International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Meeting Scheduling Application using Availability and Priority Attendance Heuristics

Aswin Wibisurya, Ikhtiar Faahakhododo

Abstract: ***Meeting scheduling is a repetitive and time consuming task for many organizations. Emails and electronic calendars has been used to help a meeting host in this process. However, it does not automate the process of searching the optimal time slot. Manual scheduling may result in suboptimal schedule. Therefore, automation is needed for meeting scheduling problem. The purpose of this research is to propose an applied model consisting of both acquiring participants' existing schedule, and searching for an optimal time slot. Previous studies groups the solution of meeting scheduling into either constraint satisfaction or heuristics approach. Heuristics is more appropriate for a dynamic environment. The heuristics-based model is designed to consider participant availability and participant prioritization. The more participants are available, the better the time is as a candidate for optimum schedule. In the proposed model, the availability of certain key person, experts, or host may carry more weight than normal participant. An Android based application is developed as a prove of concept of the proposed model. Google Calendar API is used in this model to acquire the existing schedule, then each time slot is assigned a score based on availability weighting. The time slot with the highest score is considered the optimal solution. Evaluation is done by simulating the scheduling part for various numbers of meetings and time slots. The result shows that the model is capable of searching the optimal meeting schedule in less than one second for each of the experiment.***

Keywords: ***Meeting scheduling problem, agent-based, heuristics, mobile application.***

I. INTRODUCTION

Meeting scheduling has long been considered repetitive and time consuming for many organizations. People need to communicate with each other to ask for available time, invite for a meeting, propose another time for the meeting, reschedule, or even cancel a meeting. Each person has their own constraint of available time and preferences, which makes it even harder to find a commonly available time.

Technology has existed to address this problem, e.g. electronic invitation and calendar sharing. Though computer-assisted, an admin or host still needs to ask for participants' availability, and manually search for a schedule. Not all possibilities of the meeting time is are considered with regard to each persons' availability, which may result in suboptimal solution. Therefore, automation is needed for meeting scheduling problem.

Several researches have been done in meeting scheduling problem. They either propose a theoretical model to calculate the optimal meeting schedule [1]–[4], or the architecture of a system to acquire the existing schedule and negotiation for a new schedule [5]–[7]. Very few deal with them both, which are both required to create an applied solution.

The purpose of this research is to propose an applied model of agent-based meeting scheduling using availability and prioritized attendance. The proposed model is a complete solution from determining the range of the time period in which a time slot should be placed, determining meeting participants, acquiring participants existing schedule, and recommendation of the most optimal time slots for the meeting schedule.

In this research, participant's availability is the main factor to determine the optimal schedule. The more participants are available, the better the time is as a candidate for optimum schedule. However, not all participants are considered equal. The host, experts, or key person may be more prioritized to attend. Therefore, prioritization of specific persons to be available is also considered when calculating the score of a time slot to be chosen.

Since availability is an important factor in the proposed model, if some participants' availability is unknown, the system may propose a suboptimal schedule. It is important to find an approach to acquire participants' availability before scheduling the meeting.

An Android based application is developed as a prove of concept of the proposed model. Google Calendar API is used because of its ubiquity in mobile environment.

The proposed model is limited to search only for existing known schedules of participants mentioned in his online calendar. Participants' existing but not mentioned will not be considered and will result in conflicting schedule. Therefore, adding appointment to calendar should be enforced when implementing this model.

Evaluation to this model is done by simulating to schedule a meeting in a range of date with existing schedules for participants and measuring the time needed to generate it.

Revised Manuscript Received on December 30, 2019.

