# Towards Efficient Near-Optimal Agenda Scheduling using Human-Centered Heuristics

**Peerat Vichivanives**

April 2023

CMU-CS-23-106

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Thesis Committee:**
Stephanie Rosenthal, Chair
Aaron Steinfeld
Laura Hiatt (NRL)

*Submitted in partial fulfillment of the requirements for the degree of Master of Science.*

**Keywords:** Task Scheduling, Human Strategies and Agent Heuristics

# Abstract

A person's agenda often has appointments with both location and temporal constraints, like doctor's appointments, as well as tasks that can be completed at any time and at one of multiple available locations, such as going to an ATM. When faced with a new task to add to their agenda, people are capable of doing so quickly but may find it cognitively challenging to consider the many constraints involved causing them to make mistakes and miss other agenda events. A tool that could help schedule these tasks could mitigate these challenges, but optimal schedulers can be very computationally expensive as they consider many potential completion times, especially when tasks have multiple potential locations.

In this work, we aim to contribute computational heuristics that are nearly optimal for adding tasks into existing agendas, yet are more computationally efficient compared to optimal schedulers. Towards this goal, we first study human strategies for scheduling tasks that can be completed in multiple locations. We found that people primarily use 3 heuristic strategies for scheduling: a) fitting the task as early in the day as possible, b) in the longest open time slot, or c) in a time slot between other activities that are spatially close by the new task. We then implemented human-centered heuristics in line with these strategies along with several other computational heuristics that could be utilized within an automated scheduling framework, and compare them in terms of their run-time and the inefficiency of the schedule length compared to optimal. Additionally, we tested two types of task insertions, a Single Task and a pair of tasks that must be temporally ordered (Order Specific Task) (such as a pickup and delivery), in order to understand how our human heuristics perform in varied conditions.

We found that our heuristics offer different trade-offs between computational run-time and schedule length inefficiency. Relative to the optimal scheduler, our human-centered heuristics are 100,000 to 1 million times faster at inserting a Single Task, but the resulting schedules are an average of 2-6% longer. Our other computational heuristics are slower than the human-centered heuristics, but are still 2x-600x faster than the optimal scheduler, and they find schedules that are closer to optimal. When scheduling Order Specific Tasks, we find similar results where our human-centered heuristics are still 100,000 to 1 million times faster at computing a schedule but are on average 7-14% longer. Our other computational heuristics perform 8-700 times faster than the optimal scheduler again with schedules that are closer to optimal. We conclude that it is possible to select a heuristic that balances required trade-offs in run time and schedule length depending on the needs of a particular scheduling task.

## Acknowledgments

I could not have undertaken this journey without Professor Stephanie Rosenthal, my advisor and committee chair who guided me through this project.

I would like to express my deepest gratitude to Dr. Laura Hiatt for her help throughout the project as well has all the helpful comments on the thesis itself.

I would like to express my deepest appreciation to thank Dr. Elizabeth Carter for all her help with the human study as well as the creation of the thesis defense.

Special thanks to Elchanan Haas and Arnav Gupta who have helped me with this project throughout its earlier days.

I'm extremely grateful to my thesis committee member Professor Aaron Steinfeld for his invaluable guidance and feedback.

Lastly, I would like to thank my parents for their support and belief in me.

# Contents


**1 Introduction** **1**

**2 Related Work** **3**

**3 Human Methods** **5**
3.1 Agenda Design . . . . . 5
3.2 Participants . . . . . 7
3.3 Experimental Setup and Procedure . . . . . 7
3.4 Results and Interpretation . . . . . 8

**4 Method** **11**
4.1 Optimal Scheduler . . . . . 11
4.2 Human-centered heuristics . . . . . 11
4.3 Start of Day . . . . . 13
4.4 Longest Time Gap . . . . . 13
4.5 Closest Location . . . . . 14
4.6 Longest Idle Time . . . . . 14
4.7 Random . . . . . 14
4.8 Other Heuristics . . . . . 14
4.9 Locally Optimal . . . . . 15
4.10 Warped K Means . . . . . 15

**5 Experiments** **19**
5.1 Setup . . . . . 19
5.2 Results . . . . . 20
5.3 Single Task . . . . . 21
5.4 Order Specific Tasks . . . . . 23
5.5 Comparing the two . . . . . 26

**6 Discussion** **29**

**Bibliography** **33**

# List of Figures


3.1 The village map that participants were given to find appointment and task locations and calculate travel times. . . . . 6

3.2 Sample agendas for the scheduling task. Left: An agenda with eight appointments used for the *Single Fixed*, *Single Flexible*, and *Two Task* conditions. Right: An agenda with seven appointments and one movable task for the *New Plus Existing* condition. . . . . 7

5.1 We computed the ratio of the schedule length for each heuristic compared to optimal schedule length (schedule length inefficiency) and present the percentage increase in length. Warped K Means is always optimal, while the human strategies perform the worst. . . . . 21

5.2 Run times for each of the heuristics in seconds on a log scale. Optimal and Warped K Means take the longest, while the human-centered heuristics are the fastest. . . . . 22

5.3 We computed the ratio of the schedule length for each heuristic compared to the optimal schedule length (schedule length inefficiency) and present the percentage increase in length. Warped K Means is the closest to optimal, while the human strategies perform the worst. . . . . 24

5.4 Run times for each of the heuristics in seconds. Optimal and Warped K Means take the longest, while the human-centered heuristics are the fastest. . . . . 25

5.5 We computed the ratio of the schedule length for each heuristic compared to optimal schedule length (schedule length inefficiency) and present the percent longer than optimal. . . . . 27

5.6 Run times for each of the heuristics in seconds on a log scale. The first 3 columns are from the Single Task and then the next 3 from the Order Specific Tasks. . . . 28

# Chapter 1

# Introduction

The activities that people perform on a daily basis are often time and location constrained, but not always. People must attend *appointments*, such as work meetings or doctor's visits, that occur in a specific location at a specific time. Additionally, people must complete *tasks*, such as dropping off mail at the post office, that can be fulfilled at any time in the day. Furthermore, some tasks, such as going to an ATM, can be completed at many different locations.

The cognitive load required to schedule a day's activities is substantial [18]. Many tools have been proposed to help people with scheduling their daily agendas (e.g., [4, 6, 22, 26]). However, these tools often either require people to make their own scheduling decisions (e.g., [4, 6]) or rely on optimal schedulers that take exponential time to exhaustively check every possible solution (e.g., [9]). Their need for human input or high computational costs preclude their use in real-time situations, such as for quick additions of just one or two tasks that are achievable in multiple locations, like stopping by an ATM or grocery store [2].

In contrast to the computational challenges of support tools, people manage to find nearly-optimal agendas fairly quickly and without mistakes most of the time, such as when scheduling appointments over the phone or fitting a trip to the ATM between meetings. While some previous research has examined how people execute their agendas (e.g., [5]), it is still necessary to investigate what strategies people actually use to add tasks to agendas—particularly those that can be done at multiple locations. By isolating potential strategies that people use to schedule quickly and effectively, our long-term goal is to develop efficient heuristic algorithms that can support people as they create, update, and execute their agendas to reduce the cognitive load associated with scheduling correctly.

Towards this vision, the goal of the present work is to develop heuristic scheduling algorithms that can fit individual tasks with multiple possible locations into existing agendas nearly optimally and significantly faster than optimal schedulers. We draw upon prior work that studied and adapted human strategies for task scheduling [24]. In that work, the authors created an automated scheduling heuristic that focused on the proximity of the new task to the agenda's already-scheduled activities. Their heuristic significantly reduces the time to find an optimal or nearly-optimal agenda with the new task compared to an optimal scheduler. However, this prior work does not account for the possibility that some tasks may be completed at different locations (e.g., withdrawing money from any ATM), as is common in real-world task scheduling.

Furthermore, while their approach is faster than an optimal scheduler, due to it using an optimal scheduler to solve parts of the problem, it is still relatively slow.

Our work addresses this shortcoming and proposes heuristics for domains in which tasks can be completed at multiple locations. First, we describe a study in which we capture how people fit a multi-location task into an existing agenda. Our findings show that scheduling is a high cognitive load task for people to complete accurately but people had several heuristics to help them. From those heuristics, we were inspired to create several human-centered heuristics that we then compare to an optimal scheduler. Additionally, we compare these heuristics to the Warped K Means approach proposed by [24] as well as Locally Optimal. Our analyses highlight the trade-offs between schedule length and run time. We test Single Tasks (1 task insertion that has no other constraints, such as going to an ATM to get money) and Order Specific Tasks (a pair of tasks where one must come before the other, such as going to a convenience store to get a birthday card and sending it at a post office). For Single Tasks, heuristics inspired by our study are millions of times faster than the optimal scheduler and yield schedules that are on average 5% longer than the optimal schedule. Conversely, the Warped K Means approach is only 2x faster than optimal but is guaranteed to find an optimal schedule. These results show potential for selecting heuristics that balance speed and schedule length so that they can be utilized in deployable scheduling aids in the future. As for Order Specific Tasks, we find that our heuristics are millions of times faster than the optimal scheduler but have results around 7-12% longer than the optimal schedule. Warped K Means increased to 7 times faster but is 0.57-3.81 % longer than the optimal solution. Thus we show that each of these heurisitcs can be really useful for scheduling depending on the needs of the particular task.

# Chapter 2

# Related Work

Agenda management is challenging because of the numerous constraints placed on appointments and tasks, such as potential locations and travel time [3, 18, 19]. Building a decision support tool to autonomously schedule tasks and appointments in people's calendars is an NP-hard constraint satisfaction problem (CSP) that involves modeling the temporal constraints (i.e., when an appointment must start and end) and spatial requirements (i.e., where an appointment can be completed) to determine which subset of appointments are feasible to complete [9]. Recent work has focused on framing unconstrained or flexibly-constrained tasks within the same constraint satisfaction scheduling framework [22]. Because people potentially have many appointments during their day, a decision support agent to assist in scheduling tasks must be able to solve scheduling problems more quickly than exponential time. Some approaches address this by trading off the optimality of the solution (e.g., minimizing agenda start time to end time, placing all of the appointments/tasks on the agenda) with the speed of finding a solution. For temporally- or spatially-flexible tasks in particular, [22] demonstrates an algorithm for optimizing an agenda that runs in linear time but at the cost of not scheduling all events. This algorithm was then improved using hill-climbing techniques to create better agendas [1]. In contrast, we intend to explore how we can schedule all of the tasks, but we may not end up with an optimal solution.

