ISSN:0975-9646

M. Sugumaran / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (3) , 2011, 1001-1006

A Design of Centralized Meeting Scheduler with Distance Metrics

M. Sugumaran

Department of Computer Science and Engineering ,Pondicherry Engineering College, Puducherry, India.

Abstract— **Meeting scheduling is an important routine task in many organizations. The basic problem in meeting scheduling is to find a common free time for all participants of a meeting satisfying the given requirements. It is a time-consuming, iterative and tedious task. There exists several solutions for centralized calendar management and meeting scheduling, but they are of limited functionalities, not fully automatic and requests are static. Most of the schedulers not considered change of participants, the distance between the places of the participants and the venues of meetings. In this approach equivalent classes of persons, distance between the venue and places of participants with dynamic requests is considered to schedule meetings. This approach performs better and effective for practical meeting scheduling.**

***Index Terms*— Meeting Scheduling, A*-Algorithm, Artificial Intelligence, Meeting Constraints, Office Automation.**

I. INTRODUCTION

MEETING scheduling is a member of the class of resource scheduling problems that is known to be computationally intractable [1], and hence requires heuristics to reduce the computational effort. The timetable rearrangement problem in [1] is NP-Hard and has been shown that the feasibility problem is NP-Complete [2]. In general, all constraints are not considered in the processing of meeting scheduling. Some interesting approaches in Artificial Intelligence focus on learning user preferences [3, 4, 5]. Sen and Durfee [6] deal with persons' time as the primary resources. The meeting scheduling problem is focused to reduce the failures by utilizing the cooperation and rescheduling strategies when there is no common time-slot [7]. The scheduler [1] uses heuristic for the computation of $\rho(m_i)$ from $p(m_i)$ and generates all solutions with common pruning technique, then chooses the optimal one. The common pruning technique may not reduce the computation of some or many problem instances in finding an optimal solution by computing all solutions. So, the concept of A*-Algorithm [8, 9, 10] is applied to choose the best probable node or branch among the available nodes or branches in the search tree. This solution process obtains the optimal or near optimal solution quicker by selecting only one node (or branch) at each time of exploration. That is, most of the branches of the search tree are pruned, a solution is obtained quicker even for larger problem instances without considering all possibilities.

At any point of time, the participants may be on their way to attend a meeting, attending a meeting, returning to their hometown after attending a meeting or idle in their home stations. In day-to-day life, travelling time of the participants is to be considered along with the duration of meetings for an effective schedule. But the travelling time required for the participants before scheduling of meetings and minimum time required before cancelling of meetings so as to avoid the travel are not considered in the systems [1, 3, 4, 5, 6, 12, 13, 14].

In this work, a new centralized meeting scheduler with the distance metric, that is, the travelling time required for the participants from their current places to the venues of the meetings with respect to the time of requests along with the duration of meetings, has been designed and implemented, using A*-Algorithm with dynamic requests for scheduling and rescheduling of meetings. Equivalence classes of persons for substitution is used in A*-Algorithm to speed up the process and have flexibility in scheduling of more meetings.

II. PROBLEM SPECIFICATION

Let n be a positive integer. A *timetable* of n meetings is represented as 13-tuple $T(n) = (P, M_n, D_m, \square, h, t, p, w, a, d, f, \square, \square)$, where $P = \{1, 2, \ldots, m\}$ is a set of m persons; $M_n = \{m_1, m_2, \ldots, m_n\}$ is a set of n meetings; D_m is a 2-D distance matrix of the m persons; $<$ 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 such that exactly one person in each group is required to attend the meeting m_i; $w(m_i)$ is the weight (or priority) of the meeting m_i; $a(m_i)$ is the time of request (or arrival) of meeting m_i; $d(m_i)$ is the deadline at which meeting m_i is to be scheduled; $f(m_i)$ is the time at which meeting m_i is fixed; $\tau(m_i)$ is the starting time at which meeting m_i is scheduled; and $\rho(m_i)$ is the set of attendants of meeting m_i chosen from $p(m_i)$. Further, the schedule also satisfies the following conditions:

Let m_i and m_j be any two meetings.

