

WolfPlanner 2.0

Abhishek Bhardwaj
North Carolina State
University
abhardw3@ncsu.edu

Madhu Vamsi Kalyan
Machavarapu
North Carolina State
University
mmachav@ncsu.edu

Sanya Kathuria
North Carolina State
University
skathur2@ncsu.edu

Shivam Chamoli
North Carolina State
University
schamol@ncsu.edu

ABSTRACT

With the number of tasks increasing day-by-day, it is becoming difficult for students to manage so many tasks considering the varying amount of time available and required to complete them. Students have to manage many tasks such as attend classes, complete assignments before deadlines, work on projects, attend meetings and prepare for interviews. With such a large number of tasks lined up for the student to do in a fixed time frame, it becomes very hard for the student to come up with an ideal time table which would help the student complete all the activities and also maintain its quality to an optimum level. It requires continuous and active effort by the student to create weekly schedules leading to wastage of valuable time. Manual effort in scheduling might lead to poor planning which in turn will result in excessive workloads.

In today's age where time is a critical asset for a student's success, we have come with an idea for replacing and reducing human effort by a bot which generates a personalized schedule for the student. The bot makes a schedule for the students taking into account their deadlines, breaks and dependencies between tasks. WolfPlanner, an interactive chatbot specifically made for NC State students that generates a schedule for the student based on upcoming assignments, projects, interviews, events, etc (information about which will be given to the system by the student). Furthermore, to keep up with the current technology trend of interactive assistants, we have integrated this service with Slack thereby avoiding downloading of stand-alone application. For now, we restrict our domain of users to students for ease of data collection and implementation. In the existing system, we have faced numerous challenges, including usability, multi-agent coordination, scalable constraint reasoning, robust execution, and unobtrusive learning. Our research advances basic solutions to the fundamental problems; however, inte-

grating Personal task scheduler into a deployed system has raised other important issues for the successful adoption of new technology. As a personal assistant, it must integrate easily into a user's real environment, support their normal workflow, respect their authority and privacy, provide natural user interfaces, and handle the issues that arise with deploying such a system in an open environment. To mediate that, we have come up with various improvements in the previous team's scheduling algorithm.

Keywords

Scheduler, Chatbot, Calendar, Time-table, Planner, Slack, Meeting Scheduler,

1. INTRODUCTION

In this paper, we discuss WolfPlanner - a personalized task scheduling bot for students, which generates a weekly schedule of tasks and guarantees completion of tasks within the respective deadlines without compromising on the quality of work. In order to get the best out of their education, it is important that students focus on important tasks such as their coursework and studies and spend less time on wasteful tasks like creating a time-table. The best way forward is to automate such tasks and WolfPlanner does exactly the same. Today, it is a world of personal interactive assistants and according to Gartner's report[1], 85% of customer interactions will be managed without a human by 2020. So we have developed a chatbot that is able to assist the user in scheduling a wide of range of day-to-day tasks (for a student) which includes classes, assignments, project meetings, discussions, readings to name a few. The application is a chatbot interface which will take in all of these conditions as inputs from the students to help them manage their time efficiently. While generating the task schedule, we have also taken care of time spent in activities like sleeping, eating, buffer time between two tasks such as commuting time from classroom to the destination where a project meet is scheduled and ignoring meagre chunks of time for major tasks. The bot will also give the student a free time slot in which the student can do any other activity which he wishes to do. In universities, students are involved in many group activities, and scheduling these meetups are often challenging due to conflicting schedules. They take up a lot of time and lead to hassles. So, having a collaborative approach is needed.

The bot is of a great help as it has the data of the weekly schedule of all its users. The schedule generated will help the students come up with a suitable time for arranging meetings. All user will know their free slots and can agree on a common time accordingly.