There are an ever-increasing number of calendaring, to-do list, and scheduling tools to help people determine when they can attend different appointments at various locations around town and when they can complete other tasks throughout the day. Some particularly interesting ones have been tools to support multi-person meetings, including Google Calendar and Microsoft Outlook, that allow group members to view each other's schedules but require the user to pick appropriate meeting times. Other tools automate parts of the scheduling process, including one tool that solicits time preferences from users [6, 21] and another that performs meeting time negotiations on each user's behalf [4]. However, these still require user input. Other tools learn scheduling strategies from previous user actions [10, 20] or follow existing workflows [7] in an attempt to fully automate the negotiation process of finding suitable meeting times for participants with potentially conflicting objectives. Yet other workflows try to use learning algorithms to become better over time by incorporating the user's specific goals [14].

To address the challenges related to an individual's appointment and task management rather than a team's, previous solutions have been proposed for very specific domains. For example, a tool has been developed that assists students in prioritizing and scheduling their assignments

[26]. That paper focuses on prioritizing work and due dates and not scheduling when to travel to different locations to complete tasks. Additionally, tour itinerary tools have been proposed to help people decide appropriate agendas for sightseeing at multiple locations in a city [17, 23]. If one can consider a task as "seeing a sight", then these tools do account for the possibility of a task being accomplished at one of many possible tour sites (e.g., one of numerous archaeological sites). However, there are few options for required appointments with location and time constraints to schedule tour tasks around.

[17] worked on tourist group planning. This involved scheduling a tour itself alongside grouping people to assign to tour guides. This is a very specific algorithm given that they also take into account where tourists have been before, overall popularity, and the tour guide's experience with that specific location. The algorithm used separated this process into differing tasks, such as assigning tourists to tour groups, choosing points of interest for tour groups, and picking a tour guide for tour groups. The algorithm heurisitcs used included K-means clustering (a predecessor to our Warped-K-Means Heuristic) for tourists to create the groups or calculate totals. For assigning a point of interest to tour groups, they considered maximum or minimum interest in the group.

[23] focused on the actual creation of a web-based planner to help plan a holiday in Northern Greece. This system took in preferences such as cost, interest, how much time you have, and time of the year. It then created creates a proposed list of places to visit and allowed the user to change that. It then created a proposed schedule and once again allowed the user to change it. This uses the SELFPLANNER Scheduler created by the same people in [21, 22]. Thus, this work focused on the deployment of the program to a specific use case rather than developing a new scheduling algorithm itself.

While prior work has found that humans find it cognitively challenging though possible to schedule tasks quickly while finding reasonable solutions most of the time [18], few studies have focused on human strategies for actually scheduling their activities. It has been shown that humans usually use only a few heuristics for making spatial decisions [13]. This work aims to replicate and build on the prior work to understand how people make more complex scheduling decisions [24] and then create computational heuristics that reflect the findings.

# Chapter 3

# Human Methods

To examine people's strategies for scheduling their agendas, we created an online experiment in which participants had to insert tasks into existing agendas that generally consisted of inflexible lists of appointments; i.e., neither the time or the location of the appointments could be changed to "make room" for the new task(s). Participants were asked to insert tasks under each of four circumstances:

1. adding a Single Task with only one possible location to an inflexible list of appointments (*Single Fixed Task*);
2. adding a Single Task that could be flexibly done at one of multiple locations to an inflexible list of appointments (*Single Flexible Task*);
3. adding two tasks, some with one and some with more than one potential locations, to an inflexible list of appointments (*Two Tasks*);
4. adding a Single Task with one or more possible locations to an inflexible list of appointments that also contained one task with a fixed location that could be moved to a different time if necessary (*Single Plus Existing Task*)

Conditions 3 and 4 did not inform our heuristics but did potentially impact some results. We describe them briefly and will analyze them in future work.

## 3.1 Agenda Design

A member of the research team designed 29 agendas using the map shown in Figure 3.1, seven for each of the four conditions described above plus an additional example of the *Single Fixed Task* condition to use as a test agenda when giving instructions. Each dot on the map's roads signified one minute of travel time, and the appointments and tasks were timed and located to limit potential solutions. For example, a five-minute visit to drop off supplies at the Art Studio would not fit in a 15-minute gap between a Health Center Appointment and Library Story Time (even though it is on the way) because it takes 14 minutes (i.e., black dots) to get from the Health Center to the Library. Task locations and agenda *time gaps* (times between fixed appointments) varied widely.


Figure 3.1: The village map that participants were given to find appointment and task locations and calculate travel times.

For the *Single Fixed Task*, *Single Flexible Task*, and *Two Tasks* trial conditions, each agenda had eight appointments. These agenda items had an "A" code for appointment and were colored red to signify that they were immovable in both time and location. The *Single Fixed Task* and *Single Flexible Task* agendas were created so that there was only one time gap in the schedule to accommodate the new task and, in the case of *Single Flexible Task*, only one possible task location that would fit in that time gap (see Figure 3.2, left). Similarly, the agendas for the *Two Tasks* condition had two gaps, one to accommodate each of the new tasks. For the *Single Plus Existing Task* condition, the agendas were created with seven appointments and one task (with "T" code and colored blue) that could be moved if needed to find an appropriate gap, with one ideal solution (see Figure 3.2, right). Correct solutions were evenly distributed across time gaps. In other words, one of the seven agendas could be solved by putting the task between appointments 1 and 2, one could be solved by putting the task between appointments 2 and 3, etc.

| | Time | Agenda Item | Code |
|---|---|---|---|
| 1 | 9:00-9:50AM | Farm Chores | A |
| 2 | 10:00AM-10:50AM | Amphitheater Rehearsal | A |
| 3 | 11:00-11:35AM | Orange Plaza Show | A |
| 4 | 12:00-1:35PM | Real Cafe for Lunch | A |
| 5 | 2:00-2:55PM | Art Studio Class | A |
| 6 | 3:00-3:00PM | Comm. Room 1 Karate | A |
| 7 | 3:20-4:15PM | Marina Sailing | A |
| 8 | 4:30-7:00PM | Fire Station Volunteering | A |

| | Time | Agenda Item | Code |
|---|---|---|---|
| 1 | 9:00-10:45AM | The Real Cafe | A |
| 2 | 11:00-11:45AM | Comm. Room 2 Mtg. | A |
| 3 | 12:00PM-1:00PM | Art Studio Class | A |
| 4 | 1:10PM-2:00PM | Observation Room | A |
| 5 | 2:10PM-2:15PM | Library Drop-off | T |
| 6 | 2:30-3:25PM | Fire Station Event | A |
| 7 | 3:30-5:00PM | Dispatch Volunteering | A |
| 8 | 5:20-7:00PM | Beach Dinner | A |

Figure 3.2: Sample agendas for the scheduling task. Left: An agenda with eight appointments used for the *Single Fixed*, *Single Flexible*, and *Two Task* conditions. Right: An agenda with seven appointments and one movable task for the *New Plus Existing* condition.

## 3.2 Participants

We used the Prolific research recruitment website[1] to find participants for this research who were 18 years of age or older, were fluent in English, and had normal or corrected-to-normal vision (to see the map). Twenty participants successfully completed the experiment and were each paid 5USD. Ten were female and ten were male, their mean age was 27.1 years (range 19-51, median 23, stdev 8.6), and they were primarily located in the UK (6), USA (4), and continental Europe (9). Our Institutional Review Board approved this research. Because the purpose of the experiment was exploratory and did not directly compare the conditions, we did not require sufficient participants for computing statistical significance.

## 3.3 Experimental Setup and Procedure

We used Qualtrics[2] to host the survey. First, participants provided informed consent, confirmed that they were eligible, and gave their anonymous but distinct Prolific IDs. Then, they were shown instructions explaining how travel times, appointments, and tasks worked in the agenda. They were given the test agenda and asked the multiple-choice question, "What is the most convenient time that your friend could go to Orange Plaza and spend 15 minutes there without disrupting the schedule above?", with a radio button option for each time gap. Participants had to select a time gap to go to Orange Plaza. Next, they were asked to sort the time gaps between appointments in the order they checked them: "In what order did you check the schedule? Please drag the available times that you checked into the correct order. You don't need to drag the ones that you didn't check." Participants only advanced to the rest of the study if they correctly

[1] https://www.prolific.co
[2] https://www.qualtrics.com

answered the first question. Otherwise, they saw a page thanking them, and their participation was terminated.

Ongoing participants were next shown four trial agendas in the same order, with one agenda each from the four conditions. The four agendas were selected randomly from each pool of seven per condition. The trial procedure was the same as that in the test example, asking participants to add a new task to the agenda and to rank the order in which they checked the time gaps to determine their answer. Note that for *Two Tasks*, they did this ranking twice. For *Single Plus Existing Task*, they also selected if they moved the existing task and, if so, where. For the *Single Fixed Task* condition questions specifically, participants were instructed, "Add a task without moving any of the red appointments with the code 'A' in the schedule. What is the most convenient time to spend [number] minutes at the [place] [completing a task]?" The number of minutes, place, and task changed for every question. They had to respond using a multiple-choice format prompt with the seven time slots listed in chronological order. Next, they had to complete the 'Pick, Group, and Rank' question as described previously. For the *Single Flexible Task* condition, the second sentence had different options signifying multiple potential task locations, such as "What is the most convenient time to spend [number] minutes to [complete a task] at [multiple locations]?" where the last brackets could include terms like "any ATM" or "either Ralphs or the Health Center". They also selected a multiple choice response and did the 'Pick, Group, and Rank' question. The two other conditions had similar prompts.

After completing the four trials, participants were prompted to self-reflect about their overall strategies. They were asked, "Which of the following strategies did you use to find slots where new tasks would fit? Please select all that you used," with four possible responses (selected from [24]): start at the beginning of the day and check in order (**Start of Day**), check the longest time gaps between appointments first (**Longest time gap**), check time gaps with nearby destinations first (**Closest Location**), and other (please describe), which had a text entry field. The next question asked, "Which strategy did you use most often to find slots where new tasks would fit? Please select only one," and had the same potential responses. Finally, they were asked a free response question, "How do you typically schedule tasks in real life?" After submitting answers to these questions, they were thanked and redirected to Prolific's main page. The experiment lasted about 20 to 25 minutes.