* Correspondence Author

Aswin Wibisurya*, Faculty Member, Mobile Application and Technology Program, Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia. E-mail: aswinwibisurya@binus.ac.id

Ikhtiar Faahakhododo, Faculty Member, Mobile Application and Technology Program, Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia. E-mail: ifaahakhododo@binus.edu

II. LITERATURE REVIEW

Previous studies have been done on meeting scheduling automation. These solutions to the problem can be categorized into two [5]. The first is constraint satisfaction. In this approach, multiple meetings are mapped into previously empty time slots. Each meeting has several constraints, e.g. availability or preferences of participants. The goal is to map these meetings into empty slots while minimizing the violation of constraints [1]. This approach is not suitable for dynamic environment, in which the list of meetings in a period is not predetermined. A meeting suddenly needs to be scheduled in between other agendas, which cannot be rescheduled. The second category is the agent approach, in which several agents representing meeting participants mention their existing schedules. The host agent then follow a certain heuristics to generate an optimal schedule. Heuristics is used to find a satisfactory solution which is considered optimal, finding the real optimal solution might need a complex computation [8]. The goal is to maximize a certain score function [3]. The variable to be scored in a heuristics can be participants' preference rate [6], participants' availability (minimizing conflicting schedule) [2], [9], room availability[8], or energy efficiency [3], [4]. This heuristics approach is more common in a dynamic environment. Therefore heuristics is chosen in our research.

BenHassine and Ho [6] proposed a model based on participants' preference. Participants need to state their preferences for all time slots, giving a value between 1 (willing to join) and 0 (not willing). Preference is used instead of availability, since someone may be available to attend but chooses not to. However, user input is needed for each possible time slots. We consider the need for user input too much to be used in our proposed model.

There is another heuristics to meeting scheduling based on preference by Mussawar and Al-Wahedi [5]. The approach also assigns participants' preference value for each time slot. Negotiations are performed on multiple rounds, since a participant's preference may change after some rounds of negotiation. To ensure convergence, the weight of participant preference is decreased for each round. The number of user input for participant in this case is the number of possible time slots, multiplied by the number of rounds of negotiation.

Glitho, et. al. created a mobile agent capable of scheduling a meeting [7]. Participants' existing schedule is fetched via an email API, then used to determine participants' availability, similar to our proposed research. Notification is also used to gather participant feedback of attending or not. Glitho et. al. stopped at good enough solution, that is a schedule is decided when there is enough participant attending the meeting.

Refanidis and Alexiadis used Google API to obtain existing user schedule [10]. In the research, constraint satisfaction approach is used to schedule meeting for individual. The proposed model can only be used for individual schedule, not for group meeting.

Hosein and Boodhoo divided constraints for meeting scheduling problem into hard constraints and soft constraints. Hard constraints are constraints that must be applied. If one said to be busy, then the slot cannot be used. Meanwhile, soft constraints can be broken. For example, personal agenda can be cancelled to attend a meeting instead [11]. The idea of hard and soft constraints is implemented in our framework with different set of constraints.

Borg [9] considered availability as a hard constraint, meaning all participants need to be available for a meeting to be scheduled in a particular time. The algorithm used in the research is round robin. The use of availability as hard constraint is considered inflexible in our approach, and we choose to use it as soft constraint instead.

III. PROPOSED METHODOLOGY

A. System Workflow

From the meeting host's point of view, the create meeting workflow is shown in Fig. 1. Screenshots of the mobile app and the explanation of the workflow is explained below.

The developed system runs on Android platform and heavily relies on Google APIs. Before the host can use the app, she needs to sign in using a Google account, so the system can acquire refresh token and access token needed to fetch the host's schedule and contacts.