The motive of our work was to develop a chatbot with which a user can communicate, provide data, ask for change in schedule, etc. thus saving time for schedule generation. A chatbot feature also guarantees ease of use of application. The bot generates schedules on a weekly or biweekly basis as per user preference. The user requests a set of tasks every week and the priority is set according to deadline. As per the user preference, the bot will generate an appropriate schedule taking into account all of the details. Further, this generated schedule may be exported to Google Calendar for easy access across all the devices. We have hosted our database on Mlab.com where the user data is stored on MongoDB. Mlab provides scalability and reliability to our system, and security to user's data. The user details are stored as objects on the NoSQL MongoDB environment making it convenient to access the details of the user who is logged in without any requirement of costly join operations which would have been needed in case of relational databases. Also storing user data in tables might lead to redundancy and lot of NULL data. Hence, we decided to use MongoDB as our database system. The advanced version of WolfPlanner works with its user and also keeps into account other users, to schedule meetings and commitments in its user's calendar. It can negotiate and provide the users with useful information about upcoming events while protecting her privacy, and learn about their prioritization preferences. To achieve these goals, we plan to integrate commercially available calendaring tools and project meetings, goal directed process management, and preference learning.

2. MOTIVATION

A major problem in student's life is managing homeworks, projects, interviews, meetings etc. and still get some free time. Our study revealed that majority of survey respondents (87%) want an application/bot which generates a schedule for them. However, a lot of people (67%) don't use any calendar for organizing their day. This shows that people wanted an application for a functionality which other calendars do not provide. Hence we came up with the idea to fulfil the user's requirements by creating an interactive bot which generates schedules for the user and also organizes meeting times for them. This will help the users manage their time efficiently and work productively. Moreover, we will be extending it to include priorities of tasks, breaks and neglection of small chunks of free time for tedious tasks. Furthermore, we will be extending the scheduling algorithm to accommodate these changes.

We can point to two main reasons why we need to change the existing system. The first is a failure to take into account the intensely personal nature of the human time management problem. Most individuals, even the busiest knowledge workers, are reluctant to relinquish control over the management of their own time. Further, people have different preferences and practices regarding how they schedule their time, how they negotiate appointments with others, and how much information they are willing to share when doing so. To be successful, a time management solution must thus address these basic issues of authority, privacy,

and preferences. These type of features are not handled as of now. The second reason is that adoption of such a system is contingent on its practical usability including important user interface issues within a real-world deployment.

Despite the proliferation of calendar tools to organize, display, and track commitments, most individuals still expend a considerable amount of effort managing their schedules. The research community has primarily focused on automated meeting scheduling: finding feasible time slots for meetings given a set of requirements on participants, times, and locations.

3. LITERATURE REVIEW

Despite the success of calendar tools in organizing and tracking activities, a lot of time is spent by people on managing their time properly. Berry et al.[2] came up with PTime, a personalized time management agent which reduces this problem of time management considerably, when it comes to arranging meetings. PTime schedules meetings based on factors like participants, time and locations. Our bot will go a step further and automatically schedule each and every task of a user along with the meetings. The biggest challenges in any task planner are the constraints in schedules and which scheduling method is used.

1. Constraints in scheduling
2. Preemptive tasks
3. Penalty for errors in scheduling

Traditional algorithms face a lot of problems when it comes to preemptive scheduling. For example, schedule quality of FCFS scheduling algorithm is highly sensitive to the preemption strategy used. Schwiegelshohn et al.[3] demonstrated that by doing certain improvements in FCFS it can be better than any of the other preemptive algorithms but the parameter settings and efficiency depends heavily on the workload. As our task scheduler deals with multiple tasks which can arrive at any time, using FCFS might lead to compromising efficient task scheduling as our workload itself is dynamic in nature. After reviewing some papers we found that the algorithm implemented by L. J. Wilkerson et al.[4] will be an effective algorithm for our task scheduler. We then compared this algorithm with First Fit algorithm[5] and found that First Fit Algorithm will be quite suitable too. For the Improvement, we are planning to execute the Multi-Level Feedback Queue scheduling algorithm along with the necessary accommodations for priorities of tasks, breaks and extension.

