Mobility21

A USDOT NATIONAL UNIVERSITY TRANSPORTATION CENTER

Carnegie Mellon University

Penn
University of Pennsylvania

The Ohio State University

CCAC

#183 Dynamic Management of Food Redistribution for 412 Food Rescue

Zachary B. Rubinstein (PI)
https://orcid.org/0000-0002-6344-8692

April, 2020

FINAL RESEARCH REPORT

1. Project Overview

This report summarizes work performed and results obtained under the Mobility 21 University Transportation Center project titled: *Dynamic Management of Food Redistribution for 412 Food Rescue*. The goal of this project was to increase the capacity of the non-profit 412 Food Rescue (412FR) to use a crowd-sourcing model of volunteer drivers to redistribute viable food waste from vendors, such as grocery stores and restaurants, to pantries and other non-profit organizations in order to provide food to disadvantaged hungry people. Specifically, this project was to develop an intelligent scheduling system that would augment their existing technology infrastructure to enable more efficient use of their transportation resources. The primary accomplishments of this project are the following:

- FoodGo – a software system for scheduling vehicles to redistribute, i.e., pick-up and drop-off, food efficiently as opportunities dynamically arise.
- An Analysis of Achieving Resource Efficiency in a Crowd-Sourcing Context – This analysis discusses possible contention between efficient use of resources, i.e., vehicles, and privacy and control concerns prominent in crowd-sourcing models.

This report has the following structure. Section 2 describes the transportation model employed by 412FR and its inherent benefits and challenges. Section 3 presents FoodGo, including its capabilities, algorithmic design, requirements, and data representations. Finally, Section 4 presents an analysis of using FoodGo in the current crowdsourcing model used by 412FR and provides insights on moving forward.

2. 412FR Food Redistribution Model and Challenges

412 Food Rescue was founded in 2015 to address a major imbalance between, on one hand, the U.S., as a society, disposing of roughly 40% of perfectly good food while, on the other hand, one out eight people go hungry. To help turn food waste into part of a solution for the hungry, 412FR developed a model for redistributing food destined for landfills to the people in need. In this model, 412FR utilizes relationships with donor retail partners who have food overage, such as grocery stores and restaurants, and nonprofits engaged in emergency food relief, such as food pantries and free meal services for disadvantaged people. To transport food from donors to the non-profits, 412FR crowdsourced volunteer drivers and developed a phone app to facilitate the recruitment and the coordination of the drivers. To date, the crowdsourcing transportation model has worked well in terms of moving available food. Only 3% of available donations fail to be redistributed due to transportation failures. But, as donations continue to rise and 412FR's model is employed in other areas of the country, efficient use of the capacity of the volunteers becomes increasingly important. This project focused on improving that efficiency.

The current workflow at 412FR from receiving a notification about a new donation to assigning a volunteer to transport it from the source donor to the destination nonprofit goes as follows:

1. A dispatcher at the central office at 412FR is notified about a new food donation.
2. The dispatcher locates the donor and surveys for proximal nonprofits that might be able to distribute the donation.
3. The dispatcher contacts the nonprofits in a ranked order until one accepts the donation.
4. The dispatcher then enters a request into a central database that triggers a query sent to people with the phone app installed and who have permitted notifications. The request includes time windows for when the donation can be picked up and for when it can be dropped off. In addition, it provides information about the amount of the donation in a standardized unit and what the foodstuff is, which is mainly a consideration if the donation is perishable.
5. Initially, volunteers whose location is within a set distance of the donor location are notified about the transportation opportunity. The location of the volunteer is either his or her current location if the volunteer allows for location tracking or is simply the volunteer's home location. The first volunteer to respond is assigned the transportation.
6. If there are no volunteers within fifteen minutes of the initial request, then the request is sent to all volunteers and, as before, the first volunteer to respond is assigned the transportation.

In this workflow, the determination of whether or not a volunteer is capable and available to service a transportation request is done solely by the volunteer. The allocation strategy is, essentially, first-come-first-serve. Other than the approximate volunteer location filtering when deciding who to notify about the request, 412FR does not assess which volunteer might most efficiently service a request. In this project, we focused on adding this type of consideration when assigning a volunteer to service a request.

