Vehicle Routing Problem with a Helper

Ekesh Kumar

Bruce L. Golden

September 2, 2021

Contents

§1 Introduction

§1.1 Preliminaries

National postal services seek to minimize the makespan of their deliveries by designing optimal routes for their vehicles. The *vehicle routing problem* (VRP) is an optimization problem in operations research in which one seeks to find an optimal set of routes for a fleet of vehicles. Vehicles spend time traveling to predefined service stops and spend a predefined amount of time at each service stop to satisfy a customer's demands. Variants of the vehicle routing problem seek to minimize differing objective functions such as the sum of the route lengths or the makespan of the routes.

In this paper, we explore a variant of the vehicle routing problem — the *vehicle routing problem with a helper* (VRPWH). Instead of having a fleet of vehicles, this problem studies the case in which a single truck driver and a helping agent (a human or robot) work together to provide service at a set of service stops. More precisely, the VRPWH problem has a single *truck driver* who drives a *delivery truck*. The truck driver works with a *helping agent* that can either ride in the delivery truck with the truck driver or travel by foot by themselves. The driver of the delivery truck and the helping agent seek to visit and service a set of predefined service stops S along a long path. We impose the additional constraint that the helping agent walks at a constant factor α slower than the delivery truck. That is, if the delivery truck travels at a speed of v, then the helping agent walks at a speed of $\frac{v}{\alpha}$. Furthermore, we impose the additional constraint that the helping agent is only permitted to service a fixed subset $S' \subseteq S$ of service stops that is known ahead of time.

We explore a few well-performing heuristics and metaheuristics for the problem at hand. Although this paper focuses on the case in which there is exactly one helping agent, the heuristics discussed in this report can easily be generalized to two or more helping agents.

§1.2 Past Work

In 2008, [1] Rhodes et al. proposed an idea about having a helper dispatch tool for UPS's delivery system. The authors delineate how the decision tool could have a direct impact on savings while also providing a more efficient delivery solution. However, the authors do not mention any routing decisions that can be applied in practice.

In 2017, [2] Lu introduced the dependent driver helper dispatching problem (DHDP), a related problem that "had never been studied before." The DHDP problem is similar to the vehicle routing problem problem: It seeks to minimize the cost required to service a set of customers with the assistance of a helper. The driver and helper can share the delivery workload. Furthermore, the DHDP problem is similar to the VRPWH problem as both problems impose the constraint that the helping agent cannot move on its own. However, the DHDP problem differs from the VRPWH problem in that the helping agent is less efficient at providing service to service stops than the truck driver. This is not the case in the VRPWH problem in which the truck driver and the helping agent require the same amount of time to provide service at a fixed service stop. The DHDP problem also differs in that the demands at a service stop can be split among both the primary driver and the helper. This is not permitted in the VRPWH problem — the truck driver or the helping agent must spend an uninterrupted amount of time to finish providing service at a service stop. Finally, Lu [2] only presents a single heuristic for the DHDP problem in the general case. In our paper, we provide several heuristics specialized for the case in which the underlying tour resembles a circle.

§2 Problem Definition

§2.1 Assumptions and Constraints

Consider a mail route with a vehicle that makes numerous stops along a path. At each stop, the mail carrier exits the vehicle and delivers mail to a set of customers clustered along a closed walk on foot. After delivering the mail, the mail carrier returns to his vehicle and continues along the path. Once the truck driver and the helping agent have finished servicing all of the customers, they must return to their initial starting point.

In this problem, we seek to find the minimum time required to satisfy customers demands and return to the starting point with the assistance of a helping agent.

We can model the problem by representing the n service stops as points $p_1, p_2, \ldots, p_n$ on a circular path in the real plane. With this circular representation, an instance of the VRPWH problem is fully defined by the following parameters:

1. A radius parameter r, which is used to describe the circumference of the underlying circle.
2. A set of service stops $S = \{p_1, \ldots, p_n\}$, where p_i denotes the location of the i^{th} service stop.
3. A time function $t : S \mapsto \mathbb{R}$, which maps each service stop to the number of time units required to service the customer at that stop.
4. A parameter α used to describe the factor by which the helping agent's walking speed is slower than the truck driver's driving speed. In other words, if the truck drives at a speed v, then the helping agent's walking speed is v/α.
5. A subset $S' \subseteq S$ of service stops describing where the helping agent can be

utilized. If $S' = \emptyset$, then the helping agent cannot be used at all. Conversely, if $S' = S$, we can employ the helping agent at any service stop.

We impose the following additional constraints:

1. The locations of the service stops are known ahead of time.
2. The time required to service each customer is known ahead of time.
3. The truck in which the truck driver and the helping agent begin in starts at the point $p_0 = (-r, 0)$.
4. The process of serving a customer cannot be split. At every service stop $u \in S$, either the truck driver or the helping agent must spend exactly $t(u)$ uninterrupted time units servicing the customer to completion.
5. The truck that the truck driver drives can only move in one of two directions (clockwise or counterclockwise) but not both. We assume a one-way street. On the other hand, the helping agent is able to walk in either direction.