For the Improvement, we are planning to execute the Modified Multi-Level Feedback Queue scheduling algorithm along with the necessary accommodations for priorities of tasks, breaks and extension to a weekly schedule.

3.1 IMPROVEMENTS

3.1.1 Existing System

The current system has a chatbot with which a user can communicate, provide data, ask for change in schedule, etc. thus saving time for schedule generation. A chatbot feature also guarantees ease of use of application. The bot generates schedules on a daily as per user preference. The user requests a set of tasks every week and the priority is set according to deadline. As per the user preference, the bot will generate an appropriate schedule taking into account all of the details. Further, this generated schedule may be

exported to Google Calendar for easy access across all the devices.

The database on Mlab.com where the user data is stored on MongoDB. Mlab provides scalability and reliability to our system, and security to user's data. The user details are stored as objects on the NoSQL MongoDB environment making it convenient to access the details of the user who is logged in without any requirement of costly join operations which would have been needed in case of relational databases. Also storing user data in tables might lead to redundancy and lot of NULL data. Hence, we decided to use MongoDB as our database system.

The advanced version of WolfPlanner works with its user and also keeps into account other users, to schedule meetings and commitments in its user's calendar. It can negotiate and provide the users with useful information about upcoming events while protecting his/her privacy, and learn about their prioritization/ preferences. To achieve these goals, we plan to integrate commercially available calendaring tools and project meetings, goal directed process management, and preference learning.

Basic Flow

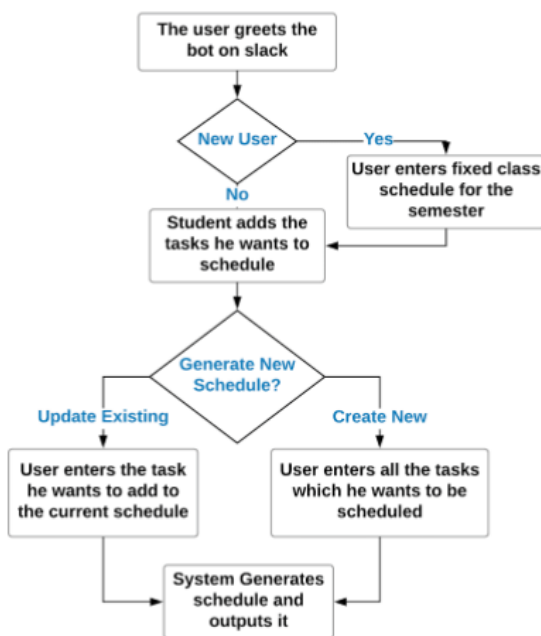


Figure 1: Flow of the system

3.1.2 Correct the implementation of daily scheduling

Right now the only scheduling implemented is for daily scheduling, which does not schedule properly either. After testing the application thoroughly, we found out that a 10 hours schedule + 8 hours schedule in the same day is not scheduled correctly. Moreover, the user is allowed to create schedules for a date that has passed.

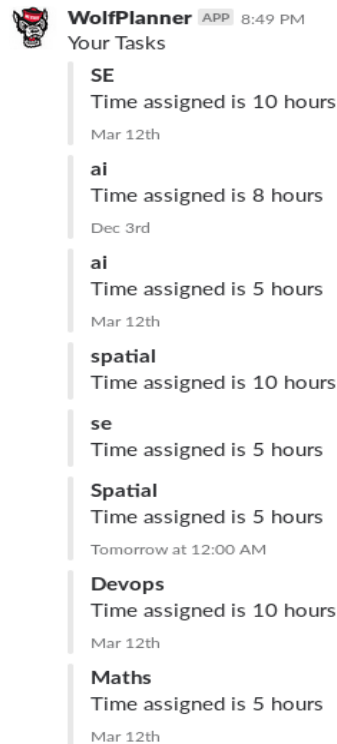


Figure 2: Schedule given by a student, who has 1 submission the next day and 4 submissions a week later