3. Technical Approach – FoodGo

The scheduling problem that 412FR has in assigning vehicles to transport food donations is known in the academic literature as a variant of the class of problems known as dynamic dial-a-ride problems (DARPs)[1]. DARPs are classified as being non-deterministic polynomial-time hard, which means that the best performing known algorithms for solving them optimally require exponential complexity. The basic structure of a DARP is that there is a set of requests for picking up orders at various locations and dropping them off at other locations, and the goal is to assign vehicles to satisfy those requests while observing both the constraints in the requests and the common physical world ones, e.g., vehicles travel at a certain rate. These problems are dynamic when the requests arrive over time rather than being known all in advance. A common variant of these problems includes time windows in which the orders have to be picked up and dropped off. Although the time windows tend to be very broad in 412FR requests, this variant is the one that applies to their delivery problem. In making the

assignments to the volunteers, the two types of constraints that must be observed in addition to the time windows are if the vehicle/driver is available to perform the pick-up and the delivery while satisfying the time-window constraints and if there is sufficient capacity during that availability period to accommodate the new order. Efficiency is typically measured in terms of an objective function that seeks to maximize or minimize some metric. A common metric used and one that we used in this project is travel duration, i.e., the objective function in this project is to minimize the total travel duration of all vehicles servicing food-transportation requests.

To generate and maintain schedules for servicing the transportation requests as they unfold over daily operations, we developed FoodGo, an intelligent dynamic scheduling system designed, in general, to provide fast and ongoing solutions to dynamic DARPs and, specifically, to provide guidance to 412FR in making assignments of volunteers to donation-travel requests. FoodGo is designed as a software service using the common httpd protocols of GET and POST commands to provide a messaging interface that supports conveying the necessary information for making new requests and getting assignment recommendations. This service is intended to be integrated within the current infrastructure that 412FR operates in support of their phone app and would run on their centralized service that currently supports the dispatchers and manages communication with apps running on the phones. We next describe the technological approach in FoodGo and then explain how it slightly changes the 412FR's workflow presented in Section 2.

As input, FoodGo incrementally receives requests for transporting food and, as output, it generates a ranked list of recommendations for assigning the request to a volunteer. That list is then used by the 412FR information technology infrastructure to solicit commitments from the volunteers. From the resulting commitments, the infrastructure chooses the commitment for the highest ranking option and both confirms the acceptance to that volunteer and informs FoodGo of that assignment.

The core algorithm for generating assignment options from a request uses a constraint-based search procedure [2,3,4]. The first step in this procedure is to translate the request into a pair of pick-up and drop-off tasks with appropriate temporal and capacity constraints, where a task is defined as two timepoints, a start and an end, connected by a duration constraint. FoodGo uses an incremental Simple Temporal Network (STN) [5] to represent timepoints and temporal constraints. The power of an STN is that it allows for easy representation of temporal constraints and feasible time ranges for the timepoints, i.e., a lower (earliest) bound and an upper (latest) bound. A fundamental capability of an STN is that it immediately propagate new bounds as constraints are posted or deleted. In addition, if a new constraint violates an existing constraint, e.g., a long duration might cause a task to exceed its deadline, the STN immediately reports the violation.

An STN is a graph comprising timepoints as vertices and temporal constraints as directed edges. There is a special vertex named Calendar Zero (CZ) that represents the start of time. Each timepoint has two sets of edges, one for lower-bound constraints and one for upper-

bound constraints. For both sets, a shortest-path, all-pairs algorithm is used to compute the time bounds on the timepoints. In the case of lower-bound constraints, the most constraining value is the greatest or latest value. Since the STN uses a shortest-path algorithm, the lower-bound edges are negated so that the latest value is the smallest. For the upper-bound constraints, the smallest or earliest value is already the most-constraining, so upper-bound edges do not need to be negated. For example, suppose there is a request that has a pick-up time between 10:00 AM and 10:15 AM and the duration of the pick-up is five minutes. Let CZ be set to 6:00 AM, then the pickup task would have the following structure. There would be a lower-bound edge from CZ to the start timepoint of the task with a weight of negative four hours. There would also be an upper-bound edge from CZ to the start timepoint of the task with a weight of four hours and fifteen minutes. Finally, there would be a lower-bound edge from the start timepoint of the task to the end timepoint of the task with a weight of negative five minutes and an upper-bound edge from the start timepoint to the end timepoint with a weight of five minutes.

