction so as to have less communication cost. Since the selection of a room or venue for m_i from R_1, R_2, …, R_k is represented by considering a group of corresponding pseudo-persons in $p(m_i)$ [11], this may be considered along with executives or before executives for negotiation. In general, on receiving a request/proposal, the host, executive and invitee searches the user calendar for free slots. If enough free slots are not available, it applies the A*-Algorithm to have reschedule of meetings. Based on this result, called tentative schedule, it chooses some (three, by default) slots and blocks, then sends them as proposals or bids. After fixing the current request, the tentative schedule of previous meetings is made permanent. The negotiation protocol is given below in three phases.

Phase 1 - Proposals to the Executives

In this first phase, the host waits for request from its user. After receiving a request, it consults its' user calendar with user preferences for availability then sends proposals to the executives. The proposals consist of a set of <Date, Hour> pairs with duration of a particular meeting. Normally, the agents use three proposals at a time. On receiving the proposals, the executives process the proposals and then send their bids to the host. The host collects the bids from the executives then decides whether to go for the next-phase, negotiation or new proposals.

- **Host receives Request and sends Proposals to Executives**: When the host receives a request from its user to schedule a meeting, it looks at the user calendar with user preferences for free time slots to schedule the current meeting request. If it finds any free time slots, prepare the proposals and sends them to the executives, otherwise applies A*-Algorithm to find if there is any possibility to schedule the meeting request. If free slots are there, based on the importance of the meeting request, the host chooses some time-slots, blocks them in its calendar and proposes them through messages to the executives.

- **Executives receive Proposals and send Bids**: Executives receive the proposals then look at their users' calendars with user preferences. The executives decide based on the priority of the meeting, instruction from the host and its own opinion, it may apply A*-Algorithm for the proposals. Then they send their bids to the host whether the proposed time slots are acceptable and also its availability of other free slots in the neighborhood of the proposals. If an executive sends a "yes" bid, the corresponding time slot is blocked in the executive's calendar. If there are any time slots already blocked for the same meeting, they are unblocked.

After collecting the bids from the executives, the host decides whether to go for second phase, negotiation with those executives who are not available for the proposals or new proposals.

Phase II - Proposals to the Invitees

In the second phase, the host sends the proposals to the invitees based on the result of the bids received from the executives and waits for the bids from the invitees. On receiving the proposals, the invitees process the proposals and then send their bids to the host. After collecting the bids from the invitees, the host decides whether to go for third phase, negotiation with those invitees not available for the proposals or new proposals.

- **Host sends Proposals to Invitees**: The host sends the proposals based on the result of the bids received from the executives to all invitees and waits for the bids from them.
- **Invitees Receive Proposals and send Bids**: The invitees receive the proposals and decide based on the priority of the meeting, instructions from the host and its own opinion, it may apply A*-Algorithm for the proposals. Then they send their bids to the host whether the proposed time slots are acceptable and also its availability of other free slots in the neighborhood of the proposals. If an invitee sends a "yes" bid, the corresponding time slot is blocked in the invitee's calendar. If there are any time slots already blocked for the same meeting, they are unblocked.
- **Host receives Bids from the Invitees and takes decision**: The host collects and evaluates the bids received from the regular invitees. If there are any common free slots accepted by all, the host picks up the earliest slot and unblocks other slots, then enter into the next phase. If there are no common free time slots acceptable to all, it negotiates and try to get the time-slots free with those invitees who are all sent "no" bids, if not, the host sends cancellation message to all executives and invitees and starts new proposals with the available information obtained so far from the executives and invitees.

Phase III - Confirmation/Cancellation to Executives/Invitees

In this phase, the host sends Confirmation/Cancellation of the current meeting to all executives as well as invitees. On receiving the Confirmation/Cancellation message from the host, the executives and invitees send one of ACK/OK/Cancel reply message to the host. Then the host performs the appropriate action.

- **Host sends Confirmation/Cancellation message to Executives/Invitees**: The host sends Confirmation or Cancellation message, based on the decision taken at the end of the second phase, to all executives and invitees. If it is a Cancellation message, the host unblocks the slots already blocked for this meeting, sends cancellation message to all executives and invitees then waits for the OK reply message from the executives and invitees to complete the transaction. Otherwise, it sends a Confirmation message to all executives and invitees and waits for the ACK message from them.
- **Executives/Invitees send ACK/OK/Cancel message to the Host**: On receiving a confirmation message from the host, the executives and invitees check whether the time slot specified in the confirmation is still available for the current meeting. If it is available, they reserve the time slot, record the scheduling of the meeting, and then send the ACK to

the host. If any of the executives or invitees find that the already blocked time slot is not available during the mean time, it applies the A* algorithm to find any user is free in that particular group for the promised slot, if possible then it reserves that promised time slot for the current meeting and sends an ACK message, otherwise sends a Cancel message to the host. If it is a Cancellation message it unblocks the blocked slots for the current meeting and send OK message.