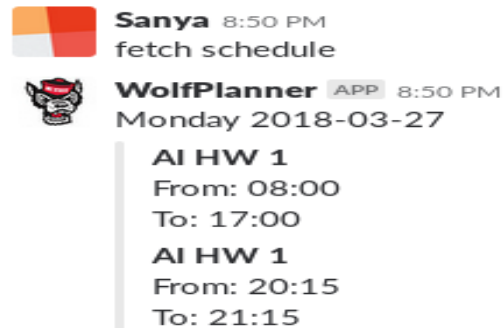


Figure 3: Output of WolfPlanner according to the present system

3.1.3 Adding the Project meetings

The bot has the data of the weekly schedule of all its users. So it can suggest a common free slot so that a meeting can be organized. It can also include the meeting venue and time needed to reach that venue. If a group has to meet to work on a particular project, this feature will give the group members an idea to plan the stages of the project as they know beforehand when the meetings can be scheduled.

Days/Time	12-2 AM	2-4 AM	4-6 AM	6-8 AM	8-10 AM	10-12 PM	2-4 PM	4-6 PM	6-8 PM	8-10 PM	10-12 PM
Mon 05/03					class	Spatial	Lunch	Spatial + break	Spatial + break		
Tues 06/03						SE	Lunch	DevOps + break	SE + break	DevOps + break	
Wed 07/03					class	class	Lunch	DevOps + break	SE + break		
Thurs 08/03						DevOps	Lunch	Maths + break	AI + break	SE + break	
Fri 09/03					class	class	Lunch	SE + break	DevOps + break	Maths + break	AI + break
Sat 10/03											
Sun 11/03											

Last two days are for ad-hoc events. Break signifies a break of 30 minutes i.e 15 minutes per hour

Figure 4: sample schedule which we plan to implement

3.1.4 Implement Weekly scheduling

Implement from scratch an algorithm for weekly scheduling.

3.2 Improve the present scheduling algorithm

By taking into consideration

1. Priorities of task .
2. Breaks in between tedious tasks.
3. Neglecting small chunks of free time for tedious tasks.

3.3 User Survey

In this section, we have described analysis of results for different questions that we asked while taking the survey to understand the utility of implementing the better version of existing time scheduling WolfPlanner and whether they need improvements such a product or not. The questions which we asked are as follows:

1. Do you face difficulties in completing tasks because of not being able to manage time properly? Figure 5: Response for Question 1

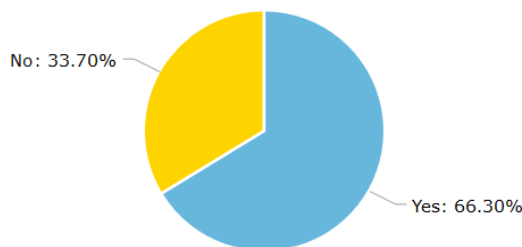


Figure 5: Difficulty due to time management

2. Do you face an issue to set up project (or other) meetings due to conflicting schedules?

The purpose of this question so that we could evaluate

the need of collaborative scheduling methods. The results suggest that people do face the issue of setting up meetings as their schedules conflict. Hence, indicating a need of a collaborative personal scheduler.

Figure 6: Response for Question 2

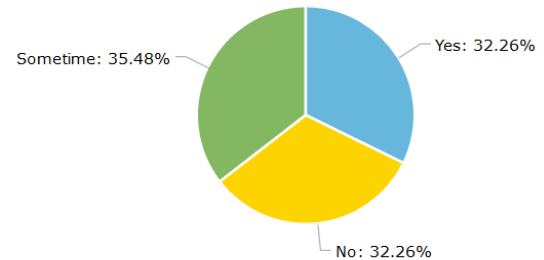


Figure 6: Issue in setting up project meeting due to conflicting schedules

3. How much do you rely on Google Calendar (or similar) for planning your day-to-day tasks?

This question was to understand the percentage of people depending on Calendar applications for daily use. This result tell us that people do not rely on calendar applications for planning their activities. But the following question (Q4) tells us that they do want a software which makes schedule for them.