An example of an instance of the VRPWH problem with parameter $r = 1$ is depicted in Figure 1:

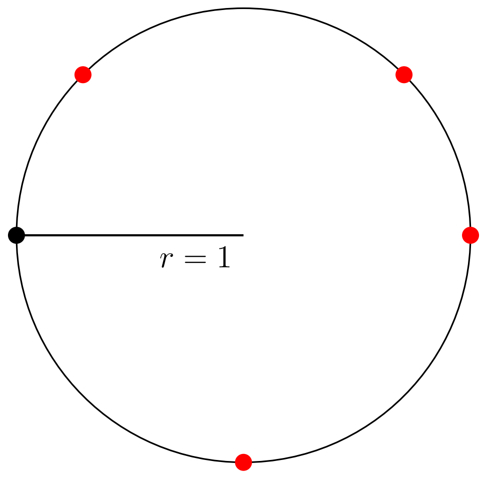

Figure 1: VRPWH Problem Instance

In this figure, the starting location of the truck is marked by a solid black dot. There are four service stops whose locations are marked with solid red dots.

For notational convenience, we define the following functions:

1. $CW : S \mapsto S$ maps a service stop to the next service stop in the clockwise direction.
2. $CCW : S \mapsto S$ maps a service stop to the next service stop in the counter-clockwise direction.

§2.2 Hardness

We now present a proof demonstrating that the VRPWH problem is at least NP-Hard:

Proposition 2.1

The VRPWH Problem is at least NP-hard.

Proof. Consider an instance of the PARTITION problem in which we are given a multiset of positive integers $S = \{a_1, \ldots, a_n\}$, and we are tasked with deciding whether or not S can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals that of S_2. It is well-known that PARTITION is an NP-Hard problem.

We will demonstrate that the VRPWH on a Circle is also NP-hard by reducing an arbitrary instance of PARTITION to an instance of the VRPWH on a Circle.

Consider an instance of PARTITION in which we are provided with a multiset of positive integers $S = \{a_1, \ldots, a_n\}$. We wish to determine whether S can be decomposed into the disjoint union of two multisubsets S_1 and S_2 such that $\sum_{a_i \in S_1} a_i = \sum_{a_j \in S_2} a_j$. We construct an instance of the VRPWH on a Circle whose optimal solution will yield such a partitioning.

Construct an instance of the VRPWH on a Circle in which the underlying circle has radius r, and we have n service stops with service times $a_1, \dots, a_n$. In the limit as $r \to 0$, the time required to go around the circle approaches zero. Furthermore, as $\alpha \to 1$, the helping agent's walking speed approaches that of the truck driver's driving speed. In other words, there is no advantage in having the helping agent dropped off at the service stops it visits over having the helping agent walking to that service stop on its own.

Thus, an optimal solution to such an instance of the VRPWH on a Circle becomes one in which the customers' demands are divided evenly. In other words, if T_1 denotes the time that the truck driver spends servicing customers and T_2 denotes the time that the helping agent spends servicing customers, then an optimal solution to the VRPWH on a Circle instance minimizes the quantity $|T_1 - T_2|$. By definition, the two multisubsets of S that attain the sums T_1 and T_2 form an optimal solution to the corresponding PARTITION instance. $\square$

§3 Properties of Efficient Solutions

When designing efficient heuristics for the VRPWH on a Circle, we take several observations into account. First, we present a few loose upper bounds for the problem at hand. Next, we introduce several strategies that are weakly dominated (i.e., there is always another course of action that is at least as good as that strategy).

By taking these bounds and dominated strategies into consideration, we can significantly reduce the search space for an efficient solution.

§3.1 Bounds

Proposition 3.1

Consider an instance of the VRPWH on a circle in which the underlying circle has radius r. Let the n service stops be $a_1, \ldots, a_n$ with respective service times $t(a_1), t(a_2), \ldots, t(a_n)$. No solution to the VRPWH will take less than $2\pi r + \frac{1}{2}\sum_i t(a_i)$ time.

Proof. If the truck driver were to service every customer by himself, he would spend a total of $\sum_i t(a_i)$ time. However, if this process were fully parallelized with the helping agent (i.e., for every time unit spent by the truck driver servicing a customer, the helping agent also spends a time unit servicing a customer), they would spend a total of $\frac{1}{2}\sum_i t(a_i)$ elapsed time units. This does not include any travel costs, which can only make this number increase. In particular, one can note that the travel costs will always be *at least* $2\pi r$ as the truck driver can only move in one direction meaning that it must go around the circle at least once. Thus, we conclude that no solution to an instance of the VRPWH on a circle will ever take less than $2\pi r + \frac{1}{2}\sum_i t(a_i)$ time. $\square$

With this result in mind, we seek to find efficient heuristics and results that will allow us to approximate an optimal solution in a computationally feasible manner.