- DM1: No person attends more than one meeting simultaneously.
- DM2: If $m_i < m_j$, then m_j starts after m_i ends, as per the need of organizational requirements and in case some persons attending both meetings m_i and m_j.
- DM3: For each group $g \in p(m_i)$, exactly one person can attend the meeting m_i.
- DM4: m_i can start at one of the time instances in $w(m_i)$.
- DM5: The venue of the meeting m_i is one of the participants' sites.
- DM6: Each meeting m_i should be scheduled at the earliest in one of the time intervals $(a(m_i) + \max \{\text{dist}(h(m_i), p(m_i))\}, d(m_i))$.
- DM7: Scheduling and rescheduling of any meeting m_i also should satisfy $d(m_i)$.

In a schedule $T(n)$, the parameters $\square$, t, p, a, d, D_m, h and w represent the input requirements of meetings, whereas $\square$ and $\square$ 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.

In this model, meetings are scheduled among a given set of persons. For any meeting, the person who proposes a meeting is called **host** of the meeting. The persons who are invited to attend that meeting are called **participants** or **invitees**. The invitees are considered for change of persons for most of the applications. At any point of time a participant may act as a host or an invitee of a meeting, but not both.

A. Example 1

Consider an example of timetable $T(5)$ of 5 meetings, shown in Fig. 1. That is, five meetings $\{m_1, \ldots, m_5\}$ are scheduled among eight persons $\{p_1, \ldots, p_8\}$ in a calendar of time intervals between 0 and 24. The eight persons are represented in the rows where as the time intervals are represented in the columns. Horizontal lines are used to represent the duration (or length) of meetings, and the names of the meetings written just above the horizontal lines. For example, the persons $\{p_2, p_8\}$ are assigned for m_3 in (3, 5) and $\{p_1, p_3, p_7\}$ are assigned for m_4 in (12, 15). For any person p_i, the interval between any two meetings represents the minimum time required to travel between the venues of two meetings.

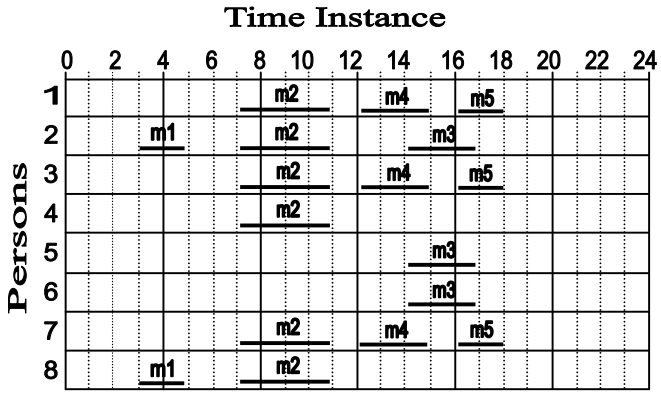

Fig. 1 Graphical Representation of $T(5)$ with DM

	1	2	3	4	5	6	7	8
1	0	1	2	3	4	3	2	1
2	1	0	1	2	3	4	3	2
3	2	1	0	1	2	3	4	3
4	3	2	1	0	1	2	3	4
5	4	3	2	1	0	1	2	3
6	3	4	3	2	1	0	1	2
7	2	3	4	3	2	1	0	1
8	1	2	3	4	3	2	1	0

Fig. 2 Weighted Distance Matrix of venues

The distance among the venues is shown in Fig. 2, and the corresponding input parameters for the Example 1 are given below.

$P = \{1, 2, 3, 4, 5, 6, 7, 8\}$.
$M_5 = \{m_1, m_2, m_3, m_4, m_5\}$.
$m_1 < m_5$ and $m_2 < m_4$.

$h(m_1) = 2$, $h(m_2) = 4$, $h(m_3) = 7$, $h(m_4) = 3$, & $h(m_5) = 4$.
$t(m_1) = 2$, $t(m_2) = 4$, $t(m_3) = 3$, $t(m_4) = 3$ & $t(m_5) = 2$.
$p(m_1) = \{\{2,3\}, \{7,8\}\}$, $p(m_2) = \{\{1\},\{2\},\{3\},\{4\},\{7\},\{8\}\}$,
$p(m_3) = \{\{2\},\{5\},\{6,7,8\}\}$, $p(m_4) = \{\{1,2\},\{3,4\},\{6,7,8\}\}$, &
$p(m_5) = \{\{1,2\},\{3,4\},\{6,7\}\}$.