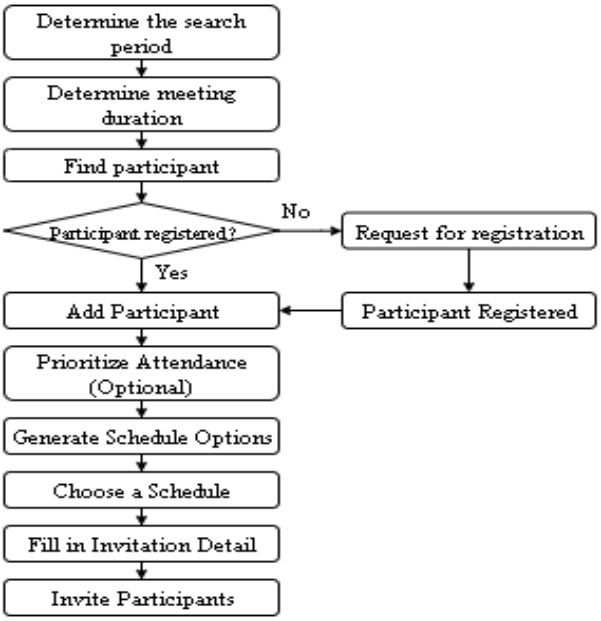

Fig. 1.Workflow of the Meeting Scheduler Model

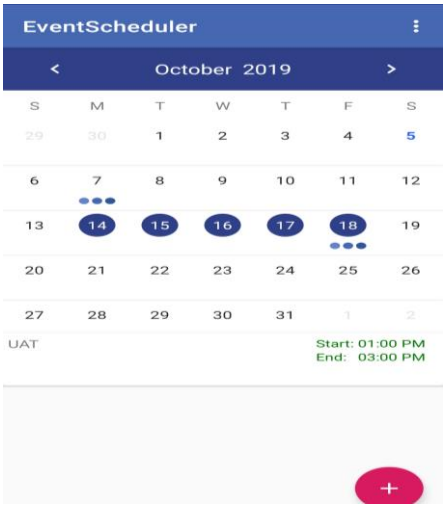

Fig. 2.Home Page

1. Home Page

After signing in, the host should input the start and end date of the period the meeting should fall in between (search period). This is done in the home page (Fig. 2). The three dots icon below the dates shows existing meetings already in schedule of the host. The host can specify the search period by clicking the calendar on the start and end date. Then, the host can create the meeting using the plus button on the lower right.

2. Create Meeting Page

The meeting host can input the parameters needed to automatically generate the schedule options. The start and end date fields are editable, but pre-populated using the search period input from the home page. The meeting duration (in hours) should also be specified. Participants are added by clicking on the field. Fig. 3 shows the create meeting page.

3. Add Participants

Participants are added by searching their names or emails. It is possible that invited participants have not yet registered to the system, so the system cannot fetch their schedules. An option to ask the participant to register is provided. The participant will receive an email along with a URL asking him to register. Once participants are determined, the host clicks the check button on the top right. Fig. 4 shows the user interface of adding participants.

4. Prioritization

The host can prioritize some participants' availability to make sure they can attend. Upon checking the prioritize button, two sections will be displayed. Attendees are categorized into high priority and low priority via drag and drop into the designated section. This prioritization is shown in Fig. 5. The score for available time slots for high priority participants will get multiplied by 2 on the scoring algorithm discussed in the next section