## 3.4 Results and Interpretation

There were 128 total responses to each of the time gap selection and ranking questions across the four trials, including 40 for the two trial conditions of interest (20 for *Single Fixed Task* and 20 for *Single Flexible Task*). We analyzed the responses in an exploratory fashion in order to uncover the heuristics that people used to schedule the new tasks. No statistical tests were performed between conditions because there were no hypotheses to test. We analyzed the order that participants ranked the time gaps to infer what strategies they used. We focused on the first time gap that was ranked because it indicates the strategy that they started with. We found that they ranked first:

- the earliest time gap 16 times,
- the longest duration time gap 10 times,

- the closest appointment location (landmark) 16 times,
- the time gap when the new task fell on the travel path between activities 5 times,
- the latest time gap 4 times, and
- other gaps 2 times.

Note that these are not mutually exclusive; the earliest time gap could also be the largest or closest, thus the sum is greater than 40.

When self-reflecting about which strategies they used to complete the trials (selecting all that apply), 15 of the 20 participants reported using the **Start of Day** strategy and checking each subsequent time gap in order, 11 participants reported checking the **Longest time gaps** first, and 6 reported checking time gaps that had the **Closest Locations** to the new task. No participants reported other strategies. When identifying a single strategy they used the most, 9 selected **Start of Day**, 9 selected **Longest time gaps**, 2 selected using **Closest Locations**, and 0 mentioned another strategy. Thus, self-reflection aligned with per-trial analyses.

Interestingly, even with these relatively simplistic strategies, it still took participants at least 5 minutes to check the 7 time gaps and complete the questions for each agenda. This aligns with previous findings [18] indicating that humans find it difficult to schedule. Additionally, participants selected the correct time gap 75% of the time for *Single Fixed Task* and 85% of the time for *Single Flexible Task*. These results imply that the task does require a high cognitive load, and that a solution could be implemented algorithmically to support human decision making around their agendas.

# Chapter 4

# Method

Our initial study showed that people employed one of three strategies when selecting the order of gaps to consider to fit new tasks into their agendas: (1) **Start of Day**, which checks time gaps from earliest to latest; (2) **Longest time gap**, which checks time gaps from longest to shortest (i.e., by subtracting one activity's scheduled end time from the next activity's scheduled start time); and (3) **Closest Location**, which checks time gaps between the closest activities to the new task's location(s) to those between activities the farthest away. Our next step was to determine how well these heuristics work when their ability to efficiently schedule new tasks, with multiple possible locations, is evaluated computationally on an array of agendas. We therefore implemented these strategies as algorithmic heuristics as well as two other computational heuristics that seem human-like. We also computed an optimal solution for comparison.

## 4.1 Optimal Scheduler

First, we wanted to define a baseline against which to benchmark everything else. The optimal solver is exactly that: it creates a perfect solution for our schedule. However, it is very slow. An optimal scheduler takes as input a set of time and location constraints for appointments and only location constraints for tasks, and it outputs an agenda in which all appointments and tasks are reachable, including navigation time between them. The scheduler also minimizes the overall duration of the agenda. This requires considering all possible times and locations for the new and existing tasks to be placed, effectively rescheduling the entire day. To generate the agenda optimally, we use a Mixed Integer Programming paradigm adopted from [15] to provide a formulation for minimizing agenda length given both tasks and appointments.

## 4.2 Human-centered heuristics

We implemented the human strategies and other sorting strategies for comparison. The generic sorting heuristic algorithm is shown in Algorithm 1. All of these human-centered heuristics follow the same pattern. The algorithm takes as input an existing agenda with $|A|$ activities as well as a new task $task$ that could be achievable at some number of possible locations $task^{loc1}, task^{loc2}, ....$ To help apply our heuristics, we first re-represent the agenda as $|A| - 1$

time gaps ($gap \in time\_gaps$) such that the $i$th time gap $gap_i$ starts when the $i$th activity (task or appointment) $a_i$ ends; $gap_i^{start} = a_i^{end}$. The time gap is then over when the next activity $a_{i+1}$ is scheduled to start; $gap_i^{end} = a_{i+1}^{start}$. The starting location of the time gap is the location of the earlier activity $gap_i^{loc\_start} = a_i^{loc}$, and the ending location is the location of the next activity $gap_i^{loc\_end} = a_{i+1}^{loc}$.

Note that because of the flexibility of the task start times, the task insertion will shift around any other task start times within the time gap (without changing the order of any tasks) in order to maximize the likelihood of the new task fitting. For example, suppose there is an appointment ending at 10:00, another starting at 11:00, and there are two 5-minute tasks in between requiring 0 travel time. The three time gaps that would be returned are:

- 10:00 - 10:50 if both existing tasks were completed just before 11:00,
- 10:05 - 10:55 if the existing tasks were split up, and
- 10:10 - 11:00 if both existing tasks were done right at 10:00 and the new task would follow the second task.

This method takes advantage of the fact that the tasks could be completed any time in the 1-hour window and allows us to quickly check whether the new task could fit in three possible time gaps, while giving the new task as much space as the agenda could possibly allow without reordering the existing agenda tasks.

The algorithm sorts these time gaps using the appropriate heuristic (`h_sort`). Because it is possible for the heuristics to sort by the new task's possible locations in addition to the times of the gaps (e.g., Closest Location), `h_sort` sorts the Cartesian product (all possible combinations) of the time gaps with the possible locations of $task$. Each heuristic's sort returns a list representing the order in which to check the resulting pairs $(gap, task^{loc})$ to try to fit $task$ into the agenda (Line 5).

The algorithm then iterates through each of the gap-location pairs, attempting to insert $task$ at $task^{loc}$ into the agenda at time $gap^{start}$ (Lines 6-9). Once a time gap is found that can accommodate the new task, the loop breaks and the resulting agenda is returned (Line 8). If any time gap is found, then the task fits in the agenda without making the agenda longer. Thus, we would be perfectly efficient in this case. If no suitable time gap is can be found before the last appointment, then the algorithm inserts the task among the tasks after the last appointment according to the heuristics and returns (Line 11-12). In this case, this could be efficient (i.e., the optimal scheduler also adds that much time to the schedule) or inefficient. Now that we have described how the algorithm handles the Single Task case, we detail how this algorithm is applied by each particular heuristic in turn. We next address Order Specific Tasks. If multiple Order Specific Tasks must be added (e.g., in the case when a delivery must occur after a pickup), the algorithm first runs Algorithm 1 for the pickup task and then runs the algorithm again except with the agenda starting after the pickup task instead of the true beginning of the day. This ensures that the algorithm only selects a delivery time after the pickup task has completed.

**Algorithm 1** General Heuristic Algorithm

**Input:** $agenda$, $task$
**Output:** $agenda$ ▷ containing the new task

$time\_gaps \leftarrow$ `time_gaps(`$agenda$`)`
$gap\_loc\_pairs \leftarrow$ `h_sort(`$time\_gaps$`,` $task$`)`
**for** `(`$gap, task^{loc}$`)` in $gap\_loc\_pairs$ **do**
    **if** $agenda$`.insert(`$task, task^{loc}, gap^{start}$`)` **then**
        **return** $agenda$
    **end if**
**end for**
$agenda$`.insert(`$task, task^{loc}, end\_of\_day$`)` ▷ task did not fit between appointments
**return** $agenda$

## 4.3 Start of Day

Checking slots beginning at the Start of Day was the most common strategy used by our participants. In our heuristic implementation, the Start of Day sorting algorithm takes the $time\_gaps$ and $task$ (with potentially several locations where it could be achieved), and returns the Cartesian product of all gaps and locations sorted by the start time of the gap $gap^{start}$. In the case of multiple task locations, the order in which those locations are sorted to be tested is arbitrary. If none of the gap-location tuples fit in between the appointments, then the task (at an arbitrary location chosen from all the locations possible) is inserted directly right after the last appointment (even if there are other tasks after the last appointment at the end of the day already). This is the case where tasks might be not optimal and using no heuristic here may increase the chances that it is not optimal.

## 4.4 Longest Time Gap

Checking the Longest time gap first was another commonly used strategy by our participants. The idea behind this strategy was that the larger time gaps are more likely to fit a new task. Thus, in our heuristic implementation, the `h_sort` algorithm sorts time gaps by largest to smallest. In the case where there are multiple possible task locations, it sorts the locations arbitrarily. However, if none of the gap-location tuples fit between the appointments, the algorithm needs to fit it at the end of the day. To do this, it generates all of the time-wise adjacent pairs of tasks that appear after the last appointment and inserts the new task in between the pair with the longest distance between them. If there are many possible locations, one is chosen randomly. We do this to approximate the "longest gap" at the end of the day. This heuristic at the end of the day should help reduce the chance and degree of inefficiency because the longest distance would mean that it is farthest apart from each other and thus there should be the largest amount of space in between them. This makes it more likely that the task will be closer to or at an optimal point in the schedule.

## 4.5 Closest Location

The Closest time gap was the another common strategy that was found in the human-subject studies. It makes sense that if a task is close to another task, you may be able to achieve both sequentially because navigation time is small. In our implementation, the heuristic sorting algorithm sorts by the distance of the $task$ to the start and end locations of the time gap. If multiple gaps have the same distance, then the order of the gaps is chosen randomly. If the task does not fit in any of the gap-location pairs, then the algorithm generates all of the time-wise adjacent pairs of tasks that appear after and including the last appointment and inserts the task next to the existing task that is closest to the new task. If no tasks exist at the end of the day, it is placed after the last appointment. Given that this has to consider each task location individually compared to the other heuristic sorting once, it would take longer computationally than the other heuristics.

## 4.6 Longest Idle Time

We developed the Longest Idle Time heuristic based on the idea that the longest time gap may or may not include long travel time between activities, so it makes sense to subtract the travel time when computing the longest time gap. While it might be hard for humans to do this, it is only a couple more mathematical operations for a machine and thus is pretty simple for it to do. The idle time of a time gap is the non-travel slack time between activities. Our heuristic computes this idle time for each time gap and then sorts by the longest to shortest idle time, breaking ties randomly. Similar to the Longest time gap heuristic, if there is not a gap-location pair where the task fits, the algorithm sorts the tasks after the last appointment, finds the longest travel time, and inserts the new task in there.