§3.2 Dominated Strategies

Proposition 3.2 (Dominated Strategies: Helping Agent Idleness)

It is never advantageous for the helping agent to remain idle. For each solution in which the helping agent remains idle, there is a corresponding solution that can possibly upon the initial solution in which the helping agent always spends his time moving (possibly in the truck) or servicing a customer.

Proof. Consider an instance of the VRPWH on a Circle. Let IDLE denote a feasible solution in which the helping agent remains idle for a non-zero period of time. We will construct a new solution SOL that performs at least as well as IDLE in which the helping agent remains idle for exactly zero time units.

Consider an arbitrary point in time in IDLE in which the helping agent remains idle. Let p_k be the next point (either a service stop or the starting point) that the helping agent will visit. We consider two collectively exhaustive cases (the location of the truck driver does not matter):

1. The helping agent will walk to p_k on his own. In this case, it is clearly disadvantageous to remain idle. By assumption, the helping agent is not already servicing a customer. Moreover, p_k is the next service stop (or initial point) that the helping agent will visit. By removing the idle time and instead starting to walk to p_k immediately, we obtain SOL, which improves upon IDLE by exactly the time that the helping agent remained idle before starting to walk to p_k.

2. The helping agent will be dropped off at p_k by the truck. In this case, we can construct SOL by removing the idle time of the helping agent and instead have the helping agent walk towards the truck. This is always possible because the helping agent is permitted to walk in both the clockwise and counterclockwise direction. Furthermore, this solution can only improve upon IDLE since the helping agent cannot walk faster than the truck (i.e., $\alpha \geq 1$).

There is one last special case that needs to be considered. Consider the case in which there is no "next point" p_k that the helping agent needs to visit. This can happen when the helping agent has finished servicing the subset of customers that it was assigned by a feasible solution, and it has returned to the initial starting point on its own. In this case, we can still construct SOL by having the helping agent walk in the direction that is opposite of the direction that the truck is moving in. Since the truck cannot move in both directions, we are guaranteed that the helping agent will encounter the truck. Once the helping agent encounters the truck, it should ride in the truck with the truck driver for the remainder of the solution.

This course of action cannot make the preceding feasible solution any worse since the makespan of the solution is already being limited by the time that the truck driver requires to return back to the starting point. □

The next proposition illustrates that it is only optimal for the helping agent to re-enter the truck at a service stop.

Proposition 3.3

Suppose the helping agent is no longer in the truck with the truck driver. There is an optimal solution in which the helping agent only re-enters the truck at a service stop.

Proof. Suppose, for the sake of contradiction, that the proposition is not true. In other words, this means that there exists an instance of the VRPWH problem in which it is strictly advantageous for the helping agent to re-enter the truck driver's truck at a point that is not a service stop.

Let $a_1, \ldots, a_n$ denote the n service stops, labeled in a clockwise manner around the circle. Suppose the helping agent re-enters the truck between service stops a_i and a_{i+1}, where a_{i+1} is taken to be the starting point when $i = n$. Since the truck can only move in one direction along the circle (either clockwise or counterclockwise), there are exactly two cases to consider.

In the first case, we consider the scenario in which the truck is driving towards the helping agent. This means that the helping agent must have walked ahead and ended up in front of the truck at some previous point in time. An optimal solution would only have a helping agent walk ahead if it the helping agent were to service a service stop beyond the truck's current position. However, by Proposition 3.3, it is never strictly advantageous for the the helping agent to walk backwards after walking ahead. If the helping agent stays where it is after completing the demands at a service stop, the result follows. On the other hand, consider the case in which the helping agent walks forward after completing the demands at a service stop. This can never improve the final solution if the helping agent is being picked up by the truck because the helping agent will always arrive at its subsequent destination at the same time as the truck. Thus, having the helping agent wait at the last service stop it completes results in a corresponding solution that is at least as good as one in which the helping agent walks forward after servicing its last service stop.

Next, we consider the case in which the helping agent walks towards the truck. This can only be the case if the helping agent were left behind at some point. An optimal solution would only leave the helping agent behind if it the helping agent was being used to satisfy the demands at some service stop that the truck driver drove past by. By extension, the truck driver must have gone on to service some other customer. If this were not the case, the truck driver would have just serviced the customer that the helping agent was left behind to service. Consider the last customer that the truck driver services prior to the helping agent catching up to the truck. In an optimal solution, it only makes sense for the truck to wait at that last service stop since moving ahead can only increase the gap between the helping agent and the truck (the helping agent's walking speed is less than or equal to the driving speed of the truck). By assumption, we are considering the last service stop prior to the helping agent catching up, which means that we can only decrease the overall time of our solution by having the truck wait at the last service stop that the truck driver processed.

□

§3.3 Undominated Strategies

Proposition 3.4

Having the truck driver wait is *not* a dominated strategy (i.e., it may be optimal for the truck driver to remain idle).