- **Host waits for ACK/OK/Cancel message**: If the host receives ACK from all executives and invitees then schedule the meeting in that particular slot. If at least one of them is a Cancel message, it negotiates to get the time slot free with those executives/invitees who all sent Cancel message, otherwise it unblocks the blocked time slots for the current meeting, then it may enter the first phase for new proposals with the available information obtained so far, or inform the user that the current meeting request is unable to schedule.

IV. Implementation

The DMS was implemented in Linux environment (Red Hat Linux 8.0) using the network simulator (ns2.1b8a). The OTcl interface for the ns2 offers creation of nodes of the network, the communication links between the nodes, and placing the agents on the nodes, etc. The type of traffic and other inner details are supported by the ns2. The agents located on different organizations communicate and negotiate with each other on behalf of their users over the network.

The different states of an agent of the global meeting scheduler are given in Figure 4. All agents use this state transition diagram. At any point of time, the agents may be in any one of the states. We use this state transition diagram to describe the coordination among the host, executives and invitees during the scheduling process. There are one or more transitions from each state.

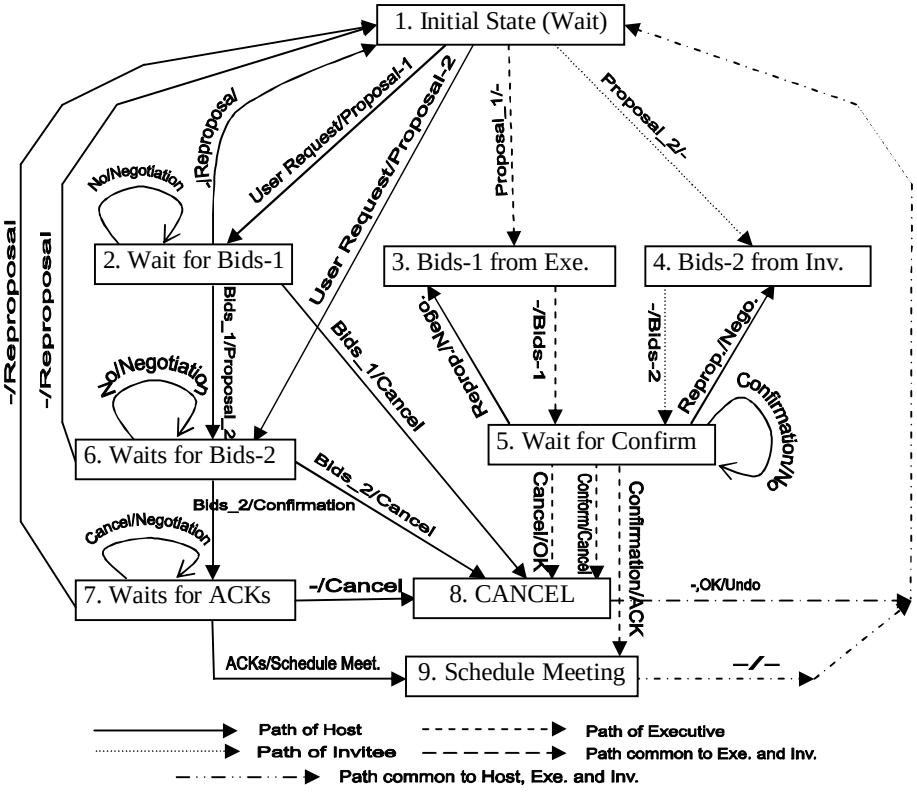


Figure 4. State transition diagram of an agent

Usually the transitions are from one state to other states, sometimes to the same state. The states are represented by rectangles and transitions by arrows. For each transition there is an input and output message with <*in*>/<*out*> format, where <*in*> represents a condition or an input message of an event and <*out*> represents an output message. A transition takes place when an event occurs, a certain condition is satisfied, or an input message is received, causes the agent to process and sends an output message. The symbol "-" is used to indicate that there is no input or output message.

All agents are initially in state 1, called the waiting state. When an agent receives a request from its user to organize a meeting, that agent becomes the host of that meeting. Then the negotiation may proceed with three or two phases, according to the inputs given by the user. The paths taken by the host, executives and invitees are given in Table 1. All agents run simultaneously using the same agent code. The implementation of the negotiation protocol with three phases is shown in Figure 4.

Table 1. Path taken by host, executives and invitees

No. of Phases	Host	Executives	Invitees
3	{1, 2, 6, 7, 9, 1}	{1, 3, 5, 9, 1}	{1, 4, 5, 9, 1}
2	{1, 6, 7, 9, 1}	–	{1, 4, 5, 9, 1}

In the first phase, all agents are initially in the waiting state 1. When an agent receives a meeting request from its user it becomes the host for that meeting, say m_{n+1}. The host prepares and sends proposal-1 to the agents, mentioned as executives by the user, enters and waits in state 2 for bids-1 from them. The agents who receive the proposal-1 become the executives of the meeting m_{n+1}, reach state 3, prepare bids-1 and send to the host then reach state 5 and wait for confirmation or cancellation message from the host. The host receives the bids-1 from all the executives, processes them and decides one of the following:

- Negotiates with those executives who are not available for the proposal-1.
- Enters the state 1 for new proposal if the executives are not having the same free slots, or a cancel message to all executives, enters state 8 to close the transaction for this meeting and then enters the state 1.
- Enters the second phase if all executives have same free slots.