## 4.7 Random

As a baseline, we designed the Random heuristic to randomly shuffle the gap-location tuples rather than sort them with a particular metric. The algorithm tests each tuple until the task fits or until there are no more tuples. If the task does not fit before the last appointment, then all gap-location tuples after the last appointment are shuffled, and the task is inserted at the given time in the given location.

## 4.8 Other Heuristics

We realize that the sorting-based heuristics above make many simplifying assumptions in order to efficiently compute an agenda. In this section, we propose two heuristics that focus on running an optimal scheduler on a small subset of the entire agenda scheduling problem, making optimal decisions on targeted portions of the agenda instead of on the agenda as a whole. This allows for much faster run times because the optimal scheduler is scheduling fewer activities; despite this, it is expected to still often find the optimal schedule because any solution is optimal as long as all tasks fit in agenda before the last appointment. This approach draws upon and expands

other work that proposes opportunistic rescheduling algorithms to try to minimize the amount of rescheduling that must be done. Foundational work by [11] and [12], for example, focused on finding similar jobs in job rescheduling tasks but did not consider spatial constraints. Similarly, [25] focused on constraining vehicle routing tasks spatially for rescheduling but without time constraints.

## 4.9 Locally Optimal

Locally Optimal was inspired by Start of Day, but instead of checking each time gap, we instead look at each appointment pair and then call the optimal solver for the problem. This allows for better results than Start of Day because we optimally solve for an appointment pair rather than a time gap, which means we can move any existing tasks around. To insert a new task using 'the Locally Optimal heuristic, we first consider each appointment pair in chronological order (i.e., the first and second, the second and third, etc.). For each pair, the algorithm creates the relevant location constraints for those two appointments, all tasks that are currently scheduled within the appointment pair, and the new task location(s). It then calls the optimal scheduler to find the optimal task order between the pair of appointments. If the optimal solver returns an ordering that fits the new task and all previous tasks in the allotted time between appointments, then the algorithm returns the agenda with the task inserted. If the task does not fit in the first appointment pair at any of its possible locations (if more than one is possible), then the algorithm tries the second appointment pair, and so on. If the new task does not fit in any appointment pair at any location, the algorithm adds it to the end of the agenda by 1) creating constraints using the last appointment location, the tasks already scheduled after that appointment, and the new task at one of its possible locations chosen at random; and 2) using the optimal solver to find the optimal (shortest navigation time) order of those tasks.

For an Order Specific Task (a pickup and delivery, for example), the algorithm first schedules the first part of the task, and then considers scheduling the second part of the task at times after the first is completed. This is done by calling the algorithm as per normal on the first task, and then setting the first task as the first appointment of the day to compute for the second task.

## 4.10 Warped K Means

The Warped K Means heuristic, drawn from [24], is similar to Locally Optimal in that it optimally solves for portions of the schedule at a time, but it intelligently chooses on which portion to focus instead of progressing chronologically. In other words, it combines the efficiency of the Locally Optimal strategy with a version of the *Closest* strategy that we identified as commonly used by people. Specifically, the heuristic adds tasks to agendas by focusing on the spatio-temporal proximity of the new task to appointments and tasks already on the agenda. Intuitively, it finds portions of the agenda that are likely to support the new task: because spatial-closeness implies that navigation time between the activities is shorter, less time is needed in those portions of the agenda to fit the new task. Additionally, because the number of activities to reschedule is much smaller than the entire agenda, the speed of solving the new agenda should greatly decrease.

**Algorithm 2** Warped K Means

**INPUT:** current_agenda, new_task
**OUTPUT:** new_agenda

$clusters$ = warped-k-means($current\_agenda$)
$new\_task\_cluster$ = $clusters$.cluster($new\_task$)
**while do** $partial\_agenda$ == null
    $tasks$ = $new\_task\_cluster$.tasks() $\bigcup$ $new\_task$
    $partial\_agenda$ = solve-optimally($tasks$)
    $prior\_cluster$ = $new\_task\_cluster$.prior-cluster()
    $next\_cluster$ = $new\_task\_cluster$.next-cluster()
    $new\_task\_cluster$.add($prior\_cluster$,$next\_cluster$)
**end while**
**return** $current\_agenda$.update($partial\_agenda$)

There are three components to this heuristic scheduling algorithm: (1) clustering the existing agenda so that tasks and appointments are binned according to spatio-temporal proximity to the new task using Warped K Means [16]; (2) finding the optimal solution to the subset of activities (including the new task) within the cluster closest to the new task; and (3) expanding the search area if no solution is found, repeating the process until a valid agenda is reached. This is similar to [24], but we include an additional factor of multiple locations. The heuristic approach is described in prose below; pseudocode appears in Algorithm 2.

The first step of our heuristic scheduling algorithm is to cluster the existing agenda using Warped K Means, which groups sequences of data (here, activities) in space and time. It then classifies the new task to an existing cluster to find the physically closest cluster for that task and proceeds with the next step. For multiple locations, we would classify each of the locations of the task to an existing cluster to find the closest location. Each of the locations would then begin their search at their relevant cluster. Then we call the optimal solver.

Given the new task's cluster, our algorithm selects the partial agenda consisting of only the activities in that cluster. It then attempts to optimally schedule that partial agenda with the new task, constrained by the partial agenda's surrounding locations, start times, and end times. Intuitively, this means that the algorithm is checking to see if the new task can be fit into that partial agenda. If it can, the resulting partial agenda is used to update the overall agenda, generating a solution to the problem. If it cannot fit the new task into the first cluster, the heuristic incrementally expands the partial agenda to include the temporally surrounding clusters, stopping when it finds a solution. If the new task has multiple locations, then we try each of the new task locations before expanding the partial agenda. If a solution is not found within the first and last appointments of the day, then the overall agenda's end time is dropped and all activities, including the tasks after the last appointment, are considered. This last step is equivalent to the optimal solver.

For an Order Specific Task (a pickup and delivery, for example), the algorithm first schedules the first part of the task, and then considers scheduling the second part of the task at times after the first is completed. This is done by calling the algorithm as per normal on the first task, and

then setting the first task as the first appointment of the day to compute for the second task. Note that for order-specific tasks, this is no longer equivalent to an optimal solver.

# Chapter 5

# Experiments

To test our human-centered heuristics along with other computational strategies, we ran a variety of experiments and analyzed their results.

## 5.1 Setup

**Experimental Conditions**

We developed many experimental conditions to represent the varying possibilities that could be found in the real world. We tested our heuristics under a variety of conditions, represented as the below parameters.

*Task Insertion Type* (Single Tasks, Order Specific Tasks) We have two types of task insertions. In Single Tasks insertion, the user simply has a new task, such as going to the ATM that needs to be added to their schedule in the day. In Order Specific Tasks one has two tasks, one of which needs to be completed after the other. For example, it could be buying a postcard at one of the convenience stores and then sending that postcard at one of the post offices. Because you cannot send a postcard before you buy it, the order of the tasks is constrained.

*Number of Task Locations* (num_task_locations) The number of task locations represents the possible locations at which the new task could be achieved. For example, going to your house is a task that can only be achievable at one location, while going to an ATM can be achieved at many possible locations in town. We varied this parameter from 1 to 5 locations per new task. In the case of Order Specific Tasks, each of the tasks will have between 1 and 5 locations.

*Agenda Slack* (slack) The slack represents the amount of time between appointments minus the travel time between the two appointments. Slack is needed between appointments for task insertion to occur; otherwise, no tasks could fit in between appointments and all of the tasks would need to go at the end of the day. The amount of slack affects how many tasks can fit in between appointments, which tasks can fit, and which time gaps can be used. Our Agenda Generation algorithm chose a random amount of time between [25, 100] for each appointment pair. These numbers were chosen experimentally to balance the number of end-of-day tasks.

**Agenda Generation**

For each experiment, we randomly generated existing agendas and an additional task to add into them. The specific agenda generation algorithm is as follows. First, we randomly sample

29 + num_task_locations unique locations on a 50x50 grid. The first 5 locations are designated as appointment locations. The next 24 represent tasks that are scheduled in the existing agenda. The remaining locations are assigned as the possible locations in which the new task could be completed. For Order Specific Tasks, we instead sample 28+(2$\times$ num_task_locations) unique locations on a 50x50 grid. Once again, the first 5 locations are designated as appointment locations. The next 23 represent tasks that are scheduled in the existing agenda. The remaining 2$\times$ num_task_locations are split with the first num_task_locations for the first task and the remaining locations are used for the second task.

After the locations are generated, we generate the start times for the appointments. The first appointment is assigned time 0. For each subsequent appointment $appt_i$ at location $appt_i^{loc}$, it is assigned a time:

$$t_i = t_{i-1} + RandInt(min\_slack, max\_slack) + distance(appt_{i-1}^{loc}, appt_i^{loc}) + 1$$

Here, $RandInt$ generates a number between $min\_slack$ (25 in this case) and $max\_slack$ (100), inclusive. The function $distance$ is the Manhattan distance between the two appointments (the amount of time needed to navigate from one appointment to the next). We add 1 for the time it takes to complete the appointment. Note that the tasks can be completed at any time and thus do not have a specified time assigned *a priori*. Next, we generate the agenda by running the optimal solver with constraints that include the appointments at their given times and locations and all of the tasks and their locations except for the new task. The optimal scheduler ensures that the agenda is as short as possible. With this optimal schedule with 29 (or 28 for Order Specific Tasks) appointments and tasks, we evaluate how well each heuristic approach adds the new task (potentially with multiple possible locations).

**Implementation** Our heuristics were written in Python. For the optimal solver, we used OR-TOOLS CBC-Solver for Python[8][1] with the wrapper to the MIP solver COIN-OR Branch and Cut. We use this solver because it has been successful with the problem formulation we consider here [15] and because the OR-TOOLS wrapper is publicly available, widely-used, and effective. The two heuristics that depend on an optimal solver use the same one. Experiments were conducted on an AMD 5900X CPU with 64GB RAM.