Proof. It suffices to construct an instance of the VRPWH problem in which having the truck driver wait is undominated. Consider the following instance in which $\alpha = 1 - \epsilon$ for some suitable choice of $\epsilon > 0$ (i.e., the walking speed of the helping agent is just under that of the driving speed of the truck):

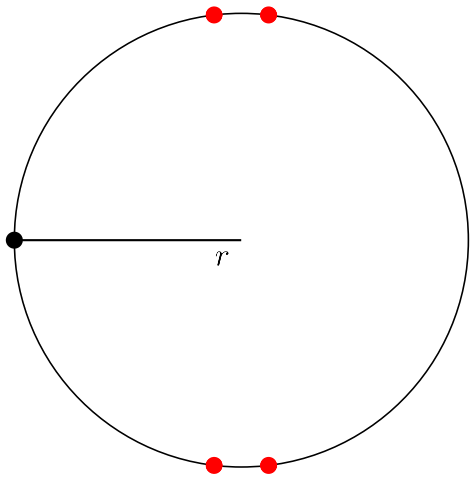

Figure 2: Primary Deliverer Waiting is Undominated

We can label the four service stops in clockwise order as p_1, p_2, p_3, and p_4. We assume that the helping agent can be utilized at each of the four service stops (that is, $S' = S$). Furthermore, we assume $t(p_i) \ll 2\pi r$, which means that the time required to service the customers is inconsequential when compared to the time required to travel around the circle.

With these parameters in mind, consider the following construction of a solution, denoted IDL, in which it makes sense for the truck driver to remain idle:

- The truck driver and the helping agent begin at the point $(-1, 0)$, denoted by the solid black circle. They begin to travel in the clockwise direction.
- Upon encountering the first service stop p_1, the helping agent is dropped off to service the customers at that station. The truck driver continues along its route in the clockwise direction until it reaches the second service stop, p_2. The truck driver services the customer at this second station.
- By construction, we can come up with an instance of the VRPWH problem in which the truck driver and the helping agent finish servicing p_1 and p_2 at exactly the same time.
- If $r \gg 0$, it would take a notable amount of more time for the helping agent to travel to the next service stop on his own than if it were dropped off by the truck. Furthermore, it would be too costly for the truck driver to make a full lap around the circle to pick up the helping agent. Since the distance between p_1 and p_2 is relatively small, the merits of having the truck driver wait for the helping agent to arrive at p_2 are clear.

$\square$

§4 Heuristics

§4.1 Baseline Heuristic

To begin our discussion, we introduce a baseline heuristic that does not utilize the helper vehicle at all. The purpose of the baseline heuristic is to simply provide a baseline so that we can easily compare the performance of other heuristics relative to this one. This heuristic is summarized below:

1. The truck begins at the point $(-r, 0)$ with the truck driver and the helping agent in the truck.

2. While at least one unvisited service stop exists, the truck travels clockwise around the circle to the next unserviced service stop. The truck driver satisfies the demand at the service stop while the helping agent remains idle.

The baseline heuristic represents the case in which we do not have the option to deploy the helping agent at all. It acts as a "control" in our subsequent comparisons.

§4.2 Look-ahead Heuristic

§4.2.1 Heuristic

The next heuristic we will discuss is called the **look-ahead** heuristic. The look-ahead heuristic decreases the total time spent by attempting to serve future service stops with the helping agent while the truck driver stays behind to serve a service stop with a large demand. The heuristic is summarized below:

1. The truck driver and helping agent both start at the point $(-r, 0)$ on the underlying circle.

2. While at least one unvisited service stop exists, the truck travels to the next unserved service stop a_i. At a_i, we greedily exercise our option to have the truck driver service the customer at a_i. Meanwhile, the helping agent "looks ahead," naively services as many service stops as possible, and returns to a_i in such a way that the truck driver's idle time does not increase at all.

§4.2.2 Demonstration

Consider the following instance of the VRPWH on a Circle with $r = 1$ and $\alpha = 1$ in which the locations of the service stops are marked with solid red dots. The starting location of the truck is the solid black dot. For the sake of simplicity, we assume the helping agent can be dispatched at any one of the four service stops.

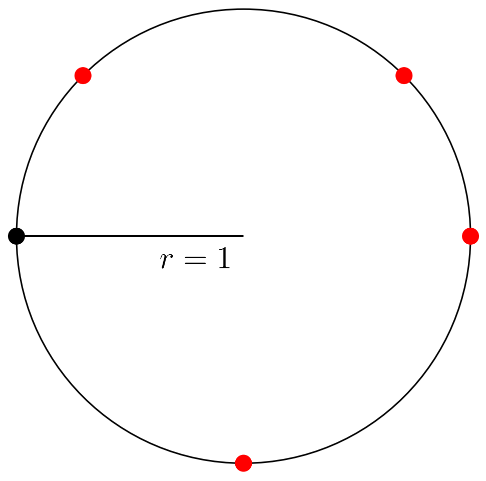

Figure 3: Look-ahead Demonstration

There are four service stops depicted in the instance above: $a_1 = (-\sqrt{2}/2, \sqrt{2}/2), a_2 = (\sqrt{2}/2, \sqrt{2}/2), a_3 = (1, 0), a_4 = (0, -1)$. Suppose the service times are given by $t(a_1) = 5$, $t(a_2) = 1$, $t(a_3) = 2$, and $t(a_4) = 20$. Furthermore, suppose that the time required to move a single unit on the Euclidean plane is equal to one time unit. Thus, we spend a total of 2π time units traveling around the entire circle once.