$w(m_1) = 3$, $w(m_2) = 3$, $w(m_3) = 4$, $w(m_4) = 3$, & $w(m_5) = 5$.
$a(m_1) = 0$, $a(m_2) = 0$, $a(m_3) = 1$, $a(m_4) = 2$, & $a(m_5) = 2$.
$d(m_1) = 16$, $d(m_2) = 16$, $d(m_3) = 20$, $d(m_4) = 22$, & $d(m_5) = 24$.
$f(m_1) = 0$, $f(m_2) = 0$, $f(m_3) = 1$, $f(m_4) = 2$, & $f(m_5) = 2$.

$\tau(m_1) = 3$, $\tau(m_2) = 7$, $\tau(m_3) = 14$, $\tau(m_4) = 12$, & $\tau(m_5) = 16$.
$\rho(m_1) = \{2, 8\}$, $\rho(m_2) = \{1, 2, 3, 4, 7, 8\}$, $\rho(m_3) = \{2, 5, 7\}$,
$\rho(m_4) = \{1, 3, 6\}$, & $\rho(m_5) = \{1, 3, 6\}$.

The timetable rearrangement problem is done dynamically. That is, as soon as a meeting request comes, the scheduler tries to schedule the meeting by considering the travelling distance between the host of the meeting request and the current places of the participants with respect to the time of request. For instance, the input parameters required for the new timetable $T(n+1)$ by rearrangement is defined as

I1: A timetable $T(n) = (P, M_n, D_m, <, h, t, p, w, a, d, f, \tau, \rho)$.
I2: A partial order $<$ on $M_{n+1} = M_n \cup \{m_{n+1}\}$.
I3: $t(m_{n+1})$, $p(m_{n+1})$, $w(m_{n+1})$, $a(m_{n+1})$, and $d(m_{n+1})$.
I4: A set F ($\subseteq M_n$) of meetings whose start times are fixed. The meetings that have higher priorities than m_{n+1} are kept in the set F.

B. Example 2

Consider an instance to schedule a new meeting m_6 in Example1 with the following input parameters.

I1: $T(5)$ as in Example 1, shown in Fig. 1.
I2: $m_1 < m_5$, and $m_2 < m_4$.
I3: $t(m_6) = 3$, $p(m_6) = \{\{1\},\{3, 4\},\{6\}\}$, $h(m_6) = 6$, $a(m_6) = 6$, $w(m_6) = 1$, & $d(m_6) = 16$.
I4: $F = \{\}$.

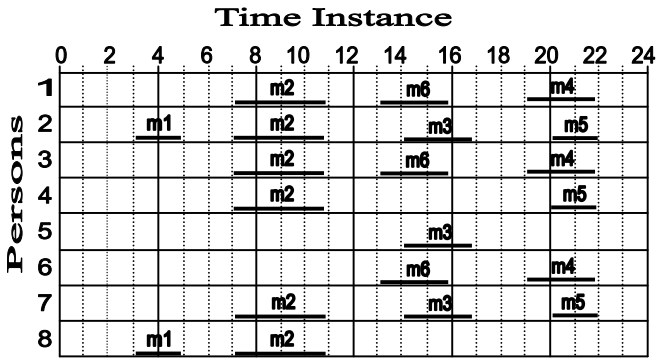

Fig. 3 Graphical Representation of $T(6)$ with DM

Fig. 3 shows an optimum timetable $T(6) = (P, M_6, D_m, <, h, t,$

p, w, a, d, f, τ', ρ') by rescheduling of meetings in $T(5)$ with a set of operations given in the next section, where $\tau'(m_4) = 19$, $\rho'(m_5) = \{2, 4, 7\}$, $\tau(m_6) = 13$ and $\rho(m_6) = \{1, 3, 6\}$. The fixing of meeting m_6 by rescheduling is explained in Section 4.

III. CENTRALIZED MEETING SCHEDULER WITH DISTANCE METRICS

In this model, six operations such as CP (Change of Person), XP (eXchange of Person), SL (Shift Left), SR (Shift Right), CSL (Continuous Shift Left) and CSR (Continuous Shift Right) are used to rearrange the scheduled meetings so as to schedule the current meeting request. The rescheduling of meetings takes place if no free slots are available. Based on these operations, heuristic value is determined. The heuristic value will be the deciding factor for selecting the next node in the search tree expansion. The method of A*-Algorithm is applied with this heuristic value for the generation of search tree. This process will speed up the searching optimally.