**Metrics for Evaluation** We measured the algorithm run time and the schedule length. The run time is reported directly. We report schedule length directly as well; however, we discuss it in the context of its inefficiency. We define schedule inefficiency as the percentage increase over the optimal schedule length for the same scheduling problem. Note that this number will never be negative because you cannot do better than optimal.

$$\text{schedule inefficiency} = (\frac{\text{schedule length of Heuristic}}{\text{schedule length of optimal}} - 1) \times 100$$

## 5.2 Results

We have two types of experiments, Single Task and Order Specific Tasks, because these are the two most common types of tasks that could have multiple locations. For a Single Task, one could

[1] http://developers.google.com/optimization/

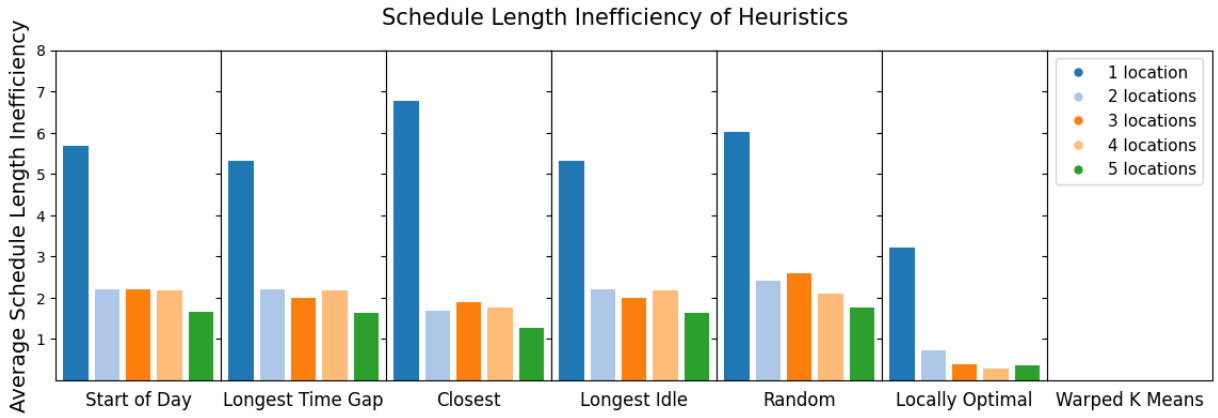


Figure 5.1: We computed the ratio of the schedule length for each heuristic compared to optimal schedule length (schedule length inefficiency) and present the percentage increase in length. Warped K Means is always optimal, while the human strategies perform the worst.

consider going to the ATM or the grocery store. For an Order Specific Task, a simple example would be picking up a postcard from a convenience store and mailing it at one of the post offices. There are cases in which the number of tasks for each of the Order Specific Tasks could vary, such as picking something up from the grocery store to give to your significant other at their office before dropping the other groceries at home. We believe that while we do not test these cases, our results should generalize well to them as well.

## 5.3 Single Task

We first ran the experiment for Single Task because we believed that was the most common type of task that would be added to a day. We created 100 agenda-task combinations for each experimental condition (the number of possible locations in which a new task can be achieved). Figure 5.1 shows the percentage change in each heuristic's schedule length compared to the optimal solver, averaged across each agenda. Figure 5.2 shows the average run time (log scale first then linear scale).

In this set of experiments, we ran a task with between 1 and 5 locations, 10 appointments, and 19 tasks already in the schedule. The slack time between appointments was randomly generated between 25 and 100 and the map size was a 50 by 50 map.

**Schedule Length**

We first analyzed the average schedule length created by each heuristic compared to the optimal length. Warped K Means always found an optimal schedule length. This is because Warped K Means either adds the new task before the last appointment, which will not increase the schedule length, or runs the optimal solver on the entire schedule, which returns the optimal schedule length. The rest of the heuristics all show a similar pattern to each other. In particular, they all have much longer schedules than optimal when there is only one possible task location, and the schedule length approaches optimal as the number of possible task locations increases.

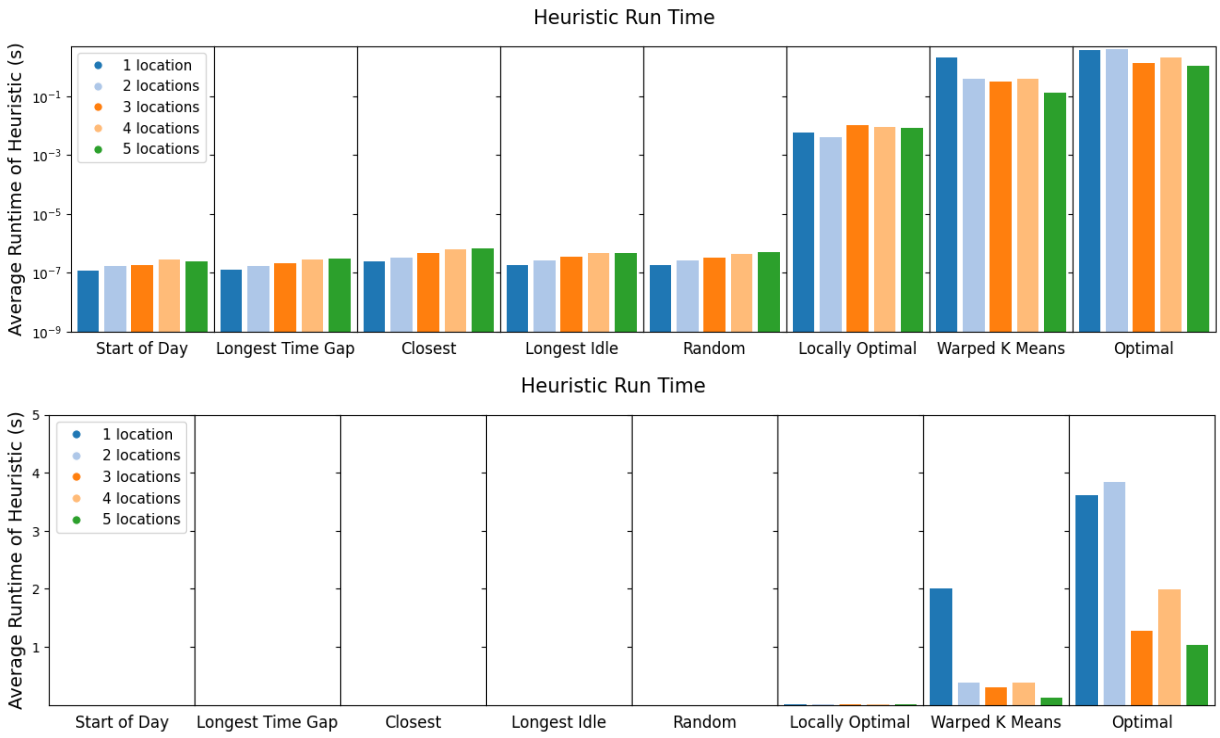


Figure 5.2: Run times for each of the heuristics in seconds on a log scale. Optimal and Warped K Means take the longest, while the human-centered heuristics are the fastest.

This is because there is a higher chance that at least one of the potential task locations will fit between the appointments. We observed that this increase was most dramatic in the transition from 1 to 2 locations and starts flattening after 5 locations.

Looking at the specifics, for the Locally Optimal heuristic, we found that the schedule length is longest when adding a task at one location (M = 3.2% longer than the optimal, SD = 4.7), but the difference drops quickly to M = 0.72% (SD = 2) in the 2-location condition and eventually to 0.36% (SD = 1.5) with 5 possible locations. The human-centered heuristics, performed worse overall because they did not try to reorder existing tasks between appointments to make space to fit the new task. The Longest Time Gap and Longest Idle Time heuristics performed identically to each other; M = 5.32% (SD = 5.8) longer than optimal for 1 location and M = 1.63% (SD = 4.4) for 5 locations. While the two heuristics sort the location-gap pairs between the appointments differently, any tasks added at the end of the day are computed identically. The Closest heuristic performed the worst; we suspect that is because while a new task may be near one appointment, it may actually be far away from the next appointment in the sequence. The Random and Start of Day heuristics performed similarly, likely because both essentially check time gaps randomly.

**Run Time** We then looked at the run time for each of our heuristics and the optimal solver. We found the optimal scheduler to be highly variable in its run time on our agendas. For 1 location, we found that the optimal solver took M = 3.615s (SD = 843). By 5 locations, it decreased to M = 1.03s (SD = 51). The standard deviation was so high because the scheduler

often found the solutions very quickly, but it took an extremely long time to calculate all of the exponential number of possible orderings of the tasks within the appointments for some agendas.

For Warped K Means, we found that the run time decreases more steeply than for optimal; 1 location ran for M = 2s (SD = 105.7), 2 locations decreased to M = 0.381s (SD = 179), and 5 locations decreased to M = 0.12s (SD = 77.7). Because the algorithm depends on the optimal solver, it also had extremely high variance in run time.

For Locally Optimal for 1 location it took M = 0.0058s (SD = 2.26) With 2 locations, it dropped to M = 0.00408s (SD = 0.822). Finally at 5 seconds, it increased to M = 0.01 (SD = 4.58). We note that gap-optimal timings did not significantly change with 1 location or multiple locations. This is likely because Locally Optimal does not benefit as much from the short-cutting, instead having to try each of the locations sequentially. Thus, the likelihood of short-cutting is overtaken by the fact that Locally Optimal has to call the optimal solver many more times for each time-gap. It should also be noted that for gap-optimal, the growth is polynomial in the schedule length and not exponential, with it only growing exponentially in the number of tasks in between appointments. Thus, with very long schedules but with only a few tasks between appointments, gap-optimal will be very fast.

The human-centered heuristics all had similar run times, which is not surprising given their similar algorithms. Unlike the optimal-based approaches, the algorithms take longer to run when the number of task locations increase because there are more possibilities to sort. For 1 location, we found that Longest Time Gap performs the best, M = $127ns$ ( SD = $4270$), and Closest performs worst, M = $247ns$ (SD = $4970ns$). For 5 locations, Longest Time Gap also performs the best, M = $294ns$ (SD = $1670$), and Closest is again worst, M = $665ns$ (SD = $18300ns$). These times are up to 7 orders of magnitude faster than the optimal scheduler.