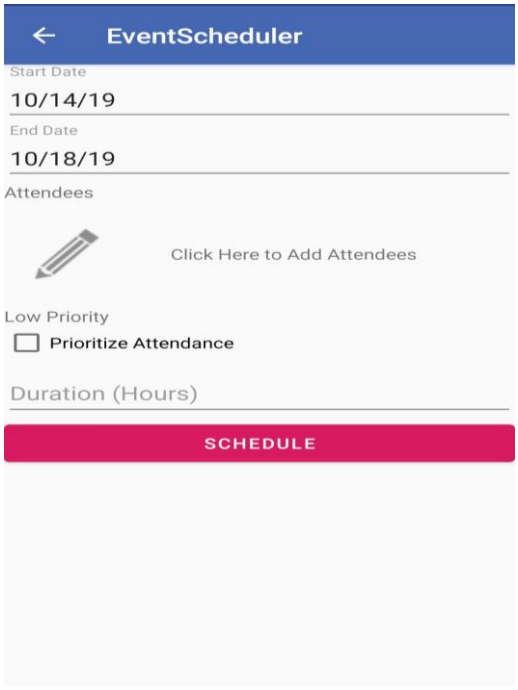

Fig. 3.Create Meeting Page

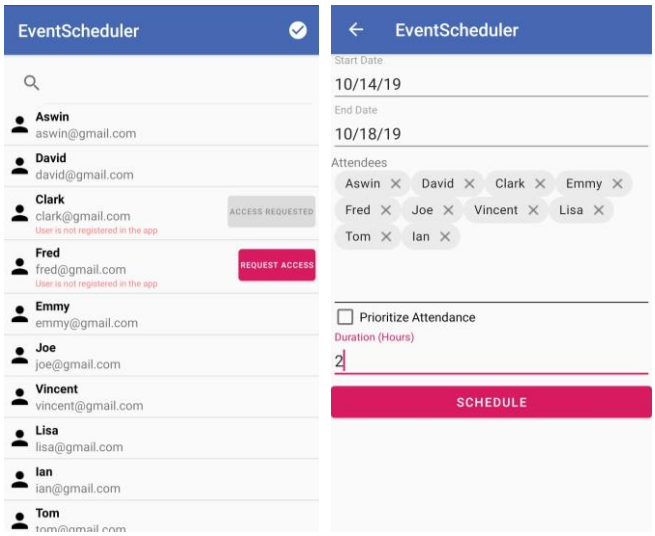

Fig. 4.Add Participants

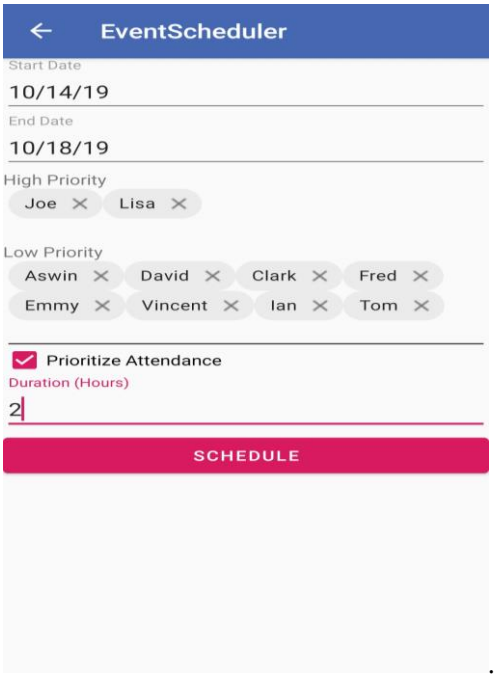

Fig. 5.Participant Prioritiaztion

5. Schedule Generation

After finishing the required fields, the host can click the schedule button. The system will calculate the score for each slot according to participants' availability and the prioritization. The algorithm to calculate these scores is discussed in later section. Three slots with the best scores are displayed as options. The display can be viewed in Fig. 6.

6. Send Invitation

After choosing a schedule option, the host fills in the detail of meeting to be sent in an invitation. The host must fill in the location, summary, and description of the meeting. This can be seen in Fig. 7. Upon clicking the button submit, an invitation email will be sent to all participants. Participants which are not available at the proposed time are still getting the invitation, since it is still possible the participant will change the agenda later.

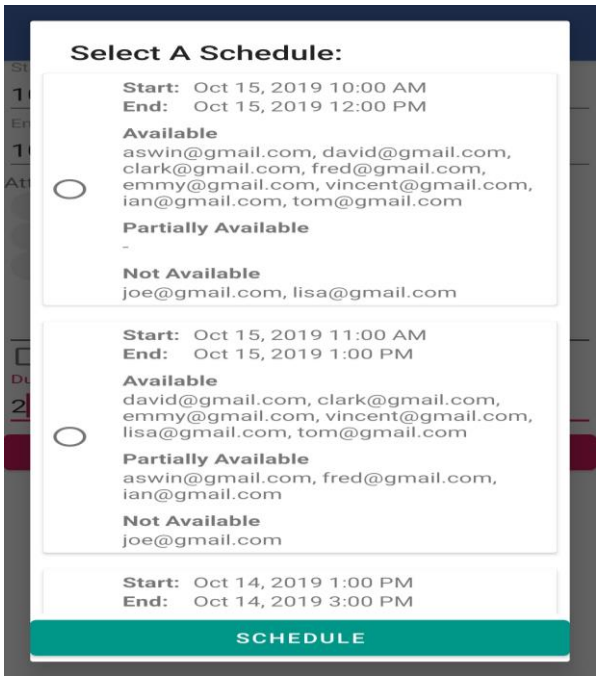

Fig. 6.Schedule Options Pop Up

.