In the second phase, the host prepares proposal-2 and sends to the agents, mentioned by the user as the invitees, then enters and waits in state 6 for bids-2 from them. The agents in the state 1 who receive the proposal-2 become the invitees of the meeting m_{n+1}, enter state 4. The invitees process the proposal-2 and send their bids-2 to the host, enter the state 5 and wait for the conformation or cancellation message. After collecting the bids-2 from the invitees, the host decides one of the following:

- Negotiates with those invitees who are not available for the proposal-2.
- Enters state 1 for new proposals if the invitees are not having free slots, or a cancellation message to all the executives and invitees.
- Confirmation message to all the participants (both executives and invitees).

In the third phase, the host sends Conformation/Cancellation message to all participants (both executives and invitees) and enters state 7/8 respectively. For this, the participants may send one of ACK/OK/Cancel reply messages to the host.

- If it is a Cancellation message, the executives and invitees reply with OK message, enter the state 8 and do all the relevant undo operations then enter the state 1.
- If it is a Confirmation message, the executives and invitees may reply with ACK/Cancel reply message, enter state 9/8 and schedule the current meeting/undo the operations, close the transaction and then enter the state 1.

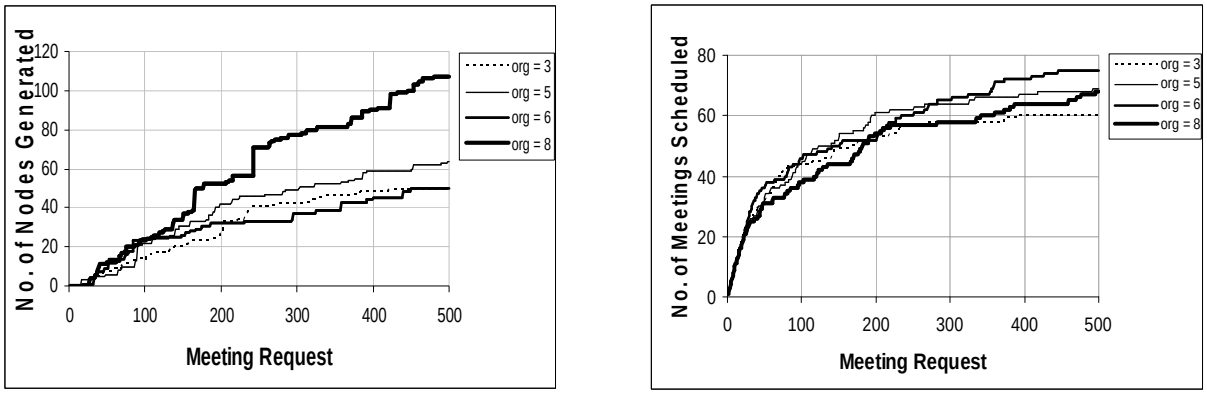
The host receives one of OK/Cancel/ACK reply messages from the participants.

- If the reply message is Cancel, the host negotiates with those executives or invitees for the confirmation slots.
- If it is ACK/OK reply message, the host enters the state 9 and schedules the meeting/undo the relevant operations then enters the state 1.

V. Results

This section contains the experiments and results of the DMS. Three kinds of experiments were performed to study the effects of DMS. The first sets of experiments were performed where global meetings are scheduled with different densities of local schedules. The second experiments were performed when the duration of meetings are varied. The last sets of experiments were performed where the numbers of slots in calendars are varied. The performance parameters of the scheduler are the number of organization (org) is 3 to 8, number of time slots in a calendar (ns) is 24, number of meetings (n) vary between 1 and 500, number of persons in each organization (m) is 8, and duration of meetings (dur) is between 1 and 4.

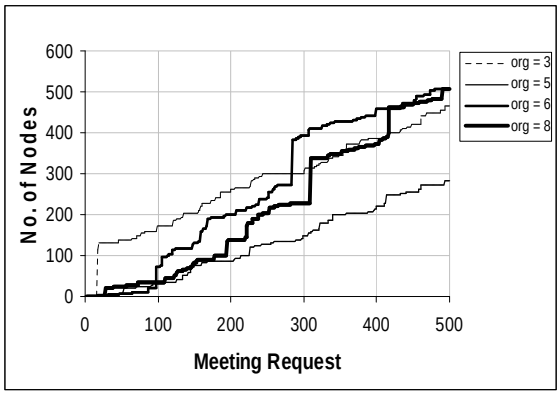
A. ***Global Meetings with No Local Meetings***: It is seen from Figure 5 that the number of meetings scheduled increases as the number of organization increases (3, 5, and 6), however it is reduced for the number of organization 8. This is due to unavailability of the participants and hence the number of nodes generated is particularly more for the case of number of organization 8. It is observed that, as there are no local meetings in the local schedules for rescheduling so as to have free slots for global meetings, the number of global meetings scheduled is reduced. Further, since the rescheduling operations are not applied in the local schedules, the numbers of nodes generated are also less.