The algorithms based on human-centered heuristics performed very fast (in hundreds of nanoseconds), but they resulted in increased schedule lengths. Our results show that the schedule length becomes much closer to optimal as the number of possible task locations increases because the algorithms are more likely to be able to fit a new task in between existing appointments. However, these algorithms also become slower as the number of task locations increases. A middle ground is our Locally Optimal heuristic, which is several orders of magnitude faster than the optimal scheduler (solving schedules in thousandths of a second) while finding schedules that are only 0.5-3% longer than optimal on average.

## 5.4 Order Specific Tasks

We also examined Order Specific Tasks because it would be another common use case for the scheduling system. Once again, we created 100 agenda-task combinations for each experimental condition (i.e., number of possible locations that a new task can be achieved).

However, in this case, we only ran 1, 2, and 5 num_task_locations because of computational constraints due to the more complicated problem. We selected these numbers to be relatively representative of the complete picture. For the currently scheduled tasks, we used 10 appointments again, but this time we included only 18 tasks to account for the 2 new tasks to be inserted for a total of 20 tasks. The slack time between appointments was randomly generated between 25 and 100 and the map size was 50 by 50.

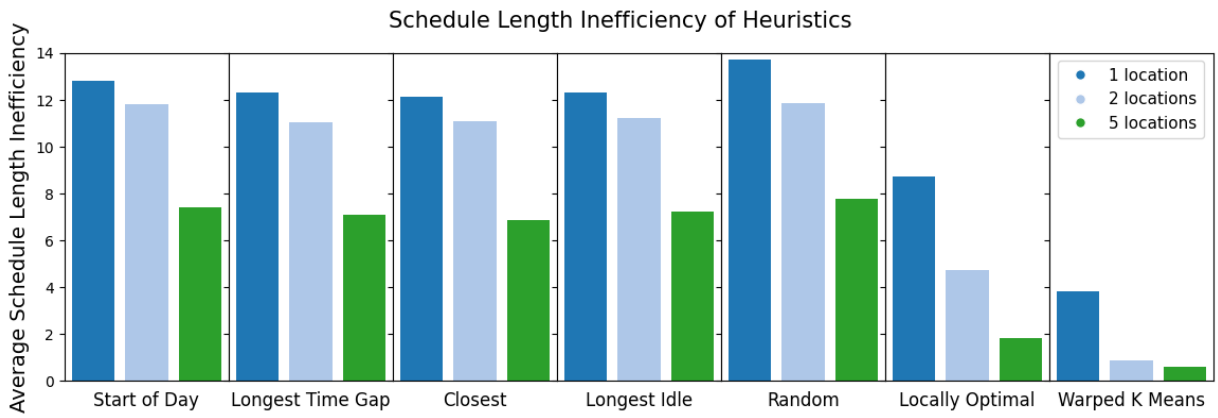


Figure 5.3: We computed the ratio of the schedule length for each heuristic compared to the optimal schedule length (schedule length inefficiency) and present the percentage increase in length. Warped K Means is the closest to optimal, while the human strategies perform the worst.

Something to note is that in this set of experiments, unlike the previous set, Warped K Means is no longer optimal because we insert one task at a time. Therefore, if the first task has the nearest cluster (out of multiple potential clusters) towards the end of the day, that cluster might be selected and force the second task to be inserted at the end of the day.

**Schedule Length**

We then looked at the inefficiency in the schedule length in the Order Specific Task case.

Unlike in the task case, Warped K Means is no longer optimal for Order Specific Task scheduling. This is because we insert one task after the other. However, it still performs the best of all non-optimal heuristics at M = 3.81% (SD = 2.18). This decreases with 2 locations to M = 0.86 % (SD = 9.97) and to M = 0.57% (SD = 6.01) for 5 locations. The reason that Warped K Means is no longer an optimal solution is because the second task has to start its day where we have inserted the first task. If the first task is inserted into a late location, this could result in inefficiency by limiting the possibilities for the second task. In some cases, the second task might end up scheduled at the end of the day.

For Locally Optimal, we find that it follows a similar pattern to in the task case where for 1 location we have M = 8.69% (SD = 6.78). With 2 locations it goes down to M = 4.73% (SD = 5.49) Finally with 5 locations it goes down to M = 1.80% (SD = 2.99).

For human-centered heuristics we found that they once again performed reasonably similarly. For 1 location, closest did the best at M = 12.14% (SD = 7.44) whilst Random did the worst at M = 13.70% (SD = 8.71). For 2 locations, longest time did the best with M = 11.04% (SD = 9.75) wheras random did the worst at M = 11.82% (SD = 10.27). Finally for 5 locations, closest once more came out on top at M = 6.86% (SD = 7.10) and random did the worst at M = 7.74% (SD = 8.29). It was notable that unlike the task, the task ordering makes random perform much worse than start of day given that random could start at a really late time gap to insert in first, thus causing the second task to have nowhere to go.

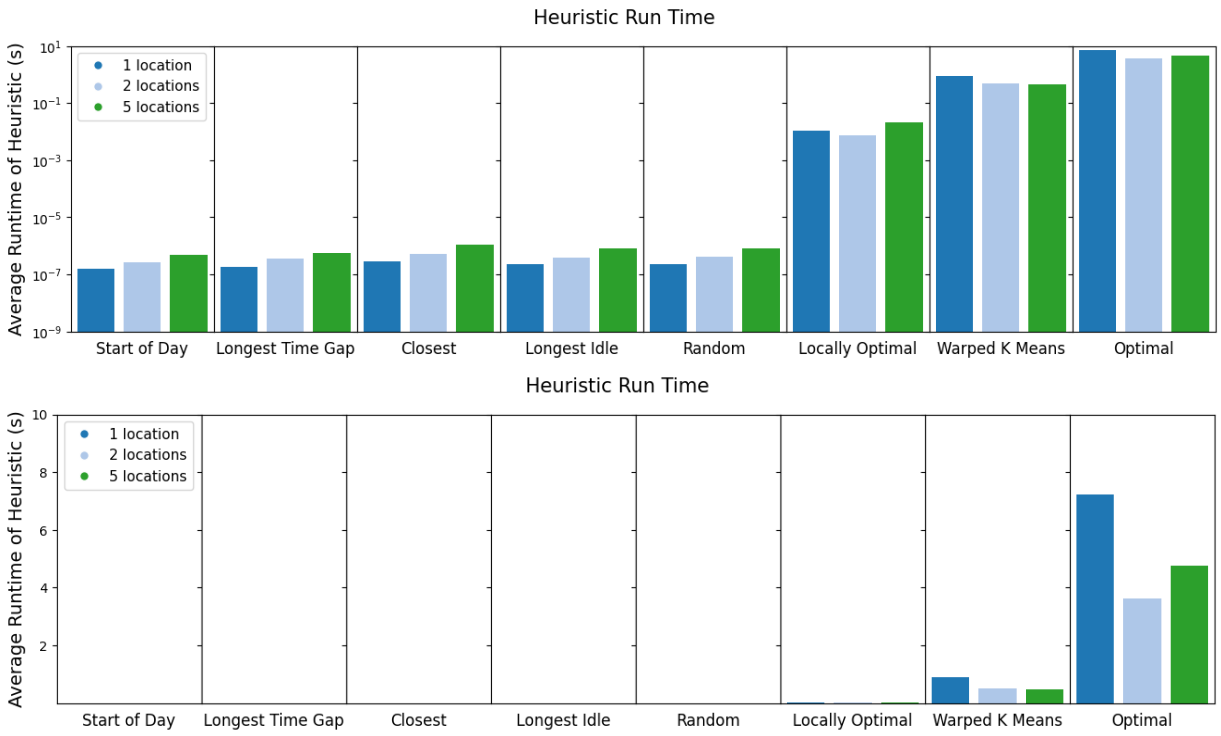


Figure 5.4: Run times for each of the heuristics in seconds. Optimal and Warped K Means take the longest, while the human-centered heuristics are the fastest.

**Run Time**

We looked at the run time in the Order Specific Task case to see how much faster the human-centered heuristics would be.

Once again we found that the optimal scheduler to be very variable in the run time of the agendas. For 1 location we found that it took a M = 7.23s (SD = 4460) With two locations it decreased to M = 3.61s (SD = 1333). At 5 locations it increased back to M = 4.76s (SD = 2004). We expect that like the first case the standard deviation is very high because it takes a long time to calculate all possibilities but there are many possible short-circuits possible for many computations such as if everything fits before the last appointment. Furthermore, we hypothesize that the mean run time decreased then increased because of the higher possibility of short-circuiting with multiple locations trading off with the increased complexity of the problem having more locations.

For Warped K Means, we once again find that it is faster than optimal in all cases. For 1 location it had a M = 0.9s (SD = 237). For 2 locations this decreased to M = 0.5s (SD= 142). With 5 locations, this continued decreasing to M = 0.45s (SD = 151). We find that unlike the Optimal scheduler in that with more locations, the speed only increases.

For Locally Optimal with 1 location we found that it took M= 0.01s (SD = 4.03). With 2 locations it decreased to M = 0.007s (SD = 1.79). For 5 locations it decreased much further up to M= 0.02s (SD = 8.10). This is similar to what happened in the optimal scheduler except that with

5 locations, gap-optimal appeared to perform much worse. This would suggest that gap-optimal is not a good option for many locations.

For human-centered heuristics, we found that once again they performed pretty similarly to each other. Once again for 1 location Longest Idle did the best with M = $127ns$ (SD = 4270) while closest did the worst with $277ns$ (SD = 4803). For 2 locations surprisingly Start of Day did the best at, $256ns$ (SD = 6068) whilst closest still did the worst at $499ns$ (SD = 9923). For 5 locations, Start of day maintains it's lead at $496ns$ (SD = 17823) whilst Closest did the worst at $1109ns$ (SD = 23294). We hypothesize that Start of day started doing much better in the multi-location case due to removing the overhead of sorting the schedule into a specific order with multiple locations.