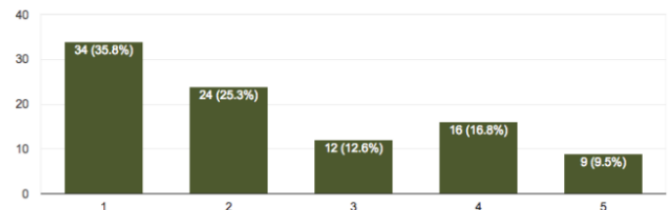


Figure 7: How much do you rely on google calender

4. Do you feel the need for a software which makes a weekly schedule for you taking into account your pending tasks and preferences?

The results of this question showed us that more than three-fourth of the population need an application that helps the user. The purpose of this question was to understand if the people face difficulties in time management. From the survey, the results tells us that there are not many people who do not experience the problem of task completion due to ineffective time management, so our bot might help to solve this problem.

Figure 8: Response for Question 4

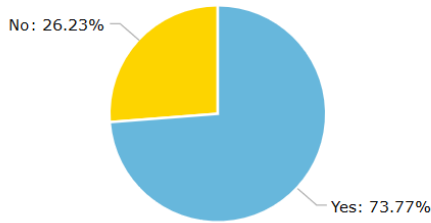


Figure 8: Need of software for weekly schedule

5. Do you think WolfPlanner will be more user friendly if it accommodates features like including regular breaks, project meetings and prioritization of tasks? This question was asked to understand the satisfaction level of people with the existing time scheduling and their interest in a more flexible scheduling algorithm. From the survey. The results showed that people will be more comfortable if we accommodate features like that since the bot becomes more practical to use. We also got few feedbacks saying it will be more real to accommodate breaks.

Figure 9: Response for Question 5.

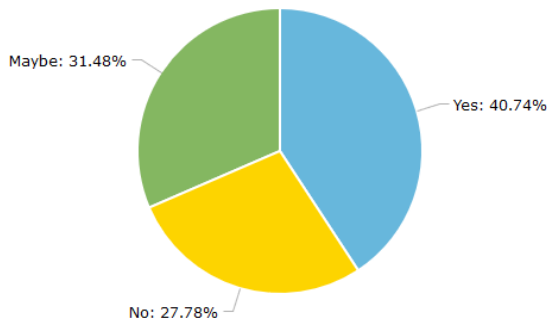


Figure 9: Do you think WolfPlanner will be more user friendly if it accommodates features like including regular breaks, project meetings and prioritization of tasks

Survey Conclusion Based on the 92 responses we got from our survey we can conclude that a personalized time management bot will be very helpful for the people. Most of the respondents do face a problem of completing tasks efficiently due to lack of proper time management. Using a bot which will schedule tasks for the individual will help with proper management of tasks and also save the user's time as there is no need to schedule tasks manually. The bot will also take care of meeting scheduling which is hassle for some of the respondents. The survey tells us that not many people rely on calendar applications like Google Calendar for task scheduling but would like to use a software like our bot. From this result we can conclude that people will start using softwares similar to calendar applications for task scheduling if there

are more important features like dynamic schedule generation.

4. IMPLEMENTATION

4.1 Software engineering methodologies

4.1.1 Agile Methodology

We plan to use Agile methodology for development of our application. So the features were built initially as small incremental features, and then the final one was built on top of those.

4.1.2 Code Review

Code Review practices were followed by creating different branches in github other than master and working on those before pushing on master. In this way, we managed to reduce bugs and error in our code and code was reviewed by the members before committing to the master. Going forward, we will do the same.