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 $\rho(m_{n+1})$ would be the set of attendees for m_{n+1}, that is a set of candidates to be chosen from $p(m_{n+1})$ one person per group. Further, let $(x, x + t(m_{n+1}))$ be the current interval considered for scheduling the meeting m_{n+1}. These assumptions are used in the illustration of six operations, which are given in the following section.

A. Rescheduling of Meetings with Distance Metrics

Consider Fig. 1 for the illustration of the rescheduling operations. Let $a(m_{n+1})$ and $d(m_{n+1})$ be the arrival and deadline of meeting m_{n+1}. The earliest feasible time for scheduling the meeting m_{n+1} (i.e., the function, $eft(m_{n+1})$) and its feasible period are shown in Fig. 4. If free slots for all persons of m_{n+1} is available in the feasible period, then the meeting m_{n+1} is scheduled, otherwise rescheduling operations for those meetings with the required persons in the feasible period will take place.

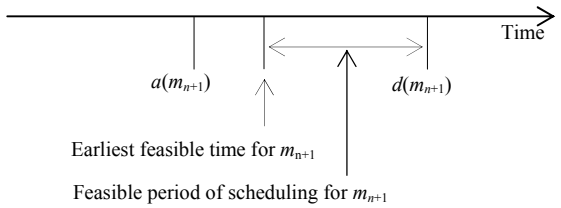

Fig. 4 Feasibility of scheduling meeting m_{n+1}

When a new meeting, say m_{n+1}, is scheduled, if none of the slots is free in the feasible period of m_{n+1}, then the set of six operations (CP, XP, SL, SR, CSL and CSR), at a time only one operation, is applied to reschedule the scheduled meetings out of the feasible period, and then the meeting m_{n+1} is scheduled. When these operations are applied for the generation of nodes of the search tree, the feasibility of scheduling meeting is verified with respect to the time of rescheduling. The following assumptions are considered for rescheduling of meetings:

- The processing time and communication time of events are not taken into account.
- To cancel a meeting m_i, it is required that the cancellation message should reach the attendants of m_i, $\rho(m_i)$, before they start for venue of m_i, otherwise the meeting m_i is not cancelled.

SL: Shifts a meeting horizontally to the left. For each person $p_j \in \rho(m_{n+1})$ and each meeting $m_i \in (x, x + t(m_{n+1}))$, if there is a free time interval $(s, s + t(m_i))$ to the left of $(x, x + t(m_{n+1}))$ such that $s \geq eft(m_i)$ with respect to the time at which the operation SL is applied, no person in $\rho(m_i)$ attends any meeting during $(s, s + t(m_i))$, and there is enough time for the message about the change to reach $\rho(m_i)$ before they leave for venue of m_i, then send the message to $\rho(m_i)$ and change the start time of m_i to s.

SR: Shifts a meeting horizontally to the right. For each person $p_j \in \rho(m_{n+1})$ and each meeting $m_i \in (x, x + t(m_{n+1}))$, if there is a free time interval $(s, s + t(m_i))$ to the right of $(x, x + t(m_{n+1}))$ such that $s + t(m_i) \leq d(m_i)$ with respect to the time at which the operation SR is applied, no person in $\rho(m_i)$ attends any meeting during $(s, s + t(m_i))$, and there is enough time for the message about the change to reach $\rho(m_i)$ before they leave for venue of m_i, then send the message to $\rho(m_i)$ and change the start time of m_i to s.

Consider Fig. 1 for the illustration of SR operation. For example, let us consider the SR of meeting m_3 so as to make free slots for scheduling m_6 in (14, 17). The meeting m_3 is shifted to the right for the persons in $\rho(m_3) = \{2, 5, 6\}$ from the time instance 14 to 17. Now, the number of slots available for m_6 is increased from one to two, shown in Fig. 5.