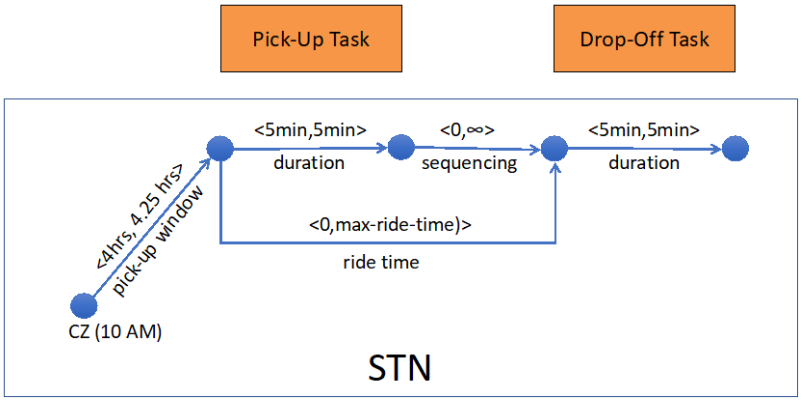

Figure 1- Task Instantiation for A Request

When a request arrives, both a pick-up and a drop-off task are instantiated into the STN with the constraints specified in the request (see Figure 1). An additional constraint permitted is based on whether or not the food is perishable. In this case, there may be a limited time that the food can withstand travel. This constraint is represented by a ride-time constraint between the start timepoint of the pick-up task and the start timepoint of the drop-off task and has an upper-bound of the maximum ride time. This task/graph structure represents all the temporal constraints for the request. In addition, there is a capacity constraint kept outside of that STN that is represented as a number of available space units in the vehicle required to service the request as well as the location constraints for the pick-up and the drop-off tasks. The next step is to use this structure to search over possible assignments and check for their feasibilities.

To understand how the task structure is used to search for assignments, it is necessary to understand first how assignments are maintained with FoodGo over time. Each vehicle's itinerary is kept on a timeline, which consists of a set of pick-up and drop-off tasks with sequence constraints, i.e., each having a zero-weight lower-bound edge and an infinite upper-bound edge, between the end timepoint of the preceding task and the start timepoint of the succeeding task. Sequence constraints ensure that the vehicle is being used for only one task

at a time. In addition, if travel is needed to get from one task to the next task, there is a travel constraint with a lower-bound of the travel duration and an infinite upper bound. Figure 2 shows an example of the routes and timelines for two vehicles.