As for computational heuristics we note that Warped K Means operated significantly (almost one order of magnitude) faster than optimal in all cases but it is no longer optimal. This is because we insert the first task and then the second task which means if there were multiple places the first task can go it could go later in the day, thus constraining the search of a location for the second task to a smaller space and making it no longer optimal. Thus, unlike in the previous case, there is a speed-time trade off for each of the cases of Optimal, Warped K Means Locally Optimal and the human-centered heuristics. This changes our recommendation from using Warped K Means instead of optimal in all cases to deciding between each of the heuristics depending on what the needs of the user is.

Once again, the human-centered heuristics performed extremely fast but they had an increase in schedule length and with multiple locations, these heuristics becomes closer to optimal. However, unlike in the task case, with Order Specific Tasks the cost of the extra schedule length became much more significant at around 8-12 percent. This is much more significant which would mean that our recommendation would be to change the usage of the human-centered heuristics in Order Specific Tasks to cases, where we can tolerate inefficiencies but would like to return an answer as quickly as possible. For the Human heuristic to use, we would recommend longest time heuristic since it appears to generalize the best with closest, the other one that came on top for most efficient, seeming to generalize much worse.

## 5.5 Comparing the two

To compare the two type of tasks, we graphed only Longest Time gap for our human-centered heuristics as it was one of the ones that performed the best alongside Warped K Means, Locally Optimal and optimal. The decision was made to only show one of the human-centered heuristics because they preformed similarly and Longest Time gap was viewed as the best option.

**Schedule Length**

We first compared the schedule length inefficiency of the Single Task and the Order Specific Task.

For schedule length we could see that for the Longest Time heuristic, the schedule length inefficiency for the Order Specific Tasks was a bit more than double of the schedule length inefficiency of the regular task for 1 location. This makes sense because inserting two tasks should cause about twice as much inefficiency as compared to inserting one in the first place and the additional inefficiency likely comes from the sorting heuristic itself as while it does come

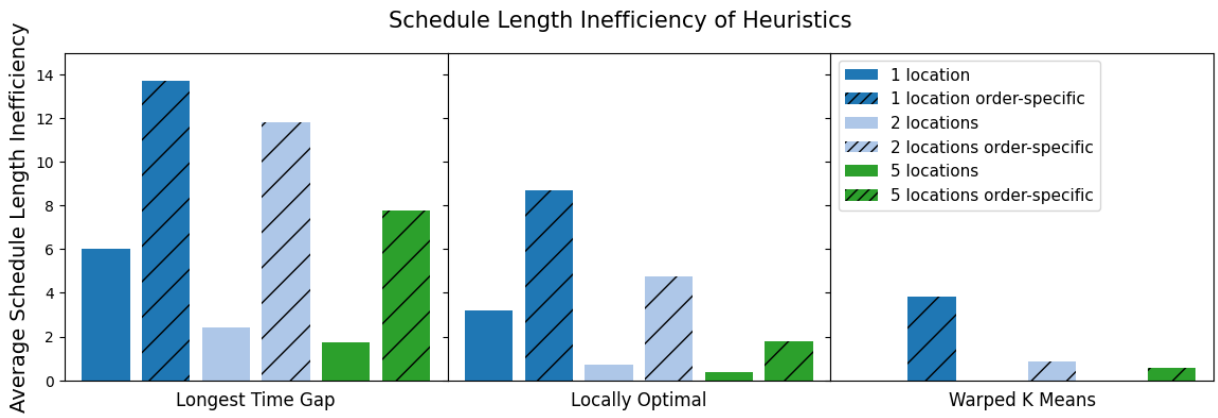


Figure 5.5: We computed the ratio of the schedule length for each heuristic compared to optimal schedule length (schedule length inefficiency) and present the percent longer than optimal.

with benefits, we could end up picking a slot later in the day if there was both a slot earlier and later in the day, causing the second task to have to be scheduled later as well. Thus, this can cause extra inefficiencies given a lower amount of time-gaps that the second task can check. For multiple locations, it is clear that the inefficiencies become much worse. This is likely because with multiple locations, the first choice by the heuristic affects the second much more and its more likely to choose a bad first option with multiple locations.

For Locally Optimal, we observe a similar behavior where the scheduling for 2 tasks takes more than double for 1 location and for multiple locations, the inefficiencies become much worse for Order Specific Tasks.

Finally for Warped K Means, unlike in the Single Task case where it is guaranteed to be optimal, in the multi-location case if there are multiple locations to place the first task, the cluster chosen could be nearer to the end of the day and thus cause the second task to be unable to be scheduled where it needs to go for optimal since the first task was scheduled after than and thus cause inefficiencies. Thus, we see that there is only inefficiencies in the Order Specific Tasks.

**Run Time**

We then compared run times for both Single Tasks and Order Specific Tasks.

For run times, it was interesting to note that for Longest time gap, it takes nearly about double the time for the first case. This makes sense given that it would take time to insert the first task then about the same amount of time again for the second task. It makes sense that it is nearly double but not quite double because the first task is likely to remove some of the search space since it is unlikely that it was scheduled at the start of the day. This is similar for Locally Optimal for 1 task. The trend continues for 2 and 5 locations.

What was interesting to note is that this no longer remained true for Warped K Means in the 1 location case. We see that the Order Specific Task took significantly less time than the normal insertion. This is likely because when inserting the 1st task since there is only 18 other task rather than 19 already there it is easier to insert without searching the entire space. Then for the second task, the search field becomes restricted by the first task start time which means it is

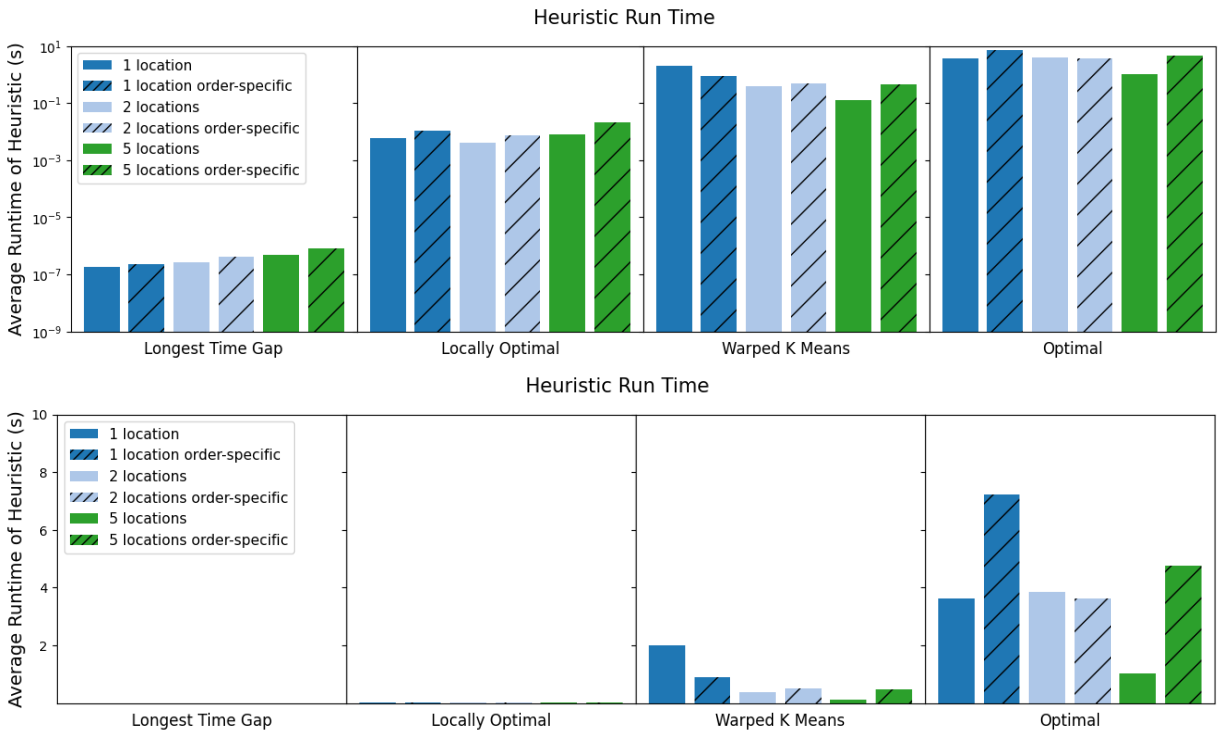


Figure 5.6: Run times for each of the heuristics in seconds on a log scale. The first 3 columns are from the Single Task and then the next 3 from the Order Specific Tasks.

simpler. This is supported by it switching back to Order Specific Tasks taking longer than the singular task most likely because it becomes easier to insert the first task due to multiple possible locations and the 1 difference in current task matters less.

Finally for optimal we see that Order Specific Tasks takes about twice as long as a Single Task. This is likely because the additional constraint takes more time to calculate. For 2 locations we see that Order Specific Tasks becomes faster and then at 5 locations, regular tasks take over again. This is likely due to the ability to short circuit by eliminating the need to check any combinations where the second task appears before the first balancing against the fact that with there are more total locations than with the singular task and thus more possible combinations.

# Chapter 6

# Discussion

We found that there was generally a speed efficiency trade-off for scheduling heuristics. There was an exception in the Warped-K-Means Single Task case where it offered a speed increase for no efficiency cost. The human-centered heuristics did extremely well in the Single Task case but performed worse in the order-specific tasks. Locally Optimal did better than the human-centered heuristics at the cost of being orders of magnitude worse in speed and Warped K Means was even slower yet provided the least inefficiency in the order-specific task case and perfectly efficient schedules in the Single Task case. We therefore find that, depending on the circumstances, our human-centered heuristics can be a great tool to use to schedule extra tasks in a day.

Our results represent an important contribution to the field because we have presented new heuristics that can very quickly add tasks to a person's schedule. This can be extremely useful for the day to day usage of people where a few percentage inefficiencies is less likely to matter as much as getting a quick answer so that the system can feel responsive for the user interactions. This responsiveness, especially on a device the user might use to schedule during the day, such as a phone which would have limited computational power, would allow the user to make quick and effective decisions about where they would need to go next, rather than wasting time waiting for the computation to complete. It can be very important for this to be the case given that if one had a tight schedule, the extra time waiting for the computation could lead to being late to the next appointment. Thus, this method would allow them to get a much better result than the trivial solution of inserting the task to the end of the day (meaning just do the task once everything else for the day is done) whilst still being almost instantaneous to compute.