In this instance of the VRPWH on a Circle, the look-ahead heuristic would proceed as follows:

1. The truck begins at the point $(-1, 0)$ with both the truck driver and the helping agent.

2. The truck travels to the point a_1 at a cost of $\pi/4$ time units. We greedily exercise our option to dispatch the truck driver to provide service at this service stop.

3. While the truck driver spends $t(a_1) = 5$ uninterrupted time units servicing a_1, the helping agent walks $\pi/2$ units to a_2, services it in one time unit, travels $\pi/2$ units back to a_1, and waits for the helping agent to finish servicing a_1. That is, the helping agent "looks ahead" to see if it can service future service stops while the helping agent stays behind servicing a different service stop.

Note that the helping agent cannot service both a_2 and a_3 without increasing the idle time of the truck driver, so it must return to a_1 immediately after servicing a_2.

4. The truck driver and helping agent travel together to a_3 in a total of $5\pi/4$ time units. The truck driver services a_3. Since the helping agent cannot service a_4 and return to a_3 without increasing the idle time of the truck driver, the helping agent remains idle for $t(a_3) = 2$ time units.

5. The helping agent and the truck driver travel together in the truck to a_4 in a total of $\pi/2$ time units. The truck driver services a_4 in $t(a_4) = 20$ time units while the helping agent remains idle.

6. Finally, the truck driver and helping agent travel together to their starting point in $\pi/2$ units. All customers' demands have been satisfied, so the heuristic terminates here.

The total cost of this solution is

$$\frac{\pi}{4} + 5 + \frac{3\pi}{4} + 2 + \frac{\pi}{2} + 20 + \frac{\pi}{2} = 27 + 2\pi.$$

Despite the straightforward manner of this heuristic, computational results demonstrate that it performs well under certain circumstances. One can note that the total time spent traveling will never exceed $2\pi r$ provided that the time needed to move one unit of distance is equal to one unit of time.

The lookahead heuristic improves upon the baseline heuristic by exercising the option to service customers ahead of the one that the helping agent is servicing in such a way that the helping agents' idle time does not increase. Computational results demonstrate that the lookahead heuristic actually manages to exercise this "lookahead" more than one might expect. For example, the table below demonstrates some of these results for varying radii when the number of service stops is fixed at 100 and the distribution of demands follows a standard normal distribution. Note, however, that we do not permit negative service times. Thus, if the generated value is below 0 or above 2μ, it is discarded and generated again. This procedure preserves symmetry and ensures demands are non-negative.

The table below depicts the mean number of service stops skipped across 10^6 random instances of the VRPWH on a Circle with a varying radii. In this instance, we assume that the helping agent can service any one service stop with probability $p = 0.50$.

# of Service Stations	Radius	Mean # of Skipped Stations
100	1	17.620
100	2	16.897
100	5	14.516
100	10	11.324
100	25	6.508
100	50	3.772

Table 4: Average Number of Skipped Stations in Randomly Generated VRPWH Instances with $p = 0.50$

As demonstrated in the table above, even when the ratio of the number of service stops to the circumference of the circle in the VRPWH instance is small, we can still improve upon the baseline heuristic more than one may expect.

§4.3 Knapsack Look-ahead Heuristic

§4.3.1 Motivation

The next heuristic we present is the **knapsack look-ahead heuristic**, which we motivate through an example.

Recall that the look-ahead heuristic greedily exercises the option to utilize the truck driver at a service stop whenever possible. The helping agent subsequently serves as many service stops beyond the current one by considering the service stops by considering them in order of their distance to the current service stop. However, this approach may have drawbacks. Consider an instance of the VRPWH on a Circle, and suppose the diagram following represents a small segment of the circle (the curvature of the circle is not depicted here):

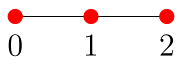

Table 5: Knapsack Look-ahead Example

For simplicity, assume that the helping agent is allowed to be dispatched at any of the three service stops. Suppose that the time required to satisfy each service stop's demands is given by $t(0) = 10, t(1) = 1$, and $t(2) = 7$.

The look-ahead heuristic would dispatch the truck driver vehicle at 0. While the truck driver serves this service stop, the helping agent would travel to 1, service it, and return. The truck driver and helping agent would then travel together to 2, and the truck driver would service it while the helping agent remains idle. The total cost of the look-ahead heuristic in this instance is $t(0)+2+t(2) = 19$ time units.

Depending on the value of the parameter α, a better solution could be to have the truck driver service the service stop at 0 while the helping agent travels and services the service stop located at 2. The helping agent could then return to pick up the truck driver at 0, they could both travel to 1, service it, and finally travel to 2 to finish. The total cost of this solution would be $t(0)+1+t(1)+1 = 13$ time units.