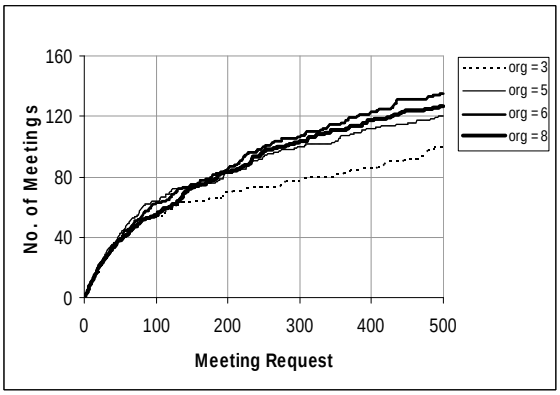
(a) (b)

Figure 5. Global meetings while no local meetings

B. ***Global Meetings with Lightly Loaded Local Meetings***: Figure 6 shows the performance of the DMS with global meetings while local meetings are lightly loaded in the local schedules. Here, the number of organizations is taken as 3, 5, 6, and 8, whereas the number of meeting requests varied between 1 and 500. For the first 50 meeting requests, the percentage of meetings scheduled is nearly 100% and nearly same for all cases of organizations. However, as the number of meeting requests increases, the percentage of meetings scheduled is gradually reduced for all cases of organizations. This is due to the more number of filled slots in the calendars of the local schedules.



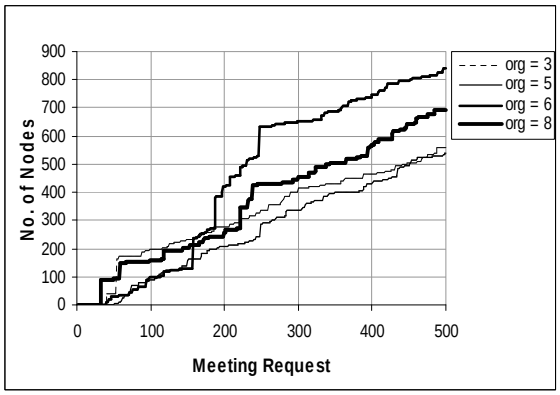
(a)



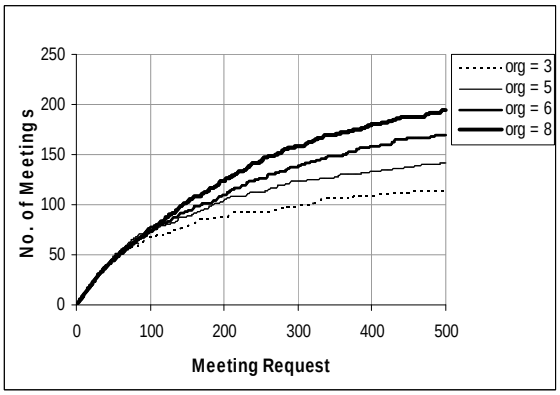
(b)

Figure 6. Global meetings while local meetings are lightly loaded

C. ***Global Meetings with Moderately Loaded Local Meetings***: Figure 7 shows the effects of scheduling global meetings while local meetings are moderately loaded. It is observed that the number of meetings scheduled is 100% for all the cases of organizations at the beginning as the slots of the local schedules are free. After that, the percentage of meetings scheduled is gradually reduced for all the cases of organizations as the number of meeting requests increases and the reductions of free slots in the calendars. However, as the number of organizations is increased the number of meetings scheduled is also increased with respect to the increasing number of meeting requests. This may be due to the availability of the participants from more number of organizations.



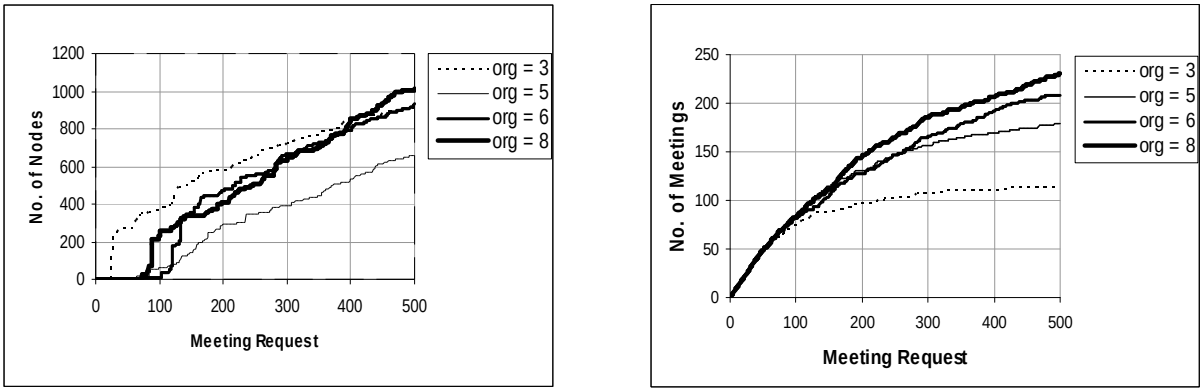
(a)