4.2 Working Of the Bot

1. Authenticate the slack user using the OAuth handler of Botkit and Slack
 - (a) When a user starts communication with the bot, the user is first authenticated. Since, we have used the BotKit toolkit, it provides a handler which helps us authenticate the user. 1.2 Slack sends a OAuth request oauth which is caught at our server and the user is verified.
2. For every new user create a new document in the MongoDB MLab collection.
 - (a) When the user enters on the Slack interface, Slack sends POST to the server at slack/receive. These are events (message.im and message.channel) which the bot has subscribed to. So whenever there is a message posted our server is notified. 2.2 The user is checked if an entry is already present in the MLab student collection. If the user is present then the user details for the further steps. 2.3 If the user document is not present, then create one. Initial entry into the user document is the user's slackID and the user's unityID.
3. Get the user input (either Tasks or Courses) and store them into the user's Mlab entry
 - (a) When the user requests to add tasks/courses, Slack sends a HTTP POST to our server at message. These are classified as Interactive Messages. 3.2 The bot sends a request to open up a dialog using the dialog.open Slack API. 3.3 The user fills up the necessary details and this is received by the bot controller. The bot connects to the MLab using mongoose, and the document is updated.
4. When 'fetch schedule' command is invoked, the bot sends the user identification details to the scheduling algorithm.

- (a) When the user requests for fetch schedule, the bot fetches the user's unityID and generates a day-date dictionary which is the current week's dates(according to which the schedule is to be generated) 4.2 Now, the bot uses the python-shell package to call the core-scheduling algorithm, passing the above variables as arguments to it. 4.3 The algorithm working is described below. The output of the algorithm is a generated schedule which is stored in the respective user document.

5. The bot displays the generated schedule on the Slack Interface.

- (a) Once the schedule is generated, the bot fetches the schedule from the MLab user document.
- (b) This fetched scheduled is formatted in a way which is easily understandable to the user and the bot displays this formatted schedule.

4.3 Scheduling Algorithm

The fundamental problems that MLFQ tries to resolve are two-fold. Firstly, It works to improve the 'turnaround' time (amount of time taken to finish the task). In our case that is really important because once the user starts working on a task he would like to finish that task within some sensible timeframe and not let it extend for a lifetime. Secondly, It wants the user to feel that the system is responsive (time taken by the system to schedule the job), in our context that means that the user wants to start working on tasks as soon as possible, when he is updating the list of tasks at hand. Moreover, everyone wants a break from the monotony of doing the same task and thus, this is where scheduling of multiple tasks as 'Round Robin' would be helpful. All this integrated with the priority system and the accommodations for various things, we believe that it could improve the lives of our peers. The algorithm is as follows :

Rule 1: If $\text{Priority}(A) > \text{Priority}(B)$, A is higher on the schedule.

Rule 2: If $\text{Priority}(A) = \text{Priority}(B)$, A & B are scheduled in 'Round Robin' of 2 hours with 15 minutes break in between.

Rule 3: When a job enters the system, it is placed in the appropriate queue (the topmost queue if it is of highest priority).

Rule 4: Once a job uses up its time allotment at a given level, its priority is reduced (i.e., it moves down one queue).

Rule 5: After some time period S, move all the jobs in the system to the topmost queue to avoid 'aging'.