In summary, we see that the lookahead heuristic can be improved by not just servicing subsequent service stops in order by their distance to the current service stop but instead servicing subsequent service stops so that the time spent by the primary driver is approximately equal to that of the helping agent. Fundamentally, the problem can be reduced to finding a subset of numbers that sum up closest to a target number.

§4.3.2 Heuristic

The knapsack look-ahead heuristic operates as follows:

1. The truck driver and helping agent begin in the truck located at the point $(-r, 0)$.
2. The truck driver and helping agent travel together in the clockwise direction. If there is an unserved service stop a_i, the truck driver and helping agent travel to it.
3. We greedily exercise our option to use the truck driver to service the service stop. In the meantime, we "look ahead" with our primary vehicle, and we service a subset of service stops further in the clockwise direction such that the time spent (round-trip time plus service time) by the helping agent is as close as possible (but not more than) $t(a_i)$. That is, the helping agent's idle time does not increase at all.

§4.4 Neighbor-Expansion Heuristic

§4.4.1 Motivation

The next heuristic we present is the **neighbor-expansion heuristic**, which is a greedy algorithm that uses some elements from both the look-ahead and knapsack look-ahead heuristics.

The preceding two heuristics we discussed both sought to decrease the total time spent by servicing some set of customers by "looking ahead" with the primary vehicle while the helping agent stayed behind at a particular service stop.

Unfortunately, not every optimal solution takes this form. It can be disadvantageous for us to immediately dispatch the truck driver at service stops and only have the helping agent move forward. Consider the following line segment that could be a part of a larger VRPWH on a Circle instance:

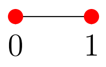

Table 6: Neighborhood-Expansion Example

Suppose we have the option to dispatch the helper vehicle at both 0 and 1 with $t(0) = 5$ and $t(1) = 15$. Having the truck driver drop the helping agent off at 1, returning to service stop 0 and servicing it, and returning to 1 would produce a solution that takes 16 time units. On the other hand, both the look-ahead heuristic and the knapsack look-ahead heuristic would take 21 time units. This demonstrates that it may not always be optimal to "look ahead" with the primary vehicle while the helper vehicle stays behind.

In order to combat the shortcomings of the look-ahead and knapsack heuristics, we propose a new heuristic that considers both "looking ahead" and "looking behind" as possible choices.

§4.4.2 Heuristic

Before introducing the neighbor-expansion heuristic, we introduce some terminology.

Definition 4.1. For each service stop $a_i \in S$, we define a set of **neighbors** N_{a_i}. The set N_{a_i} contains the set of service stops for which it is possible for the the helping agent to walk to, service, and return to a_i without exceeding a total time of $t(a_i)$.

Essentially, the neighbors of a service stop $a_i \in S'$ represent the set of service stops that the helping agent can potentially service while the truck driver services a_i without increasing the idle time of the truck driver.

The neighbor-expansion heuristic works as follows:

1. We begin with a set $A = \{a_1, a_2, \ldots, a_n\}$ containing every service stop.

2. While there are service stops in A, we perform the following:

   a) For every service stop $a_i \in A$, we compute the set of neighbors N_{a_i}.

   b) Next, for each set of neighbors N_{a_i} we compute a subset $N'_{a_i} \subseteq N_{a_i}$ of neighbors such that servicing all of the customers in N'_{a_i} (with travel costs included) is as large as possible but does not exceed $t(a_i)$. This is done in a fashion similar to the knapsack heuristic.

   c) Among all of the computed values, we take the a_i for which the difference between $t(a_i)$ and the time needed to service all of the customers in N'_{a_i} is minimal. Ties are broken arbitrarily.

   d) We remove both a_i and all of the service stops in N'_{a_i} from A, and we repeat this process until $A \cap S' = \emptyset$. The set N'_{a_i} represents the set of service stops that the helping agent will service while the truck driver services a_i.

Upon termination, the set A will be empty. Our solution is constructed as follows:

1. The truck driver and helping agent begin at the point $(-r, 0)$ and travel in the clockwise direction.

2. When the truck driver and helping agent encounter some service stop a_i that belongs to some N'_{a_k}, we should *not* process it immediately. Instead, when we visit a_k, the helping agent will be dispatched to service all of the service stops in N'_{a_k}. The service stop a_i will be serviced at that point.

3. When the truck driver and helping agent encounter some service stop a_i that does *not* belong to some N'_{a_k}, the service stop a_i should be serviced immediately.

§4.4.3 Demonstration

As a brief demonstration, consider an instance of the VRPWH on a Circle with $\alpha = 1$ in which the time required to travel one unit of distance equals one time unit:

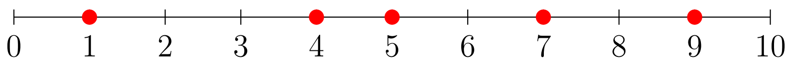

Table 7: Neighborhood-Expansion Demonstration. Curvature of the circle is not shown.