(b)

Figure 7. Global meetings while local meetings are moderately loaded

D. ***Global Meetings with Heavily Loaded Local Meetings***: Figure 8 shows the effects of rescheduling operations for scheduling global meetings while local meetings are heavily loaded. For the first 35 meeting requests, the number of meetings scheduled is 100% for all the cases of organizations. However as the number of meeting requests increases the percentage of meetings scheduled is reduced gradually for all organizations. Particularly in this case, the number of meetings scheduled is high as the number of scheduled local meetings available is more for rescheduling and hence the numbers of nodes generated are also high.



(a) (b)

Figure 8. Global meetings while local meetings are heavily loaded

E. ***Comparisons of DMS with Different Approaches***: The overall performance of the DMS using A*-Algorithm with and without negotiation protocol under different densities of local meetings, such as global meetings with no local meetings, global meetings with local meetings lightly loaded, global meetings with local meetings moderately loaded, and global meetings with local meetings heavily loaded with respect to different sets of simulations were carried out and the results are briefly given in Table 2.

Table 2. DMS with varying density of local meetings with A*

Density of Local Meetings	Average no. of Global Meetings Scheduled	
	Without Negotiation	With Negotiation
No Local Meetings	8.23	13.73
Local Meetings - Lightly Loaded	6.13	10.33
Local Meetings - Moderately Loaded	4.83	6.86
Local Meetings - Heavily Loaded	2.53	3.93

It is observed that as the density of the local schedules (or the number of local meetings) increases, the average number of global meetings scheduled is reduced with respect to the number of meeting requests in both schedulers with or without negotiation protocol. In all the four types of local meetings, the scheduler with negotiation protocol performs better than the scheduler without

negotiation protocol. It is also generally observed that for smaller numbers of meeting requests the average number of meetings scheduled is high and as the number of meeting requests increases, the average number of meetings scheduled is decreased irrespective of the type of local meetings.

The summary of the distributed meeting scheduler with the different densities of local meetings with respect to the number of organizations, number of global meeting requests, number of meetings scheduled with and without negotiation, number of local meeting requests, number of local meetings scheduled with and without A*-Algorithm, and user preferences is given in Table 3.

Table 3. Comparison of distributed meeting schedulers with different density of local meetings

Type of Meetings	No. of Org	Global Meetings			Local Meetings				
		No. of Requests	#Meetings Committed		No. of Meetings	#Meetings Committed			
			General	With Negotiation		General	Without Negotiation & A*	With User Preference	With A*
No Local Meetings	3	500	60	19	-	-	-	-	-
	4	500	68	26	-	-	-	-	-
	5	500	69	30	-	-	-	-	-
	6	500	75	35	-	-	-	-	-
	7	500	72	26	-	-	-	-	-
	8	500	68	29	-	-	-	-	-
Local Meetings-Lightly Loaded	3	379	55	14	121	45	31	9	5
	4	376	37	15	124	65	51	12	2
	5	387	60	23	113	60	52	3	5
	6	371	55	26	129	82	65	10	7
	7	366	53	20	134	89	66	16	7
	8	378	50	28	122	79	64	7	8
Local Meetings-Moderately Loaded	3	251	33	9	249	81	57	15	9
	4	255	32	6	245	106	76	11	19
	5	261	38	11	239	104	76	19	9
	6	246	32	9	254	139	102	19	18
	7	243	36	15	257	153	118	19	16
	8	249	35	11	251	159	128	18	13
Local Meetings-Heavily Loaded	3	121	18	6	379	95	66	14	15
	4	121	18	6	379	95	66	14	15
	5	113	19	4	387	160	119	20	21
	6	129	20	10	371	190	145	27	18
	7	134	22	7	366	195	149	22	24
	8	122	21	9	378	211	163	21	27

F. ***Local Meetings with Different Durations of Meetings***: A set of simulations is carried out to schedule only local meetings without global meetings to see the performance of the DMS on different durations of meetings (dur = 1, 2, and 3). Here, the parameter values used are: number of meetings is 100, number of time slots is 16, and the number of organization is 3.

In Figure 9, it is clearly seen that the percentage of meetings scheduled is reduced as the duration of the meetings increases. The scheduler based on A*-Algorithm performs better than without A*-Algorithm in all the three cases of meeting durations.