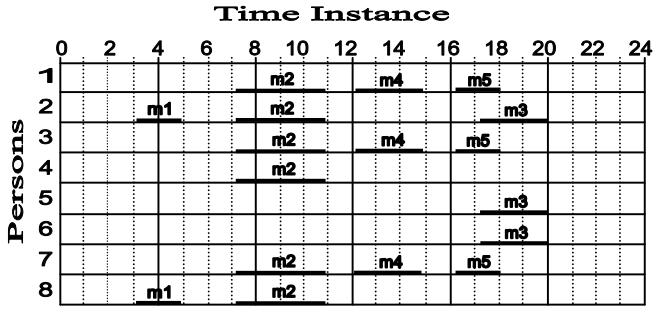

Fig. 5 SR on m_3 with DM

CP: Changes person vertically within a group. This operation makes free slots for the persons $\rho(m_{n+1})$ in $(x, x + t(m_{n+1}))$ by changing one person with another within the respective groups. For each meeting $m_i \in (x, x + t(m_{n+1}))$, if no person in $\rho(m_i)$ except some person $p_j \in \rho(m_i)$ attends m_i and if there is a person p_k in $\{a \mid a, p_j \in g_r$ and $g_r \in p(m_i), p_k \neq p_j\}$ who does not attend any meeting during $(\tau(m_i), \tau(m_i) + t(m_i))$, and both p_j and p_k satisfy the time constraints before and after the current meeting, then change the attendant p_j of m_i to p_k. The

operation CP m_3 p_6 p_8, changes person p_6 with person p_8 for the meeting m_3, is shown in Fig. 6.

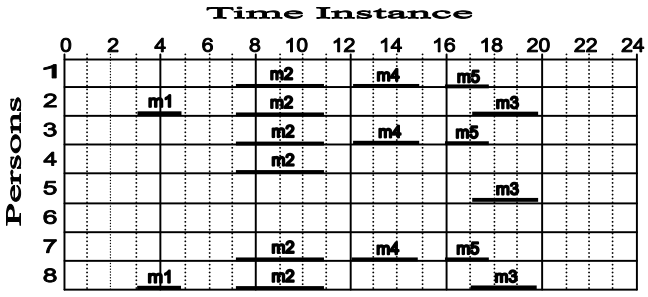

Fig. 6 CP on m_3 with DM

XP: Exchanges persons within a group on different meetings. This makes free slots for the current meeting by making changes diagonally. That is, for any two persons p_j, $p_k \in g_p$ in $p(m_i)$ and p_j, $p_k \in g_q$ in $p(m_n)$ such that $p_j \in \rho(m_i)$ and $p_k \in \rho(m_n)$, then swap p_j and p_k between $\rho(m_i)$ and $\rho(m_n)$ if p_j is free in $(x, x + t(m_n))$, p_k is free in $(x, x + t(m_i))$ and both p_j and p_k satisfy the constraints after the previous meeting and before the next meetings.

CSL (Continuous Shift Left): Shifts the meetings in $(x, x + t(m_{n+1}))$ horizontally to the left in order. For each person $p_j \in \rho(m_{n+1})$ 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 $s \geq eft(m_i)$, no person in $\rho(m_i)$ attends any meeting during $(s, s + t(m_i))$, the message about the change to reach all of $\rho(m_i)$ before they leave for venue of m_i and shifting the meetings in $(s, s + t(m_i))$ to the left which are satisfying the constraints, then change the start time of m_i to s and send the message to $\rho(m_i)$.

CSR (Continuous Shift Right): Shifts the meetings in $(x, x + t(m_{n+1}))$ horizontally to the right in order. For each person $p_j \in \rho(m_{n+1})$ 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 $s + t(m_i) \leq d(m_i)$, no person in $\rho(m_i)$ attends any meeting during $(s, s + t(m_i))$, the message about the change to reach all of $\rho(m_i)$ before they leave for venue of m_i and shifting the meetings in $(s, s + t(m_i))$ to the right which are satisfying the constraints, then change the start time of m_i to s and send the message to $\rho(m_i)$. The CSR of meetings m_4 and m_5 with reference to Fig. 1 is shown in Fig. 7.

