Constrained Re-Planning in Spatial Crowdsourcing

DESIGN DOCUMENT

Team 51
Client/Advisor: Goce Trajcevski

Team Member	Roles
Logan Anderson	Test Engineer, Software Engineer
Nicholas Heger	Software Engineer, Progress Manager
Steven Sheets	Report Manager, Software Engineer, Test Engineer
James Volpe	Meeting Scribe/Facilitator, Software Engineer
Jared Weiland	Software Engineer, Progress Manager

Team Email: sdmay21-51@iastate.edu
Team Website: https://sdmay21-51.sd.ece.iastate.edu

Revised: 10/04/2020 – Version 1

Executive Summary

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

- Agile
- Black-box testing
- Object-oriented programming
- Subscriber-publisher model

Summary of Requirements

List all requirements as bullet points in brief.

- Create/research algorithm for task management
- Create a server to host algorithm
- Create a database to store data for users and workers
- Develop both a mobile and web-based application to allow utilization of optimized algorithm

Applicable Courses from Iowa State University Curriculum

- COM S 227: Object-oriented Programming
- COM S 228: Introduction to Data Structures
- COM S 309: Software Development Practices
- COM S 311: Introduction to the Design and Analysis of Algorithms
- COM S 363: Introduction to Database Management Systems
- CPR E 310: Theoretical Foundations of Computer Engineering
- S E 309: Software Development Practices
- S E 319: Construction of User Interfaces
- S E 329: Software Project Management
- S E 339: Software Architecture and Design

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- React
- Traffic API
- Task sorting/assignment algorithm
- MongoDB

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

We would like to acknowledge our faculty advisor, Goce Trajcevski, for his advice and guidance throughout this project. Dr. Trajcevski has helped further our understanding of the project's goals and has helped keep us on track and meeting deadlines. We would also like to thank our TA, Rachel Shannon, for being consistently available to answer questions.

1.2 Problem and Project Statement

Spatial crowdsourcing (SC) is an increasingly popular category of crowdsourcing in the era of mobile Internet and sharing economy, where tasks are spatiotemporal (belonging to both space and time or to space-time.) and must be completed at a specific location and time. It is a matching problem whereby one has: (1) a set of workers with their skills and geo-locations; (2) a set of job-sites with tasks requiring certain skills (and, sometimes there is a constraint on the sequence of tasks). Spatial crowdsourcing determines an assignment of workers to job-sites for a given task – taking into consideration the travel-time. However, oftentimes there are unexpected time-disturbances – e.g., traffic accidents, prolonged execution of previous tasks, etc., which render an existing assignment no longer optimal (in terms of completed tasks per day).

The purpose of this project is to develop algorithms and tools that will re-plan the assignments of workers to new/different job-sites when variables change unexpectedly. This is so that one can still optimize the overall number of completed tasks per day, while obeying certain constraints (e.g., minimizing the overtime pay of the re-assigned workers).

1.3 Operational Environment

The operational environment for this project will be web browsers and mobile devices. Since our end products are a web app and mobile app, there will be no physical constraints our project adheres to.

1.4 Requirements

- Functional Requirements
  - Allow task generators and workers to be able to create accounts (stored in DB)
  - Take worker inputs of skills, location, and reputation
  - Take task inputs of skills required and location
  - Optimize a schedule based on worker and task inputs
  - Re-optimize this schedule in the event of new information
  - Alert workers of tasks to complete
  - Web UI for the addition of tasks and visualization of work schedule
- Non-functional Requirements
  - Function with few bugs or issues that impede the users experience
  - Protect users' personal information from others
  - Optimized applications to run efficiently on mobile devices
  - Be able to be used by a large number of users at one time

1.5 Intended Users and Uses

The primary focus of this project is to create an efficient algorithm to solve spatial crowdsourcing problems where tasks need to be assigned to workers. As such, our end products (the web and mobile applications) will be very versatile and could be used by any spatial crowdsourcing service such as Uber or Grubhub. The intended users would then be any current or future users of any app that seeks to use spatial crowdsourcing to accomplish tasks.