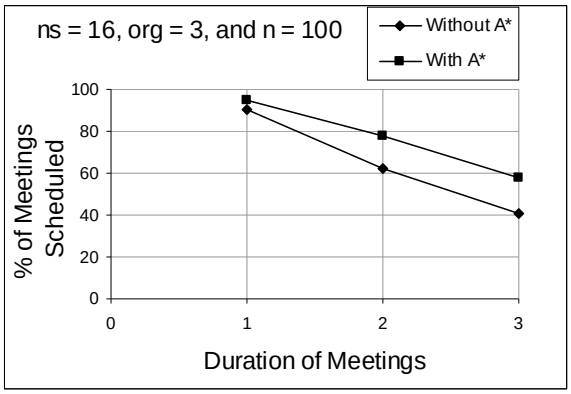


Figure 9. DMS with only local meetings for durations 1, 2, and 3

Another set of simulations is carried out to schedule global meetings when the local meetings are lightly loaded with the combinations of negotiation protocol and A*-Algorithm, particularly under different durations of meetings. For this run the parameters used are: the number of slots of the calendar (ns) is 8, number of organizations (org) is 8, duration of meetings (dur) are 1, 2, and 3, and different sets with number of meetings (n) are 1, 25, 50, 100, 200, 300, 400, and 500. The results are given from Figures 10 to 13.

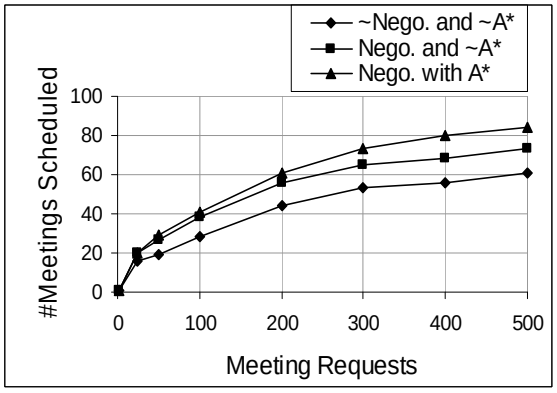


Figure 10. Local meetings lightly loaded with maximum duration 1

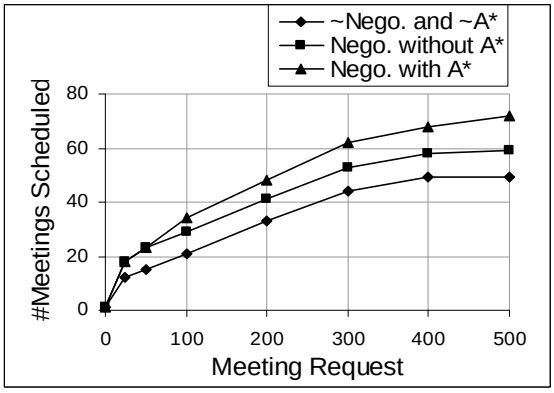


Figure 11. Local meetings lightly loaded with maximum duration 2

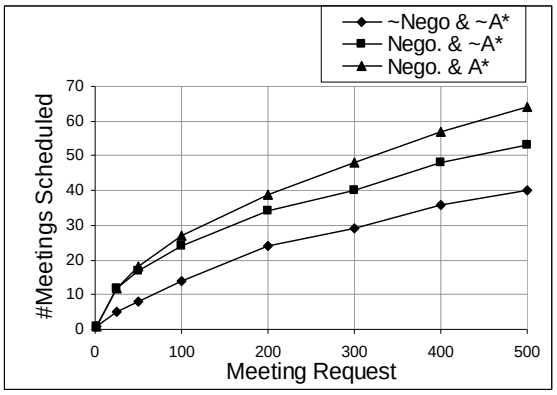


Figure 12. Local meetings lightly loaded with maximum duration 3

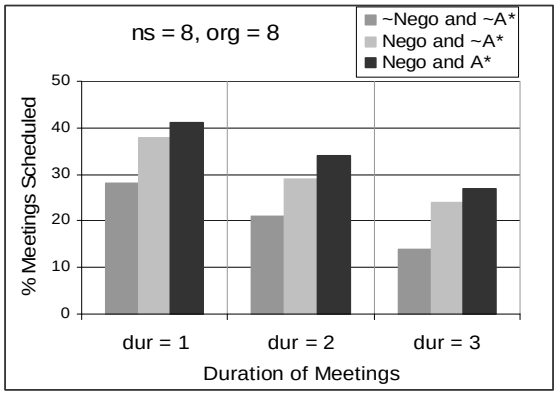


Figure 13. Local meetings lightly loaded with different duration of meetings

It is observed from the Figures 10 to 12 that the numbers of meetings scheduled are reduced as the duration of meetings increases. In all the cases, the scheduler with the combination of negotiation protocol and A*-Algorithm schedules more number of meetings than the other two combinations. The overall performance of the combinations of negotiation protocol and A*-Algorithm with respect to different duration of meetings is shown in Figure 13.