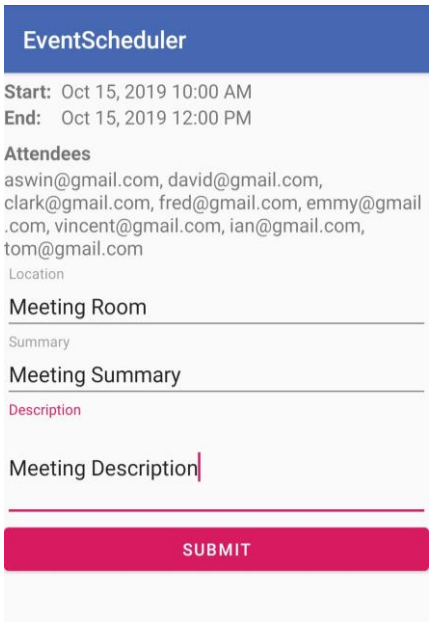

Fig. 7.Send Invitation

B. Meeting Scheduling Algorithm

The proposed algorithm consists of generating the availability matrix, from which it can be known if a person is available in a specific time; and generating the matrix according to the availability heuristics. The optimum time for the meeting is determined by the time corresponding to the slot with maximum score.

The pseudocode of the proposed approach is the following:

```
let P = number of participants
let S, E = start time, end time of the search period
let N = number of working hours between S and E
let H = meeting duration in hours
let slots = 2D matrix of zero values with size P * N
for each person p
  for each meeting m
    let s, e = start time, end time of m
    if e > E then
      e = E
    end if
    let startSlot = number of working hours between S and s
    let endSlot = number of working hours between S and e
    for i = s to e
      increment slots[p][i]
    end for
  end for
end for
let D = number of working days from S to E
let slotsT = transpose slots
let dailySlots = group slotsT per day, so that the dimension of dailySlots is: D * slots per day * P
let scores = 3D matrix of zero values with size D * slots per day * P
let totalScores = 2D matrix of zero values with size D * slots per day
for i = 0 to D - 1
  let ub = last index of dailySlots[i]
  for each person p
    let count = 1
    for j = ub down to 0
      if dailySlots[i][j][p] = 0
        if p is high priority participant
          scores[i][j][p] = 2 * count
        else
          scores[i][j][p] = count
        end if
        totalScores[i][j] = totalScores[i][j] + count
        if count < H
          increment count by 1
        end if
      else
        count = 1
      end if
      dailySlots[i][j][p]
    end for
  end for
end for
```

Priority scheduling is done by adding a multiplication factor of 2. The score for the high priority participant for his available time slot is two times the score for regular a participant.

The totalScores is the result matrix of the pseudocode, containing the score assigned to each time slot of each day during the search duration. Three slots with the best scores are chosen from the highest from the totalScores.

Analysis of the algorithm shows that the meeting slots can be scored in O(m*s) time complexity, where m is the total number of meetings of all participants and s is the number of slots in the search period.

IV. RESULT ANALYSIS

Evaluation is done by measuring the time needed to search for optimal schedules with various combinations of number of meetings and slots in the search duration. For this purpose, random meetings are generated by computer with the same probability of falling into each time slot. For each number of meetings and slots size, measurement is done five times, then averaged. The evaluation is done on a personal computer with Intel Core i7, 1.99 GHz processor and 8GB RAM. Table 1 shows the result of the evaluation.

Evaluation shows that for experiment of 324 slots (36 working days, 9 hours each) and 1800 meetings, the average time is 656 milliseconds. The average time is below 1 second for each of the experiment, which is considered acceptable waiting time for average user.