1.6 Assumptions and Limitations

– Two separate lists, with a short justification as needed.

– Extremely important, as it can be one of the primary places where the client can go to determine if the end product will meet their needs.

– Examples of assumptions: The maximum number of simultaneous users/customers will be ten; Blue is the best background color and will be used; The end product will not be used outside the United States.

– Example of limitations: The end product shall be no larger than 5”x8”x3” (client requirement); The cost to produce the end product shall not exceed one hundred dollars (a market survey result); The system must operate at 120 or 220 volts and 50 or 60 Hertz (the most common household voltages worldwide).

– For limitations, include tests not performed, classes of users not included, budget/schedule limitations, geographical constraints, etc.

- Assumptions
    - Privacy is handled through outside sources. like location ghosting for hiding user location
    - There is only one task per an assignment
    - Tasks are assigned in sequence
- Limitations
    - Traffic APIs have a process cap on the number of routes that can be run per a period
    - Will need to be able to run on multiple types of mobile devices
    - Will need connection to the internet to receive updated information

1.7 Expected End Product and Deliverables

The main deliverables from this project is expected to be a mobile (as well as desktop) app that will take a set of tasks/workers assignment, along with the data used for such assignments. Upon notification that some values in the data used for the original assignments have changes (e.g., the average speed or travel-time along a road segment), the app will: (A) calculate the optimal re-assignment; (B) notify the affected workers (and job-sites) who are subject to such re-assignment. This will be finished and finalized by April 15.

2 Project Plan

2.1 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks.

For our project, the tasks can be decomposed quite simply. Users known as “task generators” will be stored in a database and generate a set of tasks, each task consisting of attributes such as geolocation, necessary skills, and a time requirement for the sequence of jobs. Users known as “workers” will also be stored in our database, each containing attributes such as geolocation, their skill-sets, their ranking amongst other workers, and pricing (per hour). Finally, our objectives are mainly focused on assigning workers to tasks, assuming single-task assignment and a sequential assignment of tasks.

The necessary tasks we must complete in order to complete this plan are as follows:

- Sep. 10: complete familiarization with the literature and existing approaches, decide running scenario/use-case.
- Oct. 10: finalize the selection of datasets to be used as sources.
- Oct. 25: finalize the selection of development platforms and provide architecture design with preliminary UI format.
- Nov. 10: finalize the selection of algorithmic solutions; devise use-cases and test-cases; develop test-plans (unit testing; integration testing; etc.); provide basic UI functionality.
- Nov. 20: finalize and submit the design document; prepare presentation.
- Jan. 25: finalize the role/component assignments and start implementing collaborative modules. - Feb. 15: complete unit testing; begin integration testing.
- Mar. 5: provide alpha-version for end-user testing; collect feedback.
- Mar. 20: finalize the revisions; release beta-version; run another set of end-user testing of functionalities. - Apr. 5: finalize the user-manual; prepare for public release.
- Apr. 15: deploy the final version at GitHub/GitLab; start the final report and presentation preparation.

2.2 Risks And Risk Management/Mitigation

- Task 1 & 2) methodology may not work with our project: 10%
    - We find this unlikely as at this point we should have enough information to make an educated decision about which to use.
- Task 3) development studio does not work as intended: 50%
    - If a studio does not work as intended and no major work has been done it would be in the interest of the project to switch to a different studio. if there is a fair amount of work done then it may be better to stick with it even if it does not work as effectively as it could.
- Task 4) Test cases do not cover all necessary paths: 70%
    - add more test cases to cover missing paths

    - Testing does not work with studio: 40%
- Task 5) N/A
- Task 6) Team member don't like doing assigned tasks: 40%
    - Team member falls behind on component: 70%
    - Would need to find out why they are falling behind and adjust the schedule as necessary.
- Task 7) Users don't like elements of the UI: 80%
    - Rework UI components
    - Users don't like functionality: 60%
    - Try and make changes, but will not completely rework
- Task 8) Identical to Task 7
- Task 9) Major issue is found before release: 10% or less
    - Try and hotfix the issues for release before making more lasting repairs. disable troublesome features if needed.
- Task 10) N/A