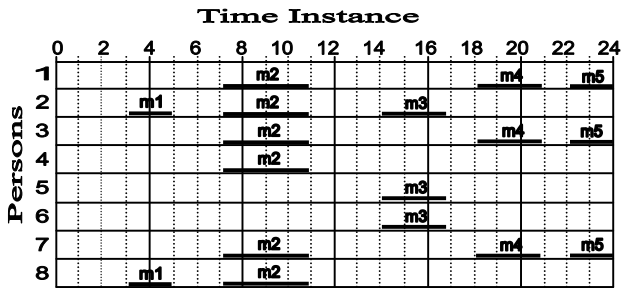

Fig. 7 CSR of m_4 and m_5 with DM

IV. A*-ALGORITHM WITH DISTANCE METRICS

Scheduling of meetings in centralized meeting scheduling with distance metrics is done dynamically. That is, when a meeting request arrives, the scheduler tries to schedule the meeting by satisfying the conditions along with the traveling distance between the host of the meeting request and the participants' current positions is considered with respect to the time of request. When a new meeting, say m_{n+1}, is scheduled, if none of the slots is free in the feasible period, then the set of six operations is applied for generation of nodes of search tree to reschedule the scheduled meetings and then the new meeting m_{n+1} is fixed. When these operations are applied at the time of generation, the feasibility of scheduling meeting with respect to the time of request is verified. A*-Algorithm with rescheduling operations for the scheduler with distance metric, is given below.

***Algorithm* Schedule**(start-time(m_{n+1}), deadline(m_{n+1}),$p(m_{n+1})$)
//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 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 terminates.
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

A. Example 3

As an example of meeting scheduling with distance metric, scheduling of a new meeting, m_6, may be considered with the root configuration, which is shown in Fig. 1. To schedule the meeting m_6, A*-Algorithm is applied with distance metric to the root configuration, $T(5)$.

The request of m_6 arrives at the time instance 6, so the earliest time at which m_6 could be scheduled is calculated using the function $eTimeIns(m_i)$:

$eTimeIns(m_6) = a(m_6) + \max \{\min \{dist(h(m_6), p_1)\}, \min \{dist(h(m_6), p_3), dist(h(m_6), p_4)\}, \min \{dist(h(m_6), p_6)\}\}$
$= 6 + \max \{3, \min \{3, 2\}, 0\}$
$= 9.$

This means that m_6 could be scheduled at the earliest time instance 9. Since the dead line of the meeting m_6, $d(m_6) = 16$, the duration (9, 16) can be considered feasible duration for scheduling m_6. That is, m_6 could be scheduled in one of the time instances {9, 10, …, 16}. After computing the feasible duration of a meeting, the scheduler has three choices in terms of increasing complexities to proceed further. That is, it has to look for (a) free time slots for all participants of the new meeting; (b) reschedule the scheduled meetings so as to find free time slots for the new meeting, or (c) cancel one or more meetings so as to schedule the new meeting. Generally, it depends on the meeting request and the current status of the calendar.

Suppose that the scheduler decides to schedule the current meeting request m_6 at the earliest time instance of the feasible duration. In this case, the scheduler has to try from the time instance 9 one after another. Since all of the participants of the meeting m_6 are not free in the time interval (9, 12) and m_2 is already scheduled in that duration, the scheduler may decide to cancel the meeting m_2 and schedule m_6. To cancel the meeting m_2 at the time instance 6, $\rho(m_2)$ should not have left for $h(m_2)$ before the time instance 6 from their current positions. That is, if at least one of them in $\rho(m_2)$ left before the time instance 6, the meeting m_2 could not be cancelled.

The earliest leaving time of $\rho(m_2)$ for m_2
= min {$start(m_2)$ – min {$dist$ (current place of p_i, $h(m_2)$), $dist$(current place of p_i, $home(p_i)$ + $dist(home(p_i)$, $h(m_2)$)), for $p(m_2)$ = {1, 2, 3, 4, 7, 8}}}
= min {7 – {3, 2, 1, 0, 3, 2}} = 4.

That is, the earliest time of leaving for $\rho(m_2)$ to m_2 is 4, at least one of $\rho(m_2)$ would have left at the time instance 4. So the meeting m_2 is not cancelled to schedule the meeting m_6 and hence the scheduler decides to schedule m_6 after m_2, that is, at the time instance 11 onwards. Since $p(m_6)$ = {{1}, {3, 4}, {6}}, the possible start-time for m_6 after m_2 is
max {11 + $dist(h(m_2), h(m_6))$, min{11 + $dist(h(m_2), h(m_6))$, 11 + $dist(h(m_2), h(m_6))$}, 0 + $dist(home(m_6), h(m_6))$}
= max {11 + 2, min {12, 12}, 0}
= 13.