G. ***Calendars with Different Number of Slots*:** A set of simulations is carried out to estimate the performance of the DMS with local meetings moderately loaded. The values of the parameter are: numbers of slots of the calendar are 8, 16, and 24; number of organizations is 6; maximum duration of meetings is 2; and different sets with number of meetings are 1, 25, 50, 100, 200, 300, 400, and 500. The simulations with the above sets conducted for different combinations of negotiation protocol and A*-Algorithm are shown in Figures 14 to 17.

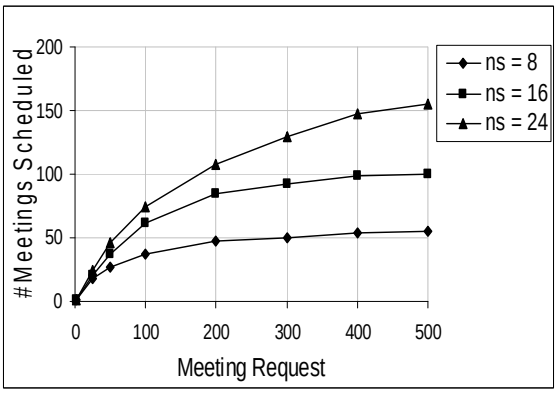


Figure 14. DMS without negotiation and without A*-Algorithm

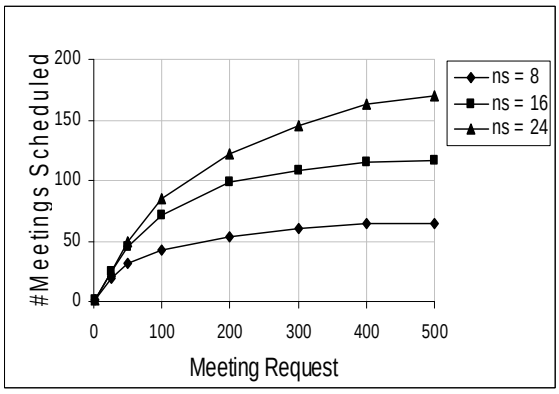


Figure 15. DMS with negotiation and without A*-Algorithm

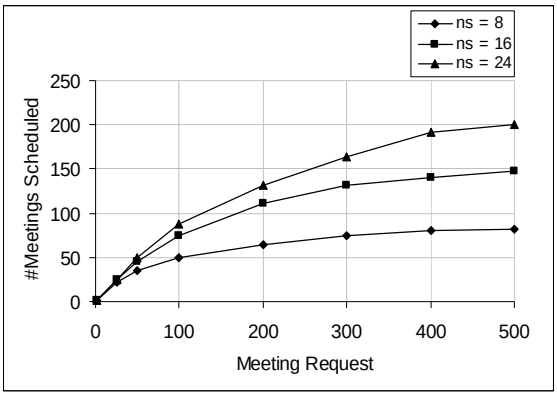


Figure 16. DMS with negotiation and A*-Algorithm

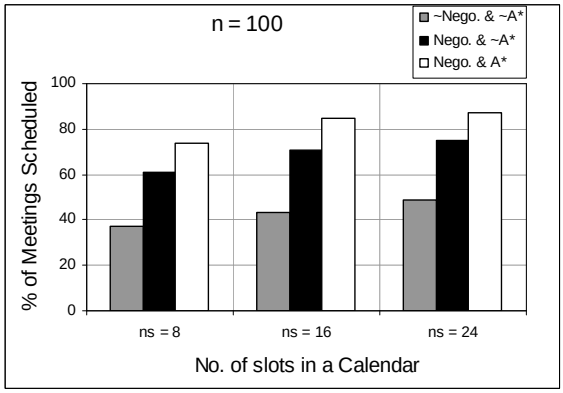


Figure 17. DMS with different numbers of time slots (8, 16, and 24)

It is clearly observed from the Figures 14 to 16, the number of meetings scheduled increases as the number of time slots increases irrespective of the approaches. The scheduler with negotiation and A*-Algorithm schedules more number of meetings than the other approaches monotonically. Figure 17 also shows that the percentage of meetings scheduled increases as the number of time slots in calendars increases. In all the three cases, the scheduler with the combination of negotiation and A*-Algorithm performs better than the other two.

VI. Conclusion

In this paper, an agent-based distributed meeting scheduler is proposed and implemented to consider equivalence classes of persons for delegation. The proposed scheduler schedules local meetings within the organizations and global meetings between organizations. The A*-Algorithm is used for searching and locally scheduling the meetings to have efficient schedules. Multi-stage negotiation protocol is also used to coordinate distributed agents for scheduling global meetings. In all the cases of simulations user preferences are taken into account. The performance of the scheduler is considered in terms of the number of meetings scheduled with respect to the number of meeting requests, and verified for four different categories such as increase in the number of meeting requests, number of time slots in the calendars, duration of meetings, and with the combination of A*-Algorithm and negotiation protocol. These categories of tests are carried out while the local schedules are lightly loaded, moderately loaded, heavily loaded, and also no local meetings. In all the cases, the scheduler produced the expected results. Combination of A*-Algorithm, equivalence classes of persons, and multi-stage negotiation protocol offers flexible, efficient and more practical schedules. The modified multi-stage negotiation protocol with A*-Algorithm and equivalence classes of persons performed better than having neither negotiation nor A*, and thus the number of proposals and processing time is considerably reduced.