For simplicity, we assume the helper vehicle can be dispatched at any one of the four service stops. Furthermore, suppose we have $t(1) = 10, t(4) = 7, t(5) = 12, t(7) = 1, t(9) = 1$. The neighbor-expansion heuristic proceeds as follows:

1. First, we initialize $A = \{1, 4, 5, 7, 9\}$ containing the locations of our service stops.

2. Next, we compute the set of neighbors for each of the four service stops. We have $N_1 = \emptyset$ (the helping agent does not have enough time to travel, service, and return to this service station stop in fewer than $t(1) = 10$ time units), $N_4 = \{7\}, N_5 = \{4, 7, 9\}, N_7 = \emptyset$, and $N_9 = \emptyset$.

3. Subsequently, for each N_{a_i}, we compute a subset N'_{a_i} of service stops for which the time needed to service all of the customers in N'_{a_i} (with travel costs included) is as large as possible but does not exceed $t(a_i)$. One can find $N'_1 = \emptyset, N'_4 = \{7\}, N'_5 = \{4\}, N_7 = \emptyset$, and $N_9 = \emptyset$. Among all of the computed values, we select service stop 4 since $|t(4) - 3 - t(7) - 3| = 0$. From here, we remove 4 and 7 from A, and we iterate again.

4. In the second iteration, we follow the same procedure, and we end up removing 5 and 9 (the other neighbor sets are empty).

5. The final solution operates generated by this heuristic operates as follows:
   a) The truck driver and helping agent begin at point 0.
   b) The truck driver and helping agent move together to 1 at which the truck driver immediately spends $t(1) = 10$ uninterrupted minutes servicing the customer.
   c) The truck driver and helping agent move together to the service stop at 4. The truck driver is dispatched here while the helping agent travels to 7, services 7, and returns to 4.
   d) The truck driver and helper agent move together to the service stop at 5. The truck driver is dispatched here while the helping agent travels to 9, services 9, and returns to 5.
   e) The truck driver and helping agent navigate to 10. All of the demands have been satisfied.

§4.5 Randomized-Expansion Heuristic

§4.5.1 Motivation

The computational results in the subsequent section demonstrate that the proposed heuristics already appear to perform much better than the baseline solution. Notably, the knapsack look-ahead and the neighbor-expansion heuristics appear to perform better than the standard look-ahead heuristic.

However, this improvement comes at a price: both the knapsack look-ahead and neighbor-expansion heuristics are more involved, and they perform constant-time operations on the same service stop several times. Although the two heuristics appear to perform very quickly when the service stops and service times are generated according to naturally arising probability distributions, step 2 of either algorithm runs in worst-case exponential time. When working with several service stops, these approaches may be computationally infeasible.

This motivates us to seek different approaches. In this section, we present a randomized metaheuristic that is structurally similar to the neighbor-expansion heuristic. However, it speeds up and potentially improves upon the neighbor-expansion heuristic by utilizing stochastic decision making. We refer to this approximation as the **randomized-expansion heuristic**.

§4.5.2 Heuristic

In the knapsack look-ahead and neighbor-expansion heuristics, the process of finding the neighbors of some service stop $a_i \in S'$ is fast (i.e., they can be computed in polynomial time), but computing the desired optimal subset of S'. The randomized-expansion heuristic improves upon the latter step by making randomized decisions.

The randomized-expansion heuristic works as follows:

1. We begin with a set $A = \{a_1, a_2, \ldots, a_n\}$ containing every service stop.
2. While A is nonempty (i.e., while there are service stops that need to be serviced), we perform the following:
   a) For each service stop $a_i \in A \cap S'$, we compute the set of neighbors N_{a_i}.
   b) Next, for each set of neighbors N_{a_i}, we generate a random subset N'_{a_i} of N_{a_i}. If the time required to service all of the service stops (with travel costs included) exceeds $t(a_i)$, we discard the generated subset. This step is guaranteed to terminate because any singleton would satisfy the desired properties.[1].
   c) For each random subset N'_{a_i}, we compute the absolute difference between $t(a_i)$ and the time required to service (with travel costs) each of the service

[1]The expected number of steps needed to terminate is polynomial in n.

stops in N'_{a_i} and return to a_i.

d) Steps a, b, c are repeated a fixed number of times. Among all of the computed absolute differences, we take the a_i and N'_{a_i} for which the absolute difference was minimal.

e) We remove both a_i and all of the service stops in N'_{a_i}, and we repeat Step 2 until $A \cap S' = \emptyset$.

3. At this point, we construct a solution as we would in the neighbor-expansion heuristic.

4. We can repeat this entire algorithm several times, and we can take the best objective value found from all repetitions.

Introducing randomness into the expansion heuristic serves two purposes:

1. First, we are able to significantly improve the running time of the previous algorithms that we proposed.

2. Next, there is a high chance that some of the found solutions will differ from what the other proposed algorithms found. These differences are analogous to the concept of mutations in genetic algorithms.