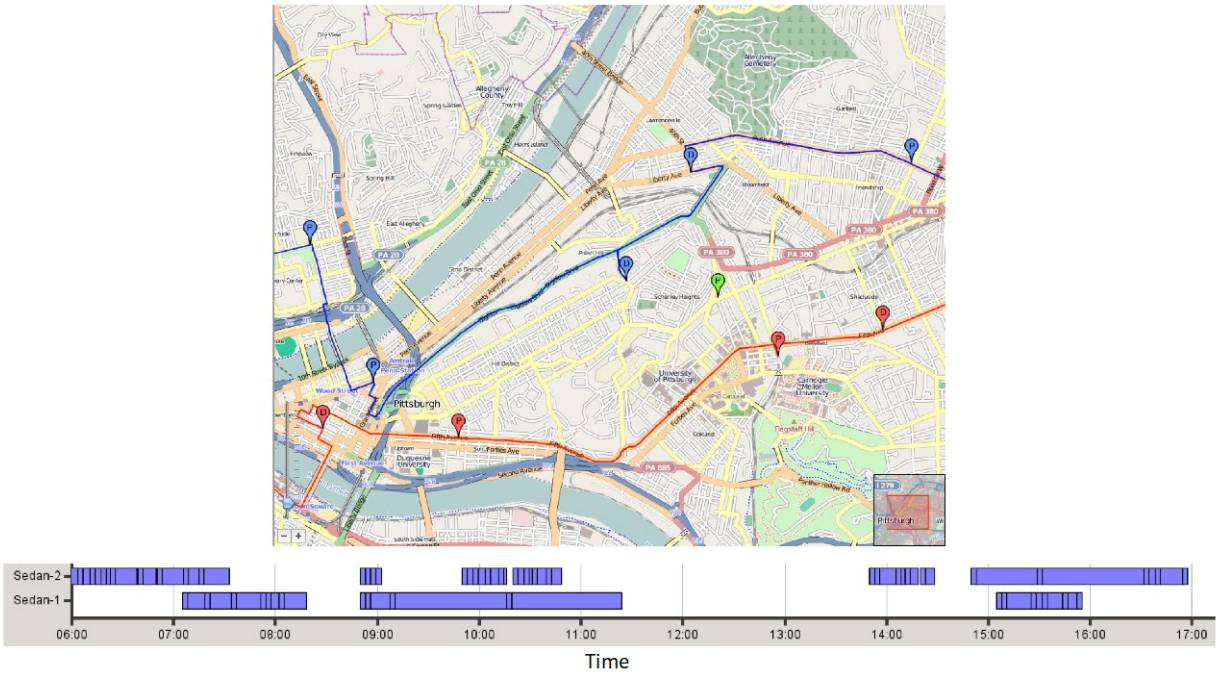

Figure 2-Example Routes and Corresponding Timelines

To test if a task can be inserted between two tasks on the timeline, the capacity constraint is checked first. If satisfied, then, the sequence and, if needed, the travel-time constraints are posted between the preceding task on the timeline and the new one. Finally, the sequence constraint and, if needed, the travel-time constraint is posted between the new task and the succeeding one on the timeline. If there are no violations in the STN while posting these constraints, then the insertion is viable. An assignment is viable if both the pick-up task and drop-off task instantiated for the request can be inserted on a timeline. To determine the goodness of an assignment, the duration added to the timeline in order to perform the task is computed and kept with the assignment option. To generate the set of possible assignments, FoodGo maps down all the timelines of known vehicles and collects the minimum-duration assignment option for each vehicle. The top *n* options, as determine by a configuration parameter, are returned to the 412FR infrastructure, which then queries volunteers using a process described later in this section.

In order to post travel duration constraints, it is necessary to be able to estimate how long it would take to go from one location to another. Currently, FoodGo uses a fairly rough calculation for this estimation. It calculates the great-circle distance, i.e, as the crow flies, between two lat/lon points and then uses a speed estimation for the vehicle to calculate a duration. Obviously, this model does not consider topographic obstacles, roads, and congestion, but it does work surprising well. In the future, we can explore using a service like Google or MapQuest to get durations (although there is an issue if this can be done quickly

enough in the context of a search that makes a large number of duration estimations) or develop our own path-generation mechanism using high-fidelity maps such as ones provided by OpenStreets.

When the options are returned to 412FR's infrastructure, it would then have to notify a set of volunteers of the opportunity. The concept is that the notification would be sent out to the volunteers in the top *n* options. Then, the infrastructure would wait for a designated amount of time (e.g., five minutes) for the volunteers to commit. If there are responses within that period of time, then the highest ranked committed assignment option is selected, and both FoodGo and the volunteer are apprised of the assignment. If there are no positive responses, then a larger set of volunteers are notified of the opportunity using the current volunteer-selection method. When FoodGo is notified of the new assignment, it adds the pick-up and drop-off tasks for the request in the appropriate positions of the affected timeline.

This constraint-based incremental procedure for scheduling new requests onto existing schedules has two main benefits. First, it is extremely efficient and produces very good, although not guaranteed to be optimal, schedules in real-time. The ability to slide tasks earlier and later within their constraints allows for finding good solutions without having to uproot the entire schedule. Second, it leaves other assignments in place, keeping the changes to the existing schedules of the volunteers to a minimum and reducing the cognitive load that would be needed to constantly track changes in their schedules.

To test FoodGo, we would have liked to have used historical data from 412FR, but the information about where drivers were at the time they began their trips and where they intended to go outside of servicing a single request is not part of the current 412FR model. Therefore, it is impossible to determine travel times in their data. Instead, we generated a synthetic data set of random locations within Allegheny County and partitioned them into a set of donors and a set of non-profits. In addition, we generated a set of volunteers that varied in capacity, availability, and initial locations. Finally, we generated a set of 500 transportation requests by randomly picking a donor, a non-profit, and the size of the food donation for each request. We used this data set to conduct two different experiments. First, we treated all the requests as coming in at the start of the day, i.e., as a static DARP. Second, we treated the requests as a dynamic DARP and simulated scheduling requests arriving throughout a day using FoodGo. We automated the scheduling decision to automatically select the highest ranking (minimum-duration) assignment option. In both experiments, we collected only the computation time because there is no historical information with which to compare the efficiency of the schedule.