Acknowledgement

The authors would like to thank the referees for their comments and suggestions to improve the quality of the paper.

References

[1] M. R. Gary, and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

[2] E. Taub, Sharing *Schedules. Mac User*, 1993, pp.155-162.

[3] S. Sen, and E. H. Durfee, A Formal Study of Distributed Meeting Scheduling: Preliminary Results. In Proceedings of the ACM Conference on Organizational Computing Systems, 1991, pp.55-68.

[4] S. Sen, and E. H. Durfee, A Formal Analysis of Communication and Commitment in Distributed Meeting Scheduler. In 11th International Workshop on Distributed Artificial Intelligence, 1992, pp.333-342.

[5] S. Sen, and E. H. Durfee, The Effects of Search Bias on Flexibility in Distributed Scheduling. In 12th International Workshop on Distributed Artificial Intelligence, 1993, pp.321-334.

[6] S. Sen, and E. H. Durfee, Adaptive Surrogate Agents. In 13th International Workshop on Distributed Artificial Intelligence, 1994, pp.320-333.

[7] S. Sen, and E. H. Durfee, Unsupervised Surrogate Agents and Search Bias Change in Flexible Distributed Scheduling. In Proceedings of the First International Conference on Multi-Agents Systems, 1995, pp.336-343.

[8] R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers, 1980, C-29, No. 12, pp.1104-1113.

[9] E. Ephrati, G. Zlotkin and J. S. Rosenschein. A Non-Manipulable Meeting Scheduling System. Thirteenth International Workshop on Distributed Artificial Intelligence, 1994, pp.105-125.

[10] K. Sycara, and J. Liu, Distributed Meeting Scheduling. In Proceedings of the Sixteenth Annual Conference of the Cognitive Society, 1994, pp.538-588.

[11] K. Sugihara, T. Kikuno and N. Yoshida. A Meeting Scheduler for Office Automation, IEEE Transactions on Software Engineering, 1989, Vol. 15, No. 10, pp.1141-1146.

[12] S. Sen. Developing an Automated Distributed Meeting Scheduler. IEEE Expert, 1997, pp.41-45.

[13] M. Sugumaran, and P. Narayanasamy, An Intelligent Multi-Agent Meeting Scheduler. In Proceedings of the International Conference on Artificial Intelligence in Engineering & Technology, 2002, pp.93-97.

[14] H. Nwana, L. Lee and Nick Jennings. Coordination in software Agent Systems, BT Technol J, Vol. 14, No. 4, 1996, pp.79-88.

[15] M. Sugumaran, K. S. Easwarakumar, and P. Narayanasamy. A New Approach for Meeting Scheduling using A*-Algorithm. In Proceedings of the IEEE TENCON International Conference on Convergent Technologies for Asia-Pacific Region, 2003, Vol. 1, 419-423.

[16] E. Rich and K. Knight, *Artificial Intelligence*, Tata McGraw-Hill, 1995.

[17] N. J. Nilsson, *Principles of Artificial Intelligence*, Narosa Publishing House, 1990.

[18] Yan Jin, Shu Lin, Qihua Situ, Xudong Wu, An Intelligent Meeting Scheduler. http://www.cs.ualberta.ca/~qihua/scheduler/report.html, 1999.

[19] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*. Second Edition, Pearson Education Series in Artificial intelligence, 2004.

[20] E. Crawford and M. Veloso, Opportunities for Learning in Multi-Agent Meeting Scheduling, http://www.mgci.memphis.edu, 2004.

M. Sugumaran received his M.Sc degree in mathematics from University of Madras during 1986 and M.Tech degree in computer science and data processing from Indian Institute of Technology, Kharagpur, India during 1991. He is currently working as Assistant Professor of Computer Science and Engineering at Pondicherry Engineering College, India. His areas of interests are theoretical computer science, analysis of algorithms, and parallel and distributed computing.

P. Narayanasamy is a Professor of Computer Science and Engineering in Anna University, India. He received his bachelor degree in electrical engineering during 1980 from University of Madras and master degree in electrical engineering during 1982 from Anna University. He received his Ph.D in computer science from Anna University during the year 1990. His current research includes computer network, mobile computing and wireless network.

K. S. Easwarakumar has completed his doctorate from Indian Institute of Technology, Madras, in the area of computer science and engineering during the year 1994. He has obtained his master's degree in computer and information sciences from Cochin University of Science and Technology during the year 1989. He is currently working as Professor of Computer Science and Engineering at Anna University, Chennai, India. His areas of interests are theoretical computer science, networks and molecular computing.