So, the required time interval for scheduling m_6 is reduced to (13, 16) from (9, 16). For this time interval, there are four slots {(13, 16), (14, 17), (15, 18), and (16, 19)} of length three. For these four slots, the root node with four branches for scheduling of m_6 is given in Fig. 8, and the search tree is shown in Fig. 9. The node numbers are given at the left of the nodes. The heuristic values of the nodes (or branches) and their costs are given as (heuristic-value:cost) at the right side of each node. All four branches have the same heuristic value, 1. Here any branch can be taken for expansion of the search tree. The branch could be the first one or any one by random selection. Assume that the first slot, (13, 16) is chosen for expansion. The edges of the search tree are provided with the operations applied in A*-Algorithm. The node-id of the root is 0 and the expanded nodes are given node-ids from 1 onwards.

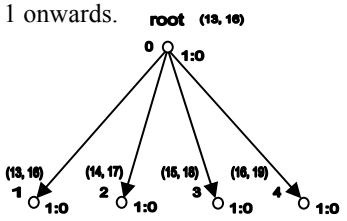

Fig. 8 Possibilities of slot (13, 16)

The operations applied for rescheduling of meetings can be divided into two sets. They are **relevant-set** and **common-set** of operations. Relevant-set operations are applied to the meetings and the participants of the meetings that are scheduled in a slot where the current request of a meeting is to be scheduled. The relevant-set operations increase the heuristic values of the children rather than the parent. Sometimes the relevant set operations may reduce the heuristic value of the children but increase the heuristic value of the grand children and their children and so on. The common-set includes all operations, that is, relevant-set and other operations applied in a search tree. For example, the relevant-set operations can be applied for shorter solution path, minimum cost path, solutions with minimum number of nodes generation and solutions to be obtained with simple operations.

Fig. 9 shows that there are 14 nodes in the search tree. The nodes are given numbers according to the levels. The root node is explored with the common-set operations. Six nodes are generated in the level one, numbered 1 to 6. The nodes with node-ids 1, 4 and 6 have heuristic value 2. Now the scheduler has to decide which node to choose to expand the search tree further. Since all three nodes have the same heuristic value, the scheduler selects the node 1 as the current node for expansion, as it has the minimum cost.

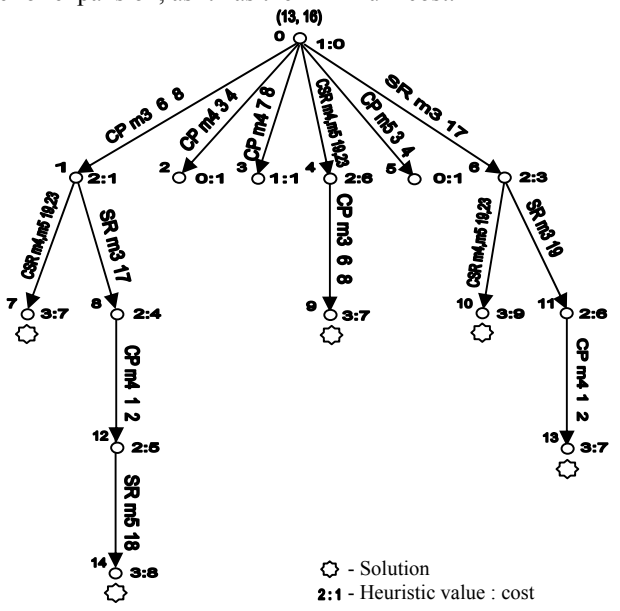

Fig. 9 Search tree for m_6 with DM

When node 1 is explored, two nodes with node-ids 7 and 8 are generated in the second level. As the node 7 is the solution node, the expansion of the search tree is terminated. So, eight nodes are generated in Fig. 9 when the common-set of operation is applied.