Table- I: Average Computing Time to Search a Schedule

slots	meetings	time (ms)
108	600	117.231
108	1200	167.9314
108	1800	246.4288
216	600	204.0088
216	1200	305.376
216	1800	448.3004
324	600	275.7892
324	1200	417.7898
324	1800	656.6888

V. CONCLUSION

In this research, an applied solution for meeting scheduling is proposed. The model calculates the best meetings options using a heuristics based on participants' availability and prioritization of participants. As a proof of concept, a mobile application is built using the Android platform and Google Calendar API. In the model, a meeting host can just input the particpants, the search period, and the meeting duration (in hours). The system will then come up with three best options based on the heuristics. The time needed to compute the schedule options is determined by total number of meetings of all participants and the number of slots in the search period. Evaluation shows that for 324 time slots and 1800 meetings, the average computing time is 656.69 milliseconds (less than one second). Therefore, we consider the computing time of the model fast enough to be used in real world problem.

Future studies can be including location-based measurement into the model to calculate the travel time between meetings.

REFERENCES

1. A. Bouhouch, C. Loqman, and A. El Qadi, "CHN and min-conflict heuristic to solve scheduling meeting problems," in *Studies in Computational Intelligence*, vol. 774, Springer Verlag, 2018, pp. 171–184.
2. P. Hosein and S. Boodhoo, "Event scheduling with soft constraints and on-demand re-optimization," *2016 IEEE Int. Conf. Knowl. Eng. Appl. ICKEA 2016*, pp. 62–66, 2016.
3. B. Chai, A. Costa, S. D. Ahipasaoglu, C. Yuen, and Z. Yang, "Optimal Meeting Scheduling in Smart Commercial Building for Energy Cost Reduction," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3060–3069, Jul. 2018.
4. B. P. Lim, M. van den Briel, S. Thi É Baux, R. Bent, and S. Backhaus, "Large neighborhood search for energy aware meeting scheduling in smart buildings," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9075, pp. 240–254.
5. O. Mussawar and K. Al-Wahedi, "Meeting scheduling using agent based modeling and multiagent decision making," *2013 3rd Int. Conf. Innov. Comput. Technol. INTECH 2013*, pp. 252–257, 2013.
6. A. B. BenHassine and T. B. Ho, "An agent-based approach to solve dynamic meeting scheduling problems with preferences," *Eng. Appl. Artif. Intell.*, vol. 20, no. 6, pp. 857–873, Sep. 2007.
7. R. H. Glitho, E. Olougouna, and S. Pierre, "Mobile agents and their use for information retrieval: A brief overview and an elaborate case study," *IEEE Netw.*, vol. 16, no. 1, pp. 34–41, 2002.
8. B. Vangerven, A. M. C. Ficker, D. R. Goossens, W. Passchyn, F. C. R. Spieksma, and G. J. Woeginger, "Conference scheduling — A personalized approach," in *Omega*, 2017, vol. 81, pp. 38–47.
9. J. Borg, "Implementation of an automated event scheduling system," 2017.
10. I. Refanidis and A. Alexiadis, "Deployment and evaluation of SelfPlanner, an automated individual task management system," *Comput. Intell.*, vol. 27, no. 1, pp. 41–59, 2011.
11. S. Boodhoo and P. Hosein, "On the distributed optimization of calendar events," in *2017 IEEE 10th International Workshop on Computational Intelligence and Applications, IWCIA 2017 - Proceedings*, 2017, vol. 2017–December, pp. 79–84.

AUTHORS PROFILE

Aswin Wibisurya, attained his undergraduate degree in 2009 and master degree in 2015, both in computer science from Bina Nusantara University. He is now a faculty member in School of Computer Science, Bina Nusantara University. His research interest includes mobile technology, algorithm optimization and multimedia information retrieval.

Ikhtiar Faahakhododo, attained his undergraduate degree in 2009 and master degree in 2017, majoring Computer Science from Bina Nusantara University. He is an active faculty member in Bina Nusantara University. His expertise includes mobile technology and computer network.