Both tests were run on a MacBook Pro with a 3.3 GHz i7 processor and 16GB of RAM. The results were that, in the static DARP experiment, the entire schedule took fifteen seconds to generate and, in the dynamic DARP experiment, no assignment-option generation took more than two seconds. These results demonstrate that FoodGo can compute solutions quickly enough to keep up with real-time decision-making. We redid these two experiments with fewer volunteers in order to test in a more resource-starved environment. In both experiments, all requests were still scheduled with slightly longer run times of twenty seconds

for the static schedule and three seconds for the maximum time for an assignment-option generation in the dynamic context. Although we were unable to judge the efficiency of the schedule against historical data from 412FR, we can point to a result from our previous work on DARPs [6] that shows how the same scheduling technique used in FoodGo matched the results of the best performing known algorithms on a benchmark set of DARPs, in terms of scheduling all requests, and was able to generate those schedules at a fraction of the compute time required by the other algorithms.

4. Analysis of Efficiency and Crowdsourcing

Crowdsourcing is an innovative way to channel people's altruistic instincts to support causes in which they believe. It allows movements to build without having to have a large commitment of resources and finances. 412FR has used it to bootstrap and, so far, to grow their initiative to redistribute food to the hungry. But, there is a tension between having the flexibility and the fluidity of using volunteer resources and making efficient use of them. The 412FR model depends more on there always being volunteers to take on transportation requests than having them be used efficiently. In order to not discourage participation by volunteers, 412FR purposely limits its demands on volunteers to participate in their eco-system. As a result, the current 412FR model does not allow for sufficient information to be gathered from volunteers in order to make them efficient.

To use volunteers efficiently, there are minimal information requirements. Fundamental to these requirements is that there has to be some notion of where the volunteers will be over time. In order to minimize travel duration when servicing a new request, it is necessary to model from where the volunteers are coming and to where they are going. The information that 412FR has about volunteers' locations is, at best, very coarse. If a volunteer accedes to the app using the phone's location services, 412FR knows the volunteer's current location. Otherwise, it knows a home location for the volunteer, that is, whatever single address the volunteer is willing to provide. Often in 412FR's operations, requests originate in the morning but are only serviced in the afternoon or evening. The current or home information is unlikely to be very relevant when the request is actually going to be serviced and, therefore, hard to leverage when trying to make an efficient assignment.

To test whether or not their current crowdsourcing population would be willing to provide more relevant location information about their general expected whereabouts throughout the day and timing information about when the pick-ups and drop-offs would be performed, 412FR conducted a limited user survey consisting of thirteen participants. The conclusion was that most volunteers would not likely provide this information. Since without this information FoodGo cannot be effective, it was not integrated into 412FR's current system, but, going forward, both 412FR and we believe there could be a role for it.

As stated earlier, 412FR's current demand does not require efficient use of volunteers but that will change as 412FR expands. At that time, efficient use of their resources will become increasingly important. While a number of their volunteers may not be willing to provide location and timing information, the belief is that there will be a small subgroup of them who

are. These "power" volunteers will be willing to provide periods of availability and location information during those periods. FoodGo will then leverage this information to make efficient assignments. There will still be an issue of deciding when to use the "power" volunteers and when to use the general volunteers, and part of that process could be considering balancing requests over all volunteers both to ease the load on any one volunteer and to keep all volunteers engaged.

References

[1] Cordeau, J. F., and Laporte, G. 2007. The dial-a-ride problem: models and algorithms. *Annals OR* 153(1):29–46.

[2] Zweben, M.; Daun, B.; Davis, E.; and Deale, M. 1994. *Scheduling and Rescheduling with Iterative Repair*. Morgan Kaufmann Publishers. chapter 8.

[3] Cesta, A.; Oddi, A.; and Smith, S. F. 2000. Iterative flattening: A scalable method for solving multi-capacity scheduling problems. In *Proceedings of The Seventeenth National Conference on Artificial Intelligence*.

[4] Chien, S.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 2000. Using iterative repair to improve responsiveness of planning and scheduling. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, 300–307.

[5] Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal con- straint networks. *Artificial Intelligence* 49:61–95.

[6] Rubinstein, Z. B., Smith, S. F., and Barbulescu, L. 2012. Incremental Management of Oversubscribed Vehicle Schedules in Dynamic Dial-A-Ride Problems. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*.