§5 Computational Results

§6 Results

In order to benchmark the proposed algorithms, we generated random instances of the VRPWH on a Circle. For a fixed radius $r > 0$ and a fixed number of points N, the locations of the N service stops were generated randomly on the circle $x^2 + y^2 = r^2$. More precisely, this is done by generating a random angle θ by sampling a uniform random variable $Z \sim U(0, 1)$ and setting $\theta = 2\pi Z$. From here, we can easily find our random point (x, y) on the circumference of the circle by setting $x = r\cos(\theta)$ and $y = r\sin(\theta)$.

On the other hand, the service times of the N service stops were generated according to a "restricted" $N(\mu, \sigma^2)$ distribution in which service times above 2μ or below 0 were regenerated in order to preserve symmetry and avoid negative service times. In each instance, a parameter p was also specified, which was used to denote the probability of any one service stop belonging to the subset of service stops that the helper vehicle can be dispatched at. Each computed cost is calculated by averaging over 10^6 simulations each.

N	r	p	μ	σ^2	α	Baseline	Lookahead	Knapsack	NE	RE
10	1	0	1	1	1	16.277	16.277	16.277	14.825	14.955
10	1	0.25	1	1	1	32.574	32.184	32.365	29.666	29.926
10	1	0.50	1	1	1	48.889	47.790	48.277	44.534	44.927
10	1	0.75	1	1	1	65.180	63.101	63.985	59.393	59.916
10	1	1	1	1	1	81.468	78.192	79.518	74.241	74.896
25	1	0	2	1	4	137.739	134.462	135.789	122.319	123.069
25	1	0.25	2	1	4	194.030	188.837	191.005	170.438	171.292
25	1	0.50	2	1	4	250.420	241.678	245.323	218.603	219.560
25	1	0.75	2	1	4	306.652	292.994	298.562	266.637	267.676
25	1	1	2	1	4	362.934	343.226	350.978	314.746	315.899
50	1	0	3	1	2	519.123	499.414	507.166	425.126	424.158
50	1	0.25	3	1	2	675.370	639.488	653.169	535.478	532.364
50	1	0.50	3	1	2	831.623	770.076	791.248	645.841	640.581
50	1	0.75	3	1	2	987.823	895.194	922.998	756.159	748.653
50	1	1	3	1	2	1144.158	1016.848	1049.867	866.607	856.817

Table 8: Performance of Heuristics on Random Instances of the VRPWH Problem. N denotes the number of service stations; r is the radius of the underlying circle; p denotes the probability of a fixed service stop belonging to the subset of service stops that the helping agent can service; μ and σ^2 denote the mean and variance of the underlying normal distribution used to generate the service times of each stop; α denotes the constant factor by which the helping agent walks slower than the truck.

§6.1 Analysis of Results

From the computational results, we first note that all of the proposed heuristics outperform the baseline heuristic. This is expected since the baseline heuristic does not utilize the helping agent at all; it simply provides us with an initial value that we can use to compare to the other heuristics. Any heuristic that utilizes the helping agent can only improve upon this upper bound.

We next note that the knapsack heuristic is outperformed by the lookahead, neighborhood expansion, and randomized expansion heuristics. The fact that the lookahead heuristic seems to always outperform the knapsack heuristic is a seemingly surprising result as the knapsack heuristic will always consider the subsets of service stops that the helping agent will service in the lookahead heuristic as a feasible solution. Thus, the knapsack heuristic must be utilizing the helping agent in a manner that is locally optimal at the time at which the decision is made but does not lead to a globally optimal solution. We conjecture that having the helping agent go too far ahead of the the primary driver in order to provide service at a service stop early on in an instance may lead to inefficient solutions as it may become difficult to find service stops for the helping agent to provide service at later on in the instance.

We see that the simple lookahead heuristic always outperformed the baseline heuristic but it never outperformed any of the other heuristic. Despite its poor performance relative to the other heuristics, the lookahead heuristic is not computationally intensive compared to the other heuristics. On the other hand, the knapsack heuristic typically outperforms the lookahead heuristic but it is extremely computationally intensive. This stems from the fact that the knapsack heuristic is a pseudopolynomial dynamic programming algorithm, and its asymptotic runtime grows according to the sum of the service demands.

The neighbor expansion heuristic was also computationally intensive; however, it was not nearly as intensive as the knapsack heuristic. On most instances, it seemed to perform worse than both the lookahead and knapsack heuristics. Finally, the randomized expansion heuristic took a Monte Carlo approach, and it seems to perform the best out of all of the proposed heuristics. The running time of the randomized expansion heuristic typically fell between that of the neighbor expansion heuristic and the knapsack heuristic.

References

1. Rhodes, K., Nehring, R., Wilk, B., Patel, N. (2007). UPS Helper Dispatch Analysis. In Systems and Information Engineering Design Symposium, 2007. SIEDS 2007. IEEE. 1-6

2. Lu, Shih-Hao. ”Driver helper dispatching problems: Three essays.” Iowa State University (2017).