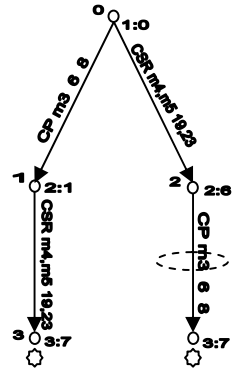

Fig. 10 Search tree for m_6 with relevant set of operations

Fig. 10 shows a search tree for the same problem as with Fig. 9 considering the relevant-set operations. At the level 1, nodes 1 and 2 are generated from the root node 0. Both nodes have heuristic value 2, but the cost of the nodes is different. At this point, A*-Algorithm chooses node 1 for expansion, and generates node 3. Since node 3 is a solution node, the process is terminated. So, in this search tree only three nodes are generated, two nodes at the first level and one node in the second level. Having less number of nodes generated, the processing time as well as the overhead involved in the generation of the search tree is reduced. Thus the relevant-set operations save a considerable amount of memory as well processing time, compared to the common-set of operations.

V. RESULTS

The simulation of the meeting scheduler with distance metrics using A*-Algorithm was carried out and the performance was compared by considering various combinations of parameters from different sets like distance matrix, number of slots in a calendar, number of persons, maximum number of groups, maximum number of persons in groups, maximum duration of meetings, and maximum number of time instances (that is, number of time instances between the arrival time and the deadline of the meeting request).

It was observed that the participants of a meeting are required not only be available during the time of meeting $t(m_i)$ but also for the time of travel from their current place to the venue of that particular meeting (t_{tra}). So, the time required for a meeting m_i from the participants' view is that $t_{tra} + t(m_i)$. Since $t_{tra} + t(m_i) \geq t(m_i)$ for all m_i, the number of meetings scheduled with distance metrics is less than scheduled meetings without distance metric. Further, for any meeting m_i, the constraints before and after m_i are to be checked for all $\rho(m_i)$, so, the processing time of the scheduler with distance metric is more than that of without distance metrics.

VI. CONCLUSION

Centralized meeting scheduling with distance metrics satisfies the dynamic requests of meeting and timing requirements before scheduling and cancellation of meetings with A*-Algorithm. This approach for the meeting scheduling with distance metrics using A*-Algorithm gives solutions with less memory requirements and processing time as the scheduler chooses only one node or branch at each level of the search tree during expansion. The number of meetings scheduled with distance metrics is less than that of scheduling of meetings without distance metrics. Further, the overheads involved in the processing time of the scheduler with distance metrics are little more than that of without distance metrics.

REFERENCES

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

[2] Garey M.R. and Johnson D.S. (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness", San Francisco, CA: Freeman.

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

[4] Maes P. (1994), "Agents that Reduce Work and Information Overload", *Communications of the ACM*, Vol.37, No.7, pp. 31-40.

[5] Mitchell T., Caruana R., Dayne F., McDermott J. and Zabowski D. (1994), "Experience with a Learning Personal Assistant", *Communications of the ACM*, Vol.37, No.7, pp. 80-91.

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

[7] Jeong W.S., Yun J.S. and Jo G.S. (1999), "Cooperation in Multi-Agent System for Meeting Scheduling", *Proceedings of the IEEE TENCON*, pp. 832-835.

[8] Nilsson N.J. (1990), "Principles of Artificial Intelligence", Narosa Publishing House

[9] Rich E. and Knight K (1995), "Artificial Intelligence", Tata McGraw-Hill.

[10] Russell S.J. and Norvig P. (2004), "Artificial Intelligence - A Modern Approach", Second Edition, Pearson Education Series in Artificial Intelligence.

[11] Ashir A., Joo K.H., Kinoshita T. and Shirotori N, (1997), "Multi-Agent Based Decision Mechanism for Distributed Meeting Scheduling System", *Proceedings of the International Conference on Parallel and Distributed Systems*, pp. 275-280.

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

[13] Garrido L. and Sycara K. (1996), "Multi-Agent Meeting Scheduling: Preliminary Experimental Results", *Proceedings of the First International Conference on Multi-Agent Systems*, pp. 95-102.

[14] Sen S. (1997), "Developing an Automated Distributed Meeting Scheduler", *IEEE Expert*, July/August, pp. 41-45.

M. Sugumaran received his M.Sc degree in mathematics from University of Madras in 1986, M.Tech degree in computer science and data processing from Indian Institute of Technology, Kharagpur, in 1991 and Ph.D in computer science and engineering from Anna University, Chennai in 2008. He is a life member of ISTE and CSI. He is currently working as Associate Professor in the Department of Computer Science and Engineering at Pondicherry Engineering College. His areas of interests include theoretical computer science, analysis of algorithms, parallel and distributed computing, spatial-temporal data and information security.