In our human study results, participants overwhelmingly reported using the Start of Day strategy. This method is simple: it is easy to keep track of which gaps have been checked and does not require any distance or time computations. Yet, overall, participants were incorrect 15% of the time with this strategy. Our experimental results of human-centered heuristics showed that they are fast to compute and produce agendas within 2% of optimal when there are 2 or more possible locations for Single Tasks and 10% in order-specific tasks. While the human-based heuristics performed worse than the other heuristics in terms of schedule length, they are likely more intuitive for a human to understand when explained. Thus, the human-centered heuristics could also be reasonable for an automated system to use when assisting humans. We also noted that while Start of Day heuristic clearly did worse than our other human-centered heuristics, it

did not do that much worse and the ease of computation for a human, or even for a machine not having to re-sort the heuristics, could still make Start of Day a reasonable strategy.

One limitation for this paper is that we kept slack time (the idle time between adjacent appointments where you are not traveling) random in the same range in all our experiments. We used a random slack time in the range [25,100] for our experiments. The slack time needed depends on the number of tasks to schedule and the size of the grid. We aimed for some of the tasks to be able to fit within the day, but not so many that would cause the new tasks to never appear at the end of the day. We found that all of our human-centered heuristics perform much worse with a shorter slack time because they spend a large proportion of their time searching gap-location pairs within the existing agenda often only to discover that the task belongs at the end of the day. For a similar reason, there was also much more variance in their run time when we set the minimum slack (25 in our experiments) to 0. Both Warped K Means and Locally Optimal also take longer to run with short slack times because they run the optimal scheduler several times on increasingly larger portions of the agenda until finally scheduling the new task at the end of the day. While we found a suitable slack parameter for our studies, more work is needed to analyze this parameter more closely.

Another limitation is that we kept our percentage of appointments to task constant. However, given that there would still be the same number of time gaps, we expect our human-centered heuristics to do the same time wise given that there would be the same number of time gaps regardless of the ratio. Also given that the percentage of appointments vs tasks was 16% for our experiments which shown similar results to [24], and as such, we would expect that Warped K Means would be even faster as the percentage of appointments increases. Using the same results, the same would be true for the optimal scheduler. However, for Locally Optimal it isn't as clear how changing the percentage of appointments would affect the scheduling system since whilst there would be smaller problems to solve costing less time, there could potentially be more problems since there would be more appointment pairs given that there are more appointments. Intuition would suggest that since one is decreasing exponentially while increasing linearly the time to run should also decrease but further work should be done to confirm this.

One possible future work would be to explore how humans like the human-centered heuristics as compared to the black box heuristics. Given that our human-centered heuristics are human inspired, it is likely to be more intuitive for people when they see how their schedule got changed as compared to when an optimal scheduler does it since it may change a lot of the schedule as compared to shifting things around which is a much more human thing to do. Thus, it would be interesting to explore this further. I would complete this study by showing the user a schedule and then a new schedule and having them rank what they like about the schedule on a likert scale. We would do this instead of having all the schedules side by side for comparison and to pick which one is preferred because that could introduce a confounding factor where people would only look at the time to determine which one to use. By using different schedules and a likert scale, this encourages them to think more deeply about which one they prefer.

Another future work would be to integrate our heuristics to a scheduler system for people to use. Given that we have these useful new heuristics, it would be important to integrate this into a scheduling system for people to use to simplify their lives and receive feedback on how they think the scheduler and the heuristics perform. We would do this by creating a scheduler and first running a user study of having people test out the scheduler and reporting their results. If we

were successful preliminary, then running a longer-term user study of having people live their lives while using the scheduler when adjusting their day for a new event. This would allow us to see how people would use or not use the scheduler on a day to day basis and how much the scheduler would help the user schedule.

One area of future work would be to do more studies on how people strategize to solve different problems. For example we could explore strategies that people use to solve more complicated tasks such as inserting multiple unrelated tasks into a day where there could be locations where either of the tasks fit. This would be very interesting to see if humans will come up with new heuristics to parallelize the task or simply insert one task and then start on the other. Given that humans can retain knowledge of the schedule after inserting one task and can keep the other task at the back of their mind while performing one, I would hypothesize that humans would parallelize the task rather than doing them sequentially.

# Bibliography


[1] Anastasios Alexiadis and Ioannis Refanidis. Optimizing individual activity personal plans through local search. *Ai Communications*, 29:185–203, 08 2015. doi: 10.3233/AIC-150680. 2

[2] Jacob Bank, Zachary Cain, Yoav Shoham, Caroline Suen, and Dan Ariely. Turning personal calendars into scheduling assistants. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, page 2667–2672, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450310161. 1

[3] Sarah Beech, Erik Geelhoed, Rachel Murphy, Julie Parker, Abigail Sellen, and Kate Shaw. The lifestyles of working parents: Implications and opportunities for new technologies. Technical report, HP Tech Report HPL-2003-88 (R. 1), 04 2004. 2

[4] Pauline M Berry, Melinda Gervasio, Bart Peintner, and Neil Yorke-Smith. Ptime: Personalized assistance for calendaring. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(4):1–22, 2011. 1, 2

[5] John Lawrence Bowman. *The day activity schedule approach to travel demand analysis*. PhD thesis, Massachusetts Institute of Technology, 1998. 1

[6] Mike Brzozowski, Kendra Carattini, Scott R. Klemmer, Patrick Mihelich, Jiang Hu, and Andrew Y. Ng. grouptime: Preference based group scheduling. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1047–1056, 2006. ISBN 1-59593-372-7. 1, 2

[7] Justin Cranshaw, Emad Elwany, Todd Newman, Rafal Kocielnik, Bowen Yu, Sandeep Soni, Jaime Teevan, and Andrés Monroy-Hernández. Calendar. help: Designing a workflow-based scheduling agent with humans in the loop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2382–2393. ACM, 2017. 2

[8] John Forrest, Ted Ralphs, Stefan Vigerske, LouHafer, Bjarni Kristjansson, jpfasano, EdwinStraver, Miles Lubin, Haroldo Gambini Santos, rlougee, and Matthew Saltzman. coin-or/cbc: Version 2.9.9, July 2018. URL `https://doi.org/10.5281/zenodo.1317566`. 5.1

[9] Tommy Gärling, Tomas Kalen, Joakim Romanus, Marcus Selart, and Bertil Vilhelmson. Computer simulation of household activity scheduling. *Environment and planning A*, 30 (4):665–679, 1998. 1, 2

[10] Matthew Gombolay, Reed Jensen, Jessica Stigile, Sung-Hyun Son, and Julie Shah. Apprenticeship scheduling: Learning to schedule from human experts. In *Proceed-*

*ings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press/International Joint Conferences on Artificial Intelligence, 2016. 2

[11] Geir Hasle and Stephen Smith. Directing an opportunistic scheduler: An empirical investigation on reactive scenarios. *IFIP Advances in Information and Communication Technology*, pages 1–11, 11 1996. 4.8

[12] H Henseler. From reactive to active scheduling by using multi-agents. In *Artificial Intelligence in Reactive Scheduling*, pages 12–18. Springer, 1995. 4.8

[13] Stephen C. Hirtle and Tommy Gärling. Heuristic rules for sequential spatial decisions. *Geoforum*, 23(2):227–238, 1992. ISSN 0016-7185. doi: https://doi.org/10.1016/0016-7185(92)90019-Z. URL `https://www.sciencedirect.com/science/article/pii/001671859290019Z`. 2

[14] Tang JiaHua and Du Zhang. Smartcalendar: Improving scheduling through overcoming temporal inconsistencies. In *2018 IEEE 17th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, pages 30–37, 2018. doi: 10.1109/ICCI-CC.2018.8482075. 2

[15] Imdat Kara and Tusan Derya. Formulations for minimizing tour duration of the traveling salesman problem with time windows. *Procedia Economics and Finance*, 26:1026–1034, 2015. 4.1, 5.1

[16] Luis A. Leiva and Enrique Vidal. Warped K-Means: An algorithm to cluster sequentially-distributed data. *Information Sciences*, 237:196–210, 2013. 4.10

[17] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. Towards next generation touring: Personalized group tours. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, pages 412–420, 2016. 2

[18] Pragnesh Jay Modi, Manuela Veloso, Stephen F. Smith, and Jean Oh. Cmradar: A personal assistant agent for calendar management. In *Agent-Oriented Information Systems II*, pages 169–181, 2005. 1, 2, 3.4

[19] Carman Neustaedter, AJ Brush, and Saul Greenberg. The calendar is crucial: Coordination and awareness through the family calendar. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(1):1–48, 2009. 2

[20] P Rayudu, Kalahasthi Neelima, Sampada Gulavani, Deepak Kori, J Arundathi, and Kabir Kharade. Artificial neural networks based smart routine plan artificial neural networks based smart routine plan. *Turkish Online Journal of Qualitative Inquiry*, 12:10184–10194, 07 2021. 2

[21] Ioannis Refanidis and Anastasios Alexiadis. Deployment and evaluation of selfplanner, an automated individual task management system. *Computational intelligence*, 27(1):41–59, 2011. 2

[22] Ioannis Refanidis and Neil Yorke-Smith. A constraint-based approach to scheduling an individual's activities. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 1 (2):1–32, 2010. 1, 2

[23] Ioannis Refanidis, Christos Emmanouilidis, Ilias Sakellariou, Anastasios Alexiadis, Remous-Aris Koutsiamanis, Konstantinos Agnantis, Aimilia Tasidou, Fotios Kokkoras, and Pavlos S Efraimidis. myvisitplanner gr: Personalized itinerary planning system for tourism. In *Hellenic Conference on Artificial Intelligence*, pages 615–629. Springer, 2014. 2

[24] Stephanie Rosenthal and Laura M. Hiatt. Human-centered decision support for agenda scheduling. In *AAMAS '20*, page 1161–1168, 2020. 1, 2, 3.3, 4.10, 6

[25] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming — CP98*, pages 417–431, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-49481-2. 4.8

[26] Luis Zabala, Cristina Perfecto, Armando Ferro, and Juanjo Unzilla. Integrating automatic task scheduling and web-based agenda in a virtual campus environment. *International Conference on Information Technology Based Higher Education and Training*, 01 2001. 1, 2