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

The proposed milestones of our project have, in essence, a 1-1 correspondence with the tasks described in Section 2.1. There are many metrics/evaluation criteria that can be used to evaluate our project. Some are as follows:

- **Usability**: Is our code easily understood? Does the UI provide simple usage? Is our documentation comprehensible?
- **Speed**: Is our software slow? Can it be faster? How could it be optimized? Load. Can our software/database deal with large numbers of users? If not, how could the database be improved.
- **Bugs**: Does our software have any bugs? How can they be squashed? Are they negatively impacting user experience?
- **Algorithmic Efficiency**: Does the algorithm make efficient schedules? How efficient should it be? Where do we draw the line between efficiency and practicality?

2.4 Project Timeline/Schedule

Figure 1: Gantt Chart

2.5 Project Tracking Procedures

Trello will be used to set tasks and track progress. major tasks are set with due dates. Tasks that need to be done are entered into a TODO column. when someone starts working on a task the page is moved to doing and link their name to it. when the task is done it is moved to the done and is archived. GitLab will be used for version control of the project. Documents related to the project are kept on Google Drive to keep a single version of the project documentation. General communication is being done through discord for communication history.

2.6 Personnel Effort Requirements

For each of these tasks, we set aside a number of days less than the time in-between tasks. This is an estimation of the number of days it would take to complete, if we spend merely half an hour each day. Keep in mind, this is with the combined effort of 5 workers.

Task	Estimated Completion Time (in hours)
Sep. 10: complete familiarization with the literature and existing approaches, decide running scenario/use-case.	7 days, 5 workers, 0.5 hours/day $7 \cdot 5 \cdot 0.5 = 15.75$ hours
Oct. 10: finalize the selection of datasets to be used as sources.	12 days, 5 workers, 0.5 hours/day $12 \cdot 5 \cdot 0.5 = 27$ hours
Oct. 25: finalize the selection of development platforms and provide architecture design with preliminary UI format.	8 days, 5 workers, 0.5 hours/day $8 \cdot 5 \cdot 0.5 = 18$ hours
Nov. 10: finalize the selection of algorithmic solutions; devise use-cases and test-cases; develop test-plans (unit testing; integration testing; etc.); provide basic UI functionality.	9 days, 5 workers, 0.5 hours/day $9 \cdot 5 \cdot 0.5 = 20.25$ hours
Nov. 20: finalize and submit the design document; prepare presentation.	6 days, 5 workers, 0.5 hours/day $6 \cdot 5 \cdot 0.5 = 15.75$ hours
Jan. 25: finalize the role/component assignments and start implementing collaborative modules.	14 days, 5 workers, 0.5 hours/day $14 \cdot 5 \cdot 0.5 = 31.5$ hours
Feb. 15: complete unit testing; begin integration testing.	16 days, 5 workers, 0.5 hours/day $16 \cdot 5 \cdot 0.5 = 36$ hours
Mar. 5: provide alpha-version for end-user testing; collect feedback.	13 days, 5 workers, 0.5 hours/day $13 \cdot 5 \cdot 0.5 = 29.25$ hours
Mar. 20: finalize the revisions; release beta-version; run another set of end-user testing of functionalities. - Apr. 5: finalize the user-manual; prepare for public release.	10 days, 5 workers, 0.5 hours/day $10 \cdot 5 \cdot 0.5 = 22.5$ hours
Apr. 15: deploy the final version at GitHub/GitLab; start the final report and presentation preparation.	8 days, 5 workers, 0.5 hours/day $8 \cdot 5 \cdot 0.5 = 18$ hours

2.7 Other Resource Requirements

Physical devices will be required for testing of web clients and mobile apps. Web testing may be done through any device with access to the internet, and mobile testing may be done through a mobile device or an emulator on a laptop or desktop computer. As most people have access to such devices it is not necessary to acquire devices specifically for testing. A server is also required. If the server is provided by the school then no additional resources will be required.

2.8 Financial Requirements

As the project progresses, it is possible that a cost for upkeep of the server may be needed. Since we anticipate a server is provided for us through this course, however, there are no expected expenses at the moment.