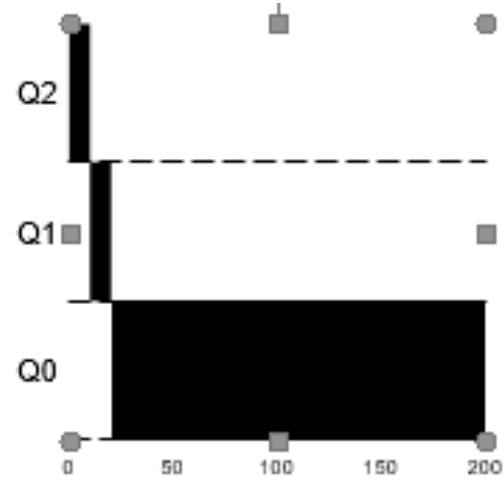


Figure 10: Long running job over time

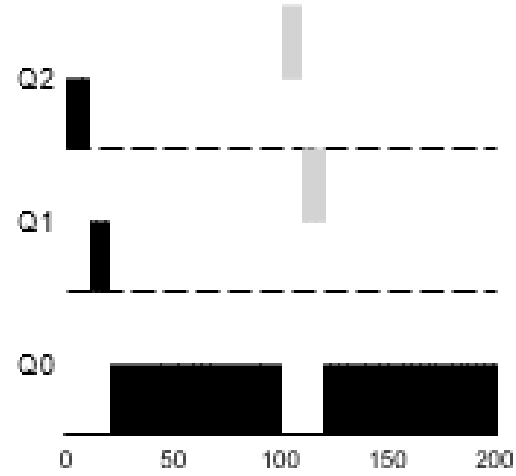


Figure 11: Along came an interactive job

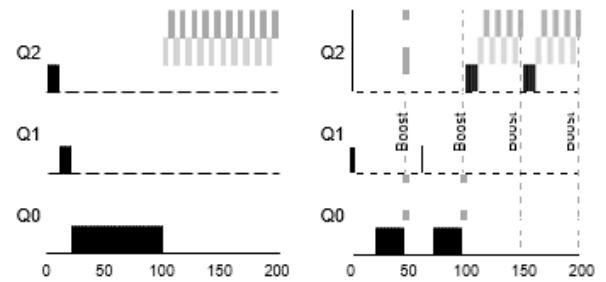


Figure 12: Without priority boost *left* and with priority boost *right*

4.3.1 Algorithm Explanation

1. If the user has a long running job, we will be decre-

menting its priority on the basis of the maximum allowed time limit in each queue level. See Figure 10.

2. Now, if the user wants to schedule another job of the highest priority, the older process is rescheduled and the new process is scheduled. See Figure 11.
3. Suppose a process is running for a long time, now two processes of higher priority come and hog the schedule (aging), see Figure 4 (left). To take care of that we do priority boosting, see Figure 12 (right).

Table 1: Evaluation of Different Methodology

Methodology	1	2	3	4	5	6
Traditional	-	+	+	++	-	--
Scheduled	++	++	-	+	+	+
On-Demand	-	--	++	--	--	++
Real-Time	-	++	++	+	++	--

Figure 13: Evaluation compared to other methods

5. ACKNOWLEDGEMENTS

We would like to extend special thanks to our Professor, Timothy Menzies and our mentor, Ken Tu for helping us to come up with this idea and giving us the opp

6. REFERENCES

1. Gartner.com, Gartner Customer 360 Summit 2011
2. Berry, Pauline, et al. Deploying a personalized time management agent. Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems.
3. Schwiegelshohn, Uwe, and Ramin Yahyapour. Analysis of first-come-first-serve parallel job scheduling. SODA. Vol. 98. 1998.
4. Wilkerson, L. J., and J. D. Irwin. An improved method for scheduling independent tasks. AIIE transactions 3.3 (1971): 239-245.
5. William Stallings Operating Systems: Internals and Design Principles 6th Prentice Hall Press Upper Saddle River, NJ, USA AEE 2008 ISBN:0136006329 9780136006329
6. MongoDB Documentation, <https://docs.mongodb.com/>
7. Slack Documentation Slack Reference Documentation : Retrieved from: <https://api.slack.com/getting-started-Department of Product Why Slack is popular? Retrieved from: https://api.slack.com/getting-started>
8. Node.js Documentation About Node.js Retrieved from: <https://nodejs.org/en/about/>
9. Botkit Documentation About Botkit Retrieved from: <https://botkit.ai/docs/>
10. Python Documentation Python Retrieved from: <https://www.python.org/about/> [Accessed: