Improved modeling and solution methods for the multi-resource routing problem

Peter Francis, Guangming Zhang and Karen Smilowitz
Department of Industrial Engineering and Management Sciences
Northwestern University

April 21, 2006

Abstract

This paper presents modeling and solution method improvements for the Multi-Resource Routing Problem (MRRP) with flexible tasks. The MRRP with flexible tasks is used to model routing and scheduling problems for intermodal drayage operations in which two resources (tractors and trailers) perform tasks to transport loaded and empty equipment. Tasks may be either well-defined, in which both the origin and the destination of a movement are given, or flexible, in which the origin or the destination is chosen by the model. This paper proposes methods to effectively manage the number of options considered for flexible tasks (either feasible origins for a known destination or feasible destinations for a known origin). This modeling change generates sufficient options to allow for low-cost solutions while maintaining reasonable computational effort. We also propose a new solution method that uses randomized route generation. Computational results from test cases show that these changes improve the quality of solutions by at least 5% in the test cases as compared to methods from previous studies.

Keywords: Transportation; routing; heuristics; large-scale optimization; logistics

1 Introduction

This paper presents modeling and solution improvements for the Multi-Resource Routing Problem (MRRP) with flexible tasks. Smilowitz (2006) introduces the MRRP with flexible tasks as a method for solving routing and scheduling problems arising in intermodal drayage operations. Drayage involves the routing of two resources (tractors and trailers) to complete a set of tasks to transport

loaded and empty equipment. The set of tasks consists of both *well-defined tasks* and *flexible tasks*, as illustrated in Figure 1. For well-defined tasks, the origin, destination, time window, and resources required (tractor and trailer) are known. For flexible tasks, only one location (either origin or destination), time window, and resources required are known. The well-defined task in Figure 1(a) requires the movement of a tractor and a trailer from the equipment yard to the shipper. The flexible task in Figure 1(b) requires that the tractor and trailer be moved to the shipper, but no origin is specified.

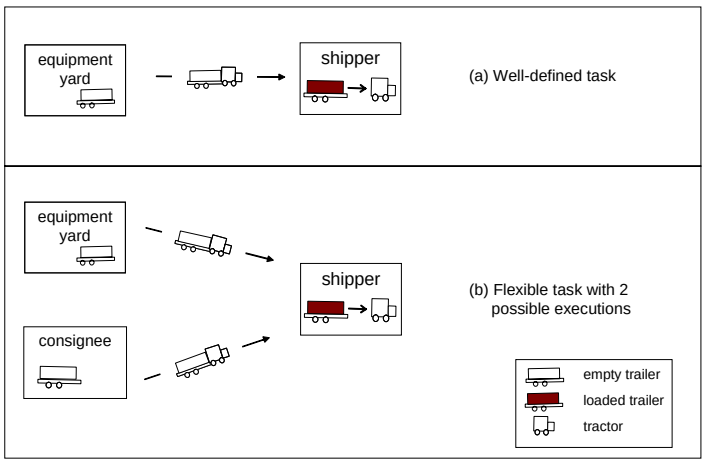


Figure 1: Illustration of drayage operations

The MRRP with flexible tasks is defined as:

Given: a set of tasks (well-defined and flexible) with required resources, processing times for resources and time windows; a fleet of each resource type; operating hours at all locations; and a network with travel times.

Find: a set of routes by resource type that satisfy all tasks while meeting a chosen objective function (minimizing fleet size, travel time) and observing operating rules for the tasks and resources.

For each flexible task, the MRRP finds an appropriate execution of that task. For a task with a flexible origin, an appropriate origin is found; for a task with a flexible destination, an appropriate destination is found. In the example in Figure 1(b), both the equipment yard and the consignee are possible executions of the flexible task of moving an empty trailer to the shipper. Potential executions must comply with time windows for the flexible task. In Smilowitz (2006), the number of feasible executions for a flexible task is limited by the distance between nodes; an execution is

considered geographically-feasible if the deadhead distance associated with the execution is within a radius ρ, which is constant across all nodes (origins/destinations for flexible tasks).

This paper introduces a method of defining node-specific radii for feasible executions rather than setting a single value for all nodes. The proposed Variable Radius (VR) method limits choices for nodes in dense areas (e.g. urban locations) and expands choices for nodes in isolated regions. As a result, we can limit the set of feasible executions for flexible tasks in a way that better reflects the geographic distribution of the nodes. Further, we present a randomized solution method, called the Greedy Randomized Procedure (GRP), to solve the resulting MRRP. The computational results demonstrate measurable improvement when applied to test cases from Smilowitz (2006).

The paper is organized as follows. Section 2 formally defines the MRRP and introduces issues related to defining the set of feasible executions and solving the MRRP. Section 3 introduces the VR method to choose feasible executions and the GRP method to generate vehicle routes. Section 4 describes the implementation of the combined GRP/VR method and presents numerical studies on large-scale instances of the MRRP. Section 5 presents a summary of the research.

2 Problem description

The MRRP is a special case of the redistribution problem, as defined in Dror *et al.* (2000, 2001). The redistribution problem designs vehicle routes to redistribute items from supply nodes to demand nodes at minimum cost or minimum required vehicle fleet, while observing vehicle capacity limits and driver work shifts. Unlike traditional pickup and delivery problems (see Savelsbergh and Sol (1995)), origins/destinations for flexible tasks are not given as inputs, but rather are left as decisions, further complicating the problem.

2.1 Formulation

Multi-resource routing problems with well-defined tasks can be modeled in two ways: (1) as arc-based network flow problems, or (2) as node-based vehicle routing problems (VRP). In arc-based formulations, the physical network is transformed into a time-space network. Time is discretized over the planning horizon. Each node represents both a physical location and an instant in time. The network arcs represent the movements of tractors and trailers between nodes. Alternatively, in node-based formulations, the origin and destination of a movement are aggregated into a single node that represents the entire movement with all the characteristics of the movement (duration,

origin, destination, time windows). The resulting problem is an asymmetric VRP in which tractors must visit each node, thereby completing all tasks to move trailers.

Applying these approaches to the MRRP with flexible tasks on a large scale is challenging. While the flexible origin/destination choice can easily be incorporated in the arc-based network, it is more difficult in the node-based network. Tasks which involve a choice of either origin or destination cannot be collapsed into a single node. The MRRP with flexible tasks is further complicated by the presence of time windows. Network flow formulations are well-suited for handling the time dependency between tasks. Such a formulation is studied in Morlok and Spasovic (1994) for drayage operations for a single rail carrier. However, the size of network flow formulations quickly becomes problematic. Ball *et al.* (1983) develop a network flow formulation for the distribution of trailers for a chemical company. They also transform the problem into a VRP which creates tractor tours to serve requested trailer movements and apply VRP solution methods.

Subsequent work on related problems with well-defined tasks has focused on node-based VRP approaches, rather than computationally intensive arc-based network flow formulations; see, for example, De Meulemeester *et al.* (1997) and Bodin *et al.* (2000). Smilowitz (2006) employs a node-based formulation for the MRRP with flexible tasks. Multiple executions for flexible tasks are generated. Each execution is represented as a node in the asymmetric VRP, and one of these nodes must be visited for each flexible task. Given a disjoint set of movements, a set partitioning formulation of the MRRP is used. Similar formulations have been effective at solving related routing problems; see Cullen *et al.* (1981), Dumas *et al.* (1991), Desrochers *et al.* (1992), Savelsbergh and Sol (1998), and Xu *et al.* (2003). These formulations partition items (here, tasks to be performed) into disjoint sets (which correspond to vehicle routes). Tractor routes must comply with the operating rules and tasks must be performed within time windows with the required tractors and trailers.

The following notation is used to formulate the MRRP:

$\mathcal{T}$: Set of tasks ($\mathcal{T} = \mathcal{T}_w \cup \mathcal{T}_f$) where $\mathcal{T}_w$ = well-defined tasks; $\mathcal{T}_f$ = flexible tasks

$\mathcal{E}_i$: Set of possible executions of flexible task $i \in \mathcal{T}_f$

$\mathcal{M}$: Set of movements [all tasks in $\mathcal{T}$ and all possible executions of flexible tasks]

$\mathcal{R}$: Set of feasible routes

c_r : Cost of route $r \in \mathcal{R}$

a_{ri} : Covering parameter: $= \begin{cases} 1 & \text{if movement } i \in \mathcal{M} \text{ is on route } r \in \mathcal{R} \\ 0 & \text{otherwise} \end{cases}$

$$x_r = \begin{cases} 1 & \text{if route } r \in \mathcal{R} \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

The set partitioning formulation of the MRRP from Smilowitz (2006) is:

$$\min \sum_{r \in \mathcal{R}} c_r x_r \tag{1a}$$

subject to

$$\sum_{r \in \mathcal{R}} a_{ri} x_r = 1 \qquad \forall i \in \mathcal{T}_w \tag{1b}$$

$$\sum_{r \in \mathcal{R}} \sum_{e \in \mathcal{E}_i} a_{re} x_r = 1 \qquad \forall i \in \mathcal{T}_f \tag{1c}$$

$$x_r \in \{0, 1\} \qquad \forall r \in \mathcal{R} \tag{1d}$$

The objective function (1a) minimizes the cost of routes where c_r is a weighted function of fixed vehicle cost and variable distance cost, such that minimizing fleet size is the primary objective. Equations (1b) are the partitioning constraints that ensure that all well-defined tasks are served by exactly one route. The partitioning constraints (1c) ensure that exactly one execution is added to the routes for each flexible task. These constraints are written as multiple-choice knapsack constraints (see Sinha and Zoltners (1979)). Finally, equations (1d) define the binary decision variables for each route.

2.2 Defining feasible executions

The ability to obtain good solutions for the MRRP with reasonable computational effort depends on the composition of sets $\mathcal{E}_i$ and $\mathcal{R}$. The size of the route set $\mathcal{R}$ increases with the possible choices for executions; i.e., the size of $\mathcal{E}_i$ for each flexible task $i \in \mathcal{T}_f$. As the number of choices increases, the solution quality may improve, but the problem becomes more difficult to solve. Smilowitz (2006) limits the set $\mathcal{E}_i$ by a fixed distance limit ρ for all nodes. Figure 2 shows an example with two flexible tasks, each associated with a shipper (nodes S_A and S_B) that requires a trailer. The trailers can be transported from nearby equipment yards or consignees, given that time windows are satisfied. When $\rho = 1$, as shown in the figure, shipper S_B in a sparse region has only a single option. However, increasing ρ to 2 to expand the options for shipper S_B results in many options for shipper S_A in a dense region, which may significantly increase the execution set $\mathcal{E}_i$ for $i = S_A$, and, in turn, increase the route set $\mathcal{R}$.

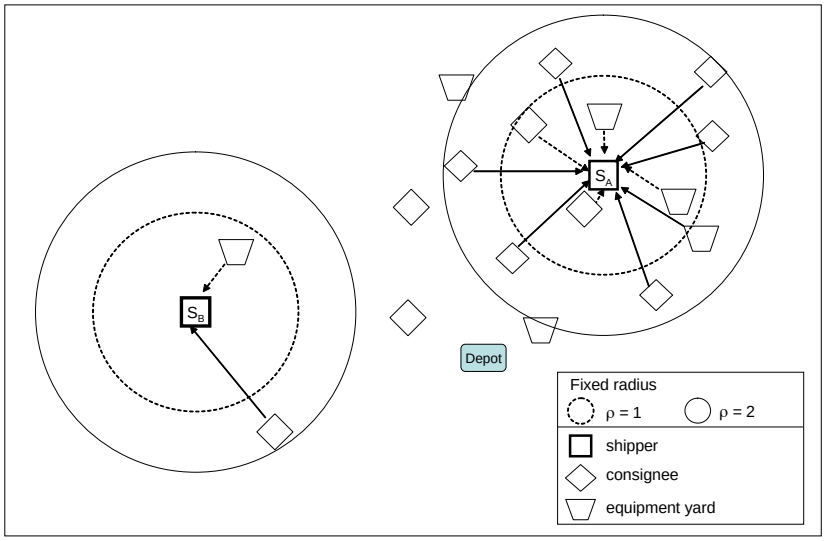


Figure 2: Illustration of the fixed radius region for feasible executions for two shipper nodes

Many practical instances of the MRRP involve nodes that are not distributed uniformly in geographic space. When using a fixed radius for every node, a node in a dense region may have a significantly larger set $\mathcal{E}_i$, compared with a node in a sparse region. The VR method, proposed in Section 3.1, introduces node-specific radii to balance the size of execution sets among nodes.

2.3 Solution methods

Even with limitations on the number of feasible executions, the number of feasible routes in a typical MRRP instance is prohibitively large and complete enumeration of the routes is not practical. Therefore, rather than enumerating all feasible routes for the set $\mathcal{R}$, a column generation approach is used with the linear relaxation of (1) to iteratively add routes. At each iteration, new routes are generated with a pricing problem using modified route costs defined by the dual variables for constraints (1b). The linear relaxation of (1) is solved again and the dual variables are updated. This process is repeated until a preset stopping criterion is reached (no new routes, maximum iterations, acceptable solution gap, or limited solution improvement). Next, an integer solution is obtained using a branch-and-bound technique. One can either continue to generate routes at all nodes of the branch-and-bound tree or employ a heuristic that uses only the final subset of routes from the initial column generation at the root node. Computational tests suggest that, for the test cases in this study, the route set generated at the root node are sufficiently diverse to yield good integer solutions without generating additional routes throughout the branch-and-bound tree.

The pricing problems employed to generate new routes within column generation are elementary shortest path problems with time windows and driver work shift constraints. These shortest path problems have been shown to be NP-hard; see Dror (1994). Two possible approaches to solve these problems are label-correcting dynamic programming and trip insertion heuristics. While dynamic programming methods yield optimal solutions for small problem instances of the MRRP with flexible tasks, the method cannot be used for larger instances; see Smilowitz (2006). Therefore, we use a trip insertion heuristic, which is based on a method for the VRP with time windows and worker shift constraints from Campbell and Savelsbergh (2004).

Let $\mathcal{U}$ be the set of movements not yet assigned to a route, and let $\mathcal{R}$ be the set of routes constructed. An algorithmic representation of the method is shown below:

Step 0:

$\mathcal{U} = \mathcal{M}$ all movements unassigned

$\mathcal{R} = \emptyset$ empty set of routes

Step 1: $\forall j \in \mathcal{U}$:

(1) $\forall r \in \mathcal{R}$: find least-cost, feasible insertion of j into r

(2) $\forall k \in \mathcal{U}$: find least-cost, feasible merger of j and k

Step 2: select best (least-cost) option from Step 1

If selection comes from (1) in Step 1

(a) update r by inserting j: $\mathcal{U} = \mathcal{U} \setminus j$

(b) if $j \in \mathcal{E}_i$ for some $i \in \mathcal{T}_f$ then $\mathcal{U} = \mathcal{U} \setminus m \quad \forall m \in \mathcal{E}_i$

If selection comes from (2) in Step 1

(a) create $\hat{r}$: merger of j & k: $\mathcal{R} = \mathcal{R} \cup \hat{r}$ and $\mathcal{U} = \mathcal{U} \setminus j, k$

(b) if j or $k \in \mathcal{E}_i$ for some $i \in \mathcal{T}_f$ then $\mathcal{U} = \mathcal{U} \setminus m \quad \forall m \in \mathcal{E}_i$

Step 3: Repeat steps 1 and 2 while $\mathcal{U} \neq \emptyset$

The method terminates with a set of feasible routes $\mathcal{R}$. Note that in Step 2(b), if $j \in \mathcal{E}_i$ is selected as the best feasible insertion (or if j or $k \in \mathcal{E}_i$ is selected as the best merger in the set of movements), all other executions for the flexible task i are removed from consideration. In the example in Figure 2, once a yard or consignee is selected to send a trailer to Shipper S_A, no other movements of trailers to S_A can be considered; there may be other consignees that would like to reposition an empty trailer to S_A, but such executions would be removed from consideration.

The VR method, presented in Section 3.1, affects the construction of set $\mathcal{E}_i$, and the GRP method, presented in Section 3.2, affects the selection of the merge in Step 2. While this paper focuses on the insertion heuristic to solve the routing subproblem, limiting execution choices with the VR method is desirable for the label-correcting dynamic programming method as well. The motivation behind these modeling and solution method changes is explored in the next section.

3 Modeling and solution techniques

This section introduces modeling and solution technique improvements for the MRRP with flexible tasks. Section 3.1 presents the VR method for defining feasible executions for flexible tasks based on node density. Section 3.2 presents the GRP method of solving the resulting MRRP.

3.1 Variable Radius method

The Variable Radius (VR) method considers the spatial distribution of nodes when defining the set of executions for a flexible task. The objective is to build a neighborhood around each node to balance the number of possible executions among all flexible tasks. If a flexible task is to originate (or terminate) from a given node, we consider possible executions only within the neighborhood of the node. Let Q denote the neighborhood size, in terms of number of nodes.

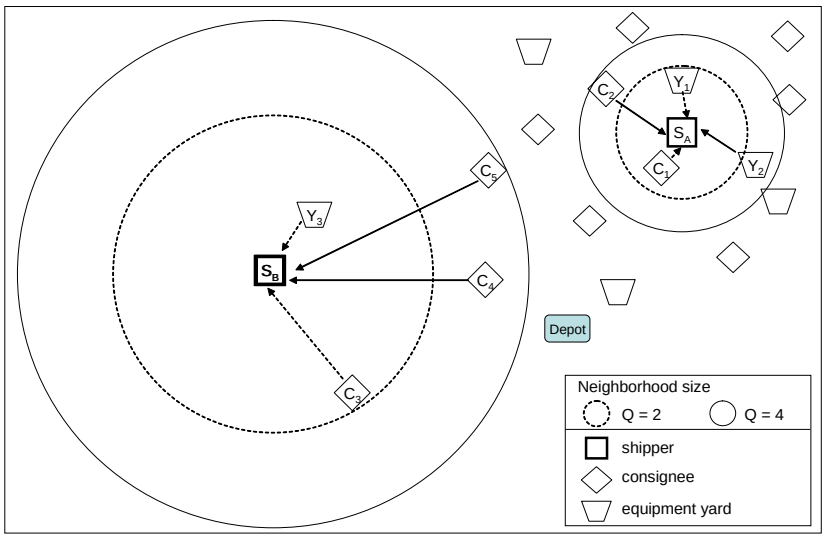


Figure 3: Illustration of Variable Radius algorithm for two shipper nodes

Figure 3 shows how feasible executions are created using the VR method for shippers S_A and S_B. With the VR method, the radius varies with the spatial density surrounding a node and the

desired neighborhood size. In Figure 3, two neighborhood sizes are shown, $Q = 2$ nodes and $Q = 4$ nodes.

The VR method is implemented as follows. Let the neighborhood N_i of node $i \in \mathcal{N}$ be the set of nodes within a distance ρ_i. The algorithm iteratively increases the value of ρ_i until the neighborhood reaches a minimum size, $|N_i| = Q$. For every node $i \in \mathcal{N}$, the following algorithm aggregates its nearest Q nodes into N_i:

Step 0:

$d_{ij} = \text{distance}(i, j), \forall i, j \in \mathcal{N}$

Step 1: $\forall i \in \mathcal{N}$:

(1) Let $\widehat{N}_i = \{j : j \in \mathcal{N}, \text{order ascending } d_{ij}\}$

(2) $N_i = \{j_l : j \in \widehat{N}_i, l = 1, ..., Q\}$

(3) $\rho_i = max\{d_{hi} : h \in N_i\}$

For each node $i \in \mathcal{N}$, the algorithm constructs an ordered list $\widehat{N}_i$ of nodes, sorted by increasing distance from i. The neighborhood N_i consists of the first Q nodes in the list $\widehat{N}_i$ and ρ_i is the distance between i and the Q^{th} node in the list $\widehat{N}_i$.

In the example in Figure 3, for $Q = 2$ the value of ρ_i for $i = S_A$ would be the distance between S_A and Y_1 and for $i = S_B$, the distance between S_B and C_3. Likewise, for $Q = 4$ the value of ρ_i for $i = S_A$ would be the distance between S_A and C_2 and for $i = S_B$, the distance between S_B and C_5. With this method, we can control the size of the execution set. Assuming time windows are not violated, for $Q = 2$, $\mathcal{E}_i$ for the task i associated with S_A would be $\{C_1, Y_1\}$ and $\mathcal{E}_i$ for the task i associated with S_B would be $\{C_3, Y_3\}$. Alternatively, with a fixed value of ρ for both shippers, the set of options would either be too small for S_B or too large for S_A.

3.2 Greedy Randomized Procedure

Intuitively, as the number of execution options increases, the solution should improve. However, a pure greedy insertion heuristic does not guarantee improvements in solution quality with increases in neighborhood size Q in the VR method. Figure 4 shows the relationship between neighborhood size Q and the two objectives, fleet size and travel time, for two typical problem instances from the data set described in Section 4.1. Note that c_r is defined in formulation (1) such that fleet size is minimized first, and then travel time. Instance 1 in Figure 4(a) exhibits the expected decrease in fleet size, but Instance 2 in Figure 4(b) does not show a monotonic decrease in fleet size. This is the

result of greedily choosing executions to serve the flexible tasks. Since a pure greedy method cannot guarantee a monotonic decrease in objective function value, we propose the use of randomization to increase the diversity of the routes generated.

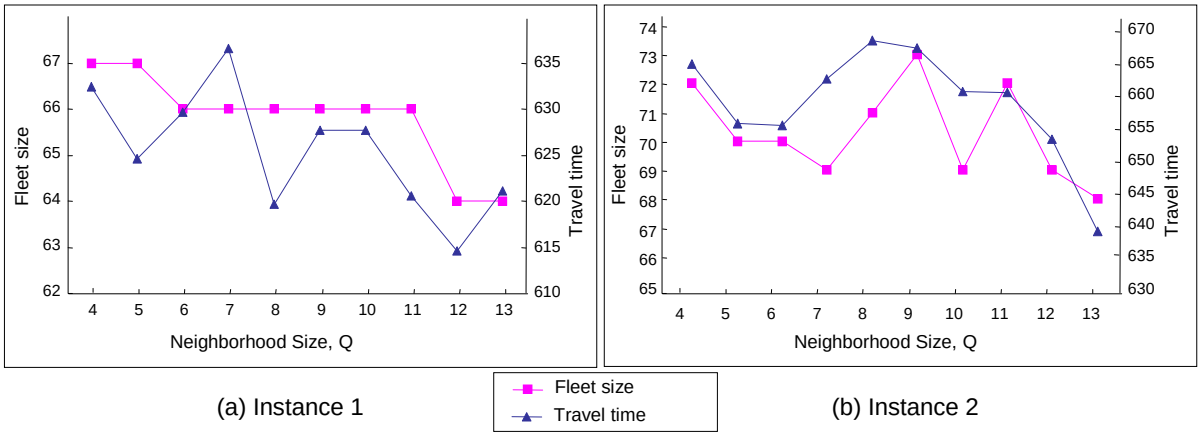


Figure 4: Inconsistent results with pure greedy method

The Greedy Randomized Procedure (GRP) solution technique is similar to Greedy Randomized Adaptive Search Procedure (GRASP) heuristics; see Feo and Resende (1989). GRASP uses a randomized greedy heuristic in a sequential adaptive procedure to first construct a feasible solution, followed by a local search procedure for improvement. A summary of the general GRASP procedure is presented in the appendix. The GRASP metaheuristic has been used to solve many combinatorial optimization problems, including machine scheduling by Feo *et al.* (1991) and set covering by Feo and Resende (1989). Carreto and Baker (1999) present a GRASP interactive approach to the VRP with backhauls. Kontoravdis and Bard (1995) use GRASP for the VRP with time windows providing two types of service, by calculating a greedy function of the insertion cost and the penalty cost. They observe that the key difference in philosophy between GRASP and other metaheuristics, such as Tabu Search and simulated annealing, is that GRASP focuses more on the initial construction of solutions than the subsequent local search procedure. For a review of metaheuristics for VRP's with time windows, see Bräysy and Gendreau (2000). Nanry and Barnes (2000) develop a reactive tabu search for similar pickup and delivery problems with fixed origins and destinations.

The GRP method introduces randomization in the route generation phase to produce a richer set of routes. Rather than accepting the best insertion in Step 2 of the insertion method as described in Section 2.3, an insertion must first pass a random test of acceptance. The insertion is accepted

with a probability: $P\,(\text{accepting best insertion}) = \alpha$. A random number $X \sim U(0,1)$ is generated and the best insertion is accepted if $X \leq \alpha$ and rejected if $X > \alpha$. Repeated many times, this random insertion method can be used to generate many different routes.

The master problem for the MRRP is solved with the routes generated using the GRP method. These routes correspond to the columns of formulation (1). It is essential to manage the number of columns in the master problem since computational results reveal an exponential growth in solution time with the number of columns. Allowing column generation to iterate until no routes with negative reduced costs exist may improve the solution to the linear relaxation of formulation (1), yet this is often not possible due to computational limits. The GRP method maintains a manageable number of columns by imposing a limit M on the number of column-generation iterations. To improve the solution quality, the entire solution method is repeated until either a maximum of K replications is reached or until the solution reaches the lower bounds from Smilowitz (2006), in which case the solution is optimal. Let z be the best known feasible solution, let z_k be the solution to (1) after M column-generation iterations at the k^{th} run of the solution method, and let z_{LB} be the lower bound obtained with the lower bound method in Smilowitz (2006) adapted for VR. The resulting solution approach consists of an iterative greedy heuristic with randomization, as follows.

Step 0:

Let $z = \infty$ and calculate z_{LB} with neighborhood size Q

Step 1:

(1) Solve $z_k = ColGen(Q, \alpha, M)$

(2) If $z_k < z$ then $z = z_k$

Step 2:

While $k \leq K$ and $z_k > z_{LB}$, repeat Step 1

In the initial step, a lower bound on the objective z_{LB} is obtained with the method from Smilowitz (2006), given a neighborhood size Q. The column generation method with GRP parameters Q, α, and M is repeated K times. At each replication, the best feasible solution is updated if $z_k < z$. If $z_k = z_{LB}$, we have found the optimal solution and the algorithm terminates.

The proposed improvements to the MRRP can be summarized as follows. The VR method more efficiently generates the set of executions for flexible tasks. In the column generation step, the GRP method introduces randomness in the assignment of movements to routes, rather than using a pure greedy assignment. The following section describes how the algorithm parameters

Q, α, M, and K are determined to implement this solution approach.

4 Computational study

Section 4.1 describes the drayage data sets used for the computational studies. Section 4.2 introduces the method of determining Q for the VR method, and Section 4.3 details the parameter setting techniques for the GRP method. Section 4.4 presents the numerical results.

4.1 Test cases for MRRP with flexible tasks

The test cases are based on data from drayage and third party logistics companies (Dahnke (2003); Corinescu (2003); Grosz (2003)) for dray movements over a region including greater Chicagoland and parts of central Illinois, southern Wisconsin and western Indiana. Since the customer data are proprietary, we create aggregated data sets from several drayage companies within close areas (i.e. the same zip code) that mask individual customer information.[1] The distance matrices for the aggregated data sets maintain the same geographical characteristics as the initial industry data. One set of aggregated test cases is used for parameter setting and a second set of aggregated test cases is used for evaluation of the GRP/VR approach. Further, we test the GRP/VR approach with a set of disaggregated industry test cases, which are not publicly available. Table 1(a) lists the test cases for parameter setting, and Tables 1(b) and 1(c) present the aggregated and disaggregated test cases for evaluation, respectively. Flexible tasks account for 50% of the total tasks on average.

Test case	Total tasks	Flexible tasks	Fixed tasks
1	25	16	9
2	25	16	9
3	25	9	16
4	25	12	13
5	25	7	18
6	50	24	26
7	50	25	25
8	50	26	24
9	50	31	19
10	50	19	31
11	75	41	34
12	75	32	43
13	75	44	31
14	75	35	40
15	75	42	33
16	100	62	38
17	100	44	56
18	100	52	48
19	100	49	51
20	100	42	58

(a) Aggregated data: parameter setting

Test case	Total tasks	Flexible tasks	Fixed tasks
1	25	13	12
2	25	10	15
3	25	10	15
4	25	10	15
5	25	19	6
6	50	27	23
7	50	23	27
8	50	31	19
9	50	24	26
10	50	25	25
11	75	43	32
12	75	43	32
13	75	42	33
14	75	37	38
15	75	34	41
16	100	60	40
17	100	51	49
18	100	55	45
19	100	40	60
20	100	55	45

(b) Aggregated data: evaluation

Test case	Total tasks	Flexible tasks	Fixed tasks
1	100	49	51
2	100	48	52
3	100	49	51
4	100	46	54
5	125	60	65
6	125	62	63
7	125	63	62
8	125	69	56
9	150	85	65
10	150	76	74
11	150	75	75
12	150	82	68
13	175	103	72
14	175	100	75
15	175	88	87
16	175	99	76
17	200	114	86
18	200	92	108
19	200	95	105
20	200	100	100

(c) Disaggregated data: evaluation

Table 1: Test cases of computational study of MRRP

[1] The data sets are available from the authors.

The operating parameters are detailed in Table 2. The model captures one day of operation, assuming the loads for the day are known when decisions are made. It is assumed that all tractor routes begin and end at one central depot, and that drivers work a continuous ten-hour work shift.

Parameter	Value
Time to pick up loaded trailer	30 minutes
Time to drop off loaded trailer	30 minutes
Time to pick up empty trailer	15 minutes
Time to drop off empty trailer	15 minutes
Time to load trailer	1 hour
Time to unload trailer	1 hour
Driver work shift	10 hours (continuous)

Table 2: Operating parameters

4.2 Variable Radius parameter setting

The VR method determines the members of the set $\mathcal{E}_i$ for each flexible task $i \in \mathcal{T}_f$ based on the density surrounding the known (fixed) location for the task. The neighborhood size is chosen to balance solution quality and solution speed. Let F_Q be a weighted combination of solution time and quality (measured by the primary objective, fleet size), which is used as a metric to guide the search for the neighborhood size Q.

We develop a nested partitioning method similar to the one proposed by Shi and Ólafsson (1997) to search among all values of Q. The method searches for the best value of F_Q by iteratively partitioning the feasible region of values for the parameter Q. The "best" value of Q is the one in which the optimal combination of minimum fleet size and minimum solution time occurs most frequently. In practice, the best solutions are typically obtained in regions characterized by better average performance; therefore, we focus on best average performance.

First, the process randomly samples the feasible region of Q. The current region is divided into m partitions according to a chosen scheme. We use a binary partitioning scheme to guide the search and limit Q to integer values between 1 and m, where m is some number less than the maximum number of nodes in the problem instance. While we could set m to the maximum number of nodes to maintain greatest flexibility; in practice, we observe that small values of m can be used since the objective tends to be insensitive for larger m. Since no changes in objective values are observed for $m > 20$ in our test cases, we set $m = 20$.

In each replication k of the search, ranking-and-selection procedures and multiple-comparison

procedures (see Matejcik and Nelson (1995)) are used to determine the amount of sampling needed from each region. If a certain subset is found to be the best, it becomes the most promising region in the next replication. Otherwise, if the surrounding region outweighs the current subset, the method backtracks to the region of the previous replication. See Ólafsson and Kim (2002) for more details of the search method.

Moves from the current partition to the most promising subset of that partition are made with a probability of 95%, which is the level of confidence that a sufficiently good value of Q has been found. Eventually, the subset of the potential values Q converges to a single value.

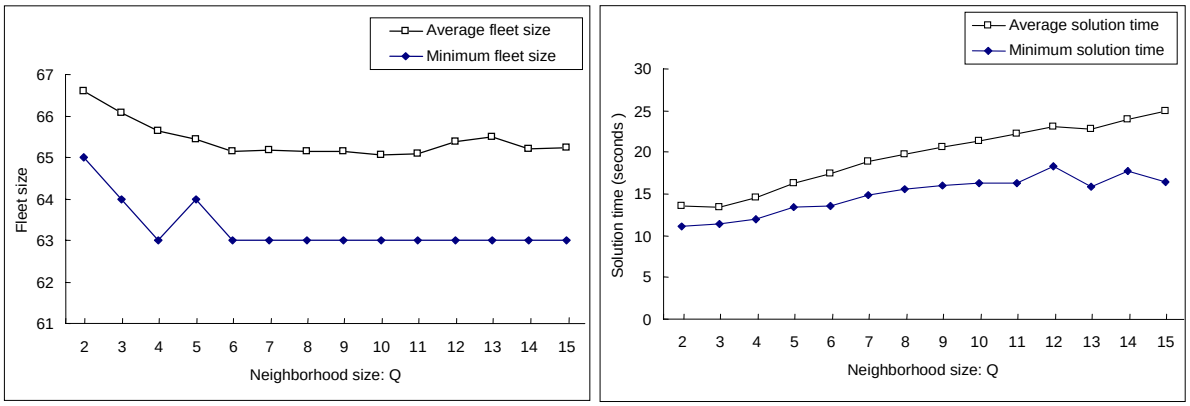
While binary partitioning is standard, we note that if m is small, then we prefer to partition the entire range into m partitions such that each partition corresponds to exactly one value of Q. This method saves time when m is small and no repeated samples are needed with successively smaller partitions over the same region.

	Q=2	Q=3	Q=4	Q=5	Q=6	Q=7	Q=8	Q=9	Q=10	Q=11	Q=12	Q=13	Q=14	Q=15
Expected objective	67.95	67.53	67.04	67.13	66.87	67.13	67.14	67.32	67.21	67.38	67.67	67.81	67.70	67.80
Multiple comparison LB	0	0	-0.13	-0.04	-0.55	-0.05	-0.03	0	0	0	0	0	0	0
Multiple comparison UB	1.38	0.96	0.47	0.56	0.05	0.55	0.57	0.75	0.64	0.81	1.10	1.23	1.13	1.23

Table 3: Nested partitioning results to determine the neighborhood size, Q

Table 3 shows an illustrative example to set the value of Q with the method described above, given an indifference zone of $\delta = 0.3$, i.e. differences of less than δ are not considered statistically significant, and each integer value of $Q = 2, 3, ..., 15$ is a subset. According to Table 3, values from $Q = 4$ to $Q = 8$ are suggested for the given test cases with 210 replications at each value of Q.

These results remain consistent when applied for longer replications. Figure 5 shows the solution quality and time for tests of 500 replications for each value of $Q = 2, 3, ..., 15$. In Figure 5(a) all values of Q in the regions from $Q = 6$ to $Q = 15$ find solutions with minimal fleet size of 63. In Figure 5(b) the average solution time for each replication increases with the neighborhood size Q. Since the objective considers both solution time and quality, these results suggest that $Q = 8$ and $Q = 9$ are the best regions. In these two cases, solutions with the minimum fleet size of 63 are obtained with a frequency of 14 and 17 out of 500 runs, which is high relative to the smaller values of Q. The result is consistent with the Q values obtained in Table 3. Additional tests on aggregated and disaggregated data sets yield the same conclusions.



(a) Fleet size versus neighborhood size

(b) Solution time versus neighborhood size

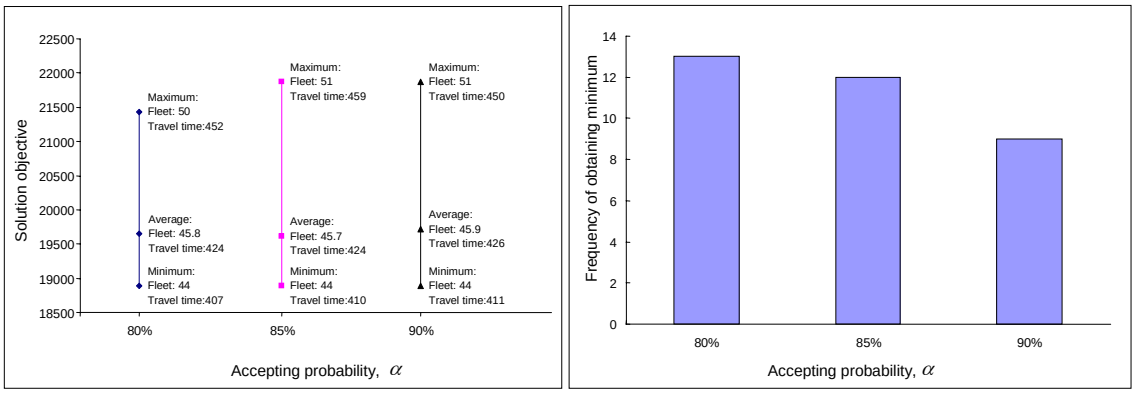
Figure 5: Solution quality and time as a function of neighborhood size, Q

4.3 GRP parameters

In this section, we evaluate how the selection of the GRP parameters, α, M, and K, impacts the effectiveness and efficiency of the combined GRP/VR method.

4.3.1 Degree of randomness

The value of α directly affects the composition of the routes generated with the GRP algorithm. A value of $\alpha = 1$ corresponds to a pure greedy algorithm, while lower α values allow more diversification in the routes generated. Empirical tests suggest that cost-minimizing solutions can be obtained with reasonable computational effort for α in the range from 80% to 90%.



(a) Range of corresponding solution objectives

(b) Solutions hitting minimum fleet size in 300 runs

Figure 6: Solution quality as a function of acceptance probability, α

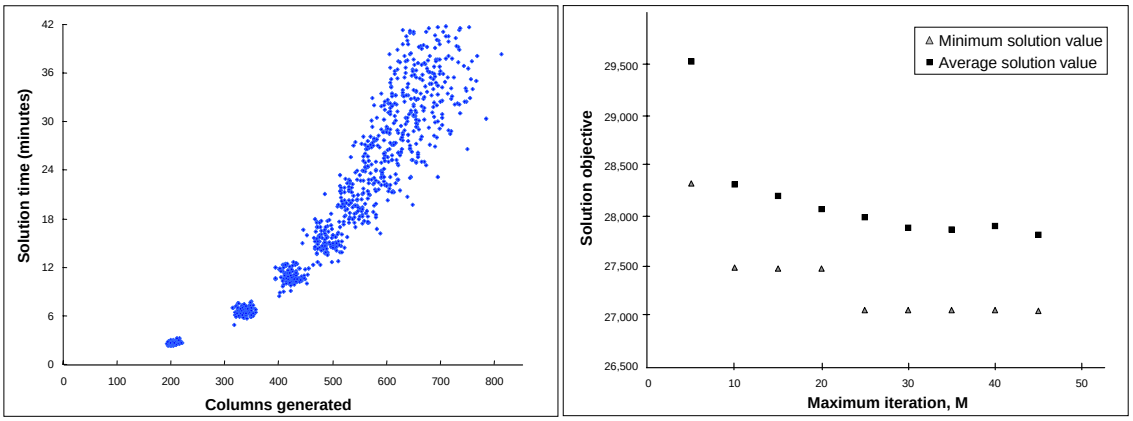
To illustrate the effect of varying α, we present empirical results for a typical test case with 100-

tasks and $K = 300$ replications, and α set at 80%, 85%, and 90% in Figure 6. Figure 6(a) compares the objective function to formulation (1) as a function of α. We convert the objective function values to fleet size and travel time in the figure. The figure plots the range of solutions (maximum, minimum and average) obtained in 300 replications of the GRP method with $\alpha = 80\%, 85\%, 90\%$. Note the lower variance in solution objective that occurs at $\alpha = 80\%$.

Figure 6(b) plots the frequency with which the minimum fleet size is found as a function of α. The frequency decreases with increases in α. As expected, the minimum fleet is more likely to be found with greater diversification. While this suggests that lower values of α are favorable, the greedy characteristic associated with higher values of α is more likely to result in reasonably good solutions in the first few replications. Hence, if the user chooses to perform fewer replications, higher α values should be used. In the following computational studies, we use $\alpha = 80\%$.

4.3.2 Number of columns generated

The total number of columns added can be controlled by the number of column-generation iterations allowed. The number of iterations is based on the computational resources available and the desired solution time limit.



(a) Solutions time as a function of columns generated

(b) Solution objective as a function of iteration count

Figure 7: Solution quality as a function of number of iterations in column generation, M

Figure 7 shows the result of empirical tests on a typical 100-task case. Figure 7(a) plots solution time for one replication of the GRP method as a function of number of columns generated and Figure 7(b) plots the solution objective as a function of the maximum number of iterations, M. As Figure 7(a) shows, the solution time grows exponentially with the number of columns generated. As Figure 7(b) shows, solving the problem with more columns (by increasing the number of iterations)

does not correspond to an improved objective value after M reaches some threshold value. Similar trends are observed in other test cases for both aggregated and disaggregated data. We conclude that smaller values of M may be used without affecting the chances of finding an improved solution and we use $M = 25$ as a reasonable tradeoff between quality and solution time. In practice, some instances are more sensitive to M, but the solution quality appears fairly stable for the instances tested.

4.3.3 Number of replications

The maximum number of replications, K, is determined by available computational time assuming that each replication is limited by a known time budget. For the purpose of parameter estimation, we use $K = 500$ for the aggregated tests cases. When solving the test cases, we perform as many replications as are possible in 1 hour for the aggregated test case, and 6 hours for the disaggregated test cases; each replication is limited to a budget of 300 CPU seconds for the smaller test cases of size less than 175 tasks and 1000 CPU seconds for the larger test cases.

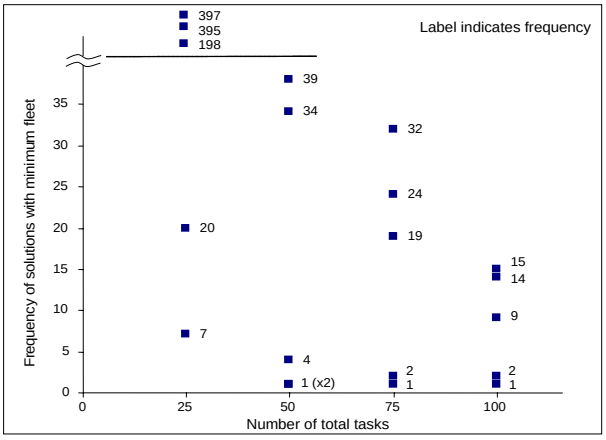


Figure 8: Frequency of finding the minimum fleet size

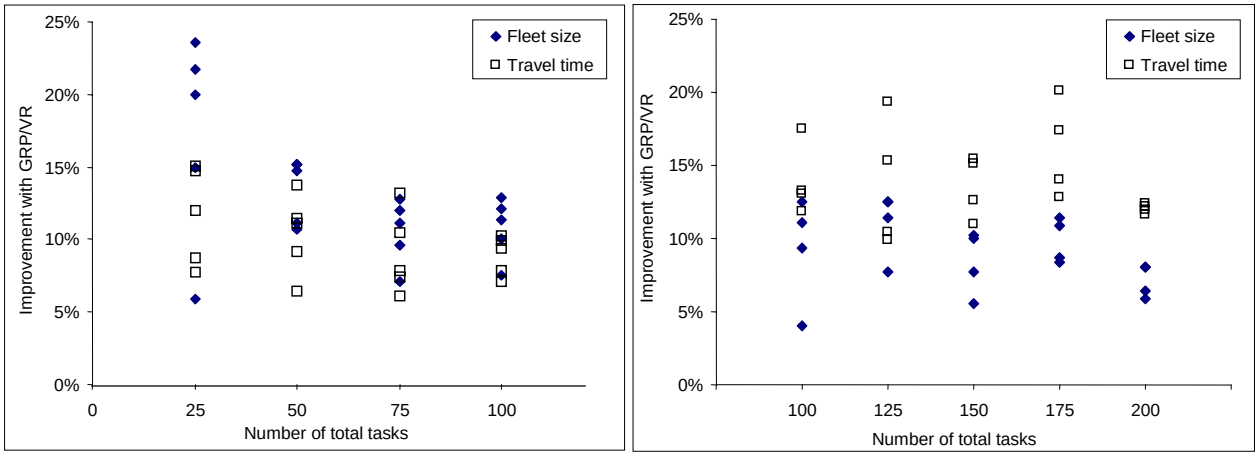
The fixed parameters are tested on the first set of test cases to estimate the probability of finding the best known solution with a given time budget. Figure 8 shows the frequency with which the best fleet size is found for the test cases. Empirical results for the larger disaggregated test cases ($\leq$ 175 tasks) indicate that the GRP method finds the best solution less often since fewer runs (300) are performed for these test cases as each replication consumes more of the 6-hour budget. We can estimate the probability of finding the minimum solution in a single run of the GRP method with the frequencies in Figure 8. Since each run is independent, we use a binomial model to approximate

the probability of finding at least one minimum solution with K replications of the GRP method. For instances with 100 tasks, a success frequency of 15 out of 500 runs corresponds approximately to a 95% probability of finding the best solution given only 100 replications (i.e., a time limit of one hour), and a success frequency of 2 out of 500 runs corresponds to a 33% probability in the same amount number of replications.

4.4 Numerical results

In this section, we apply the GRP/VR approach to two additional sets of test cases described in Tables 1(b) and 1(c) with the parameters as calibrated in Section 4.3. We show that the combined GRP/VR methods improves solution quality for the MRRP.

We evaluate the GRP/VR approach against a fixed radius, purely greedy method. Figures 9(a) and (b) present the improvement in fleet size and travel time for the aggregated and disaggregated test cases, respectively. Each data point refers to a single test case. In all aggregated test cases, improvements of 6%–24% are obtained in both fleet size and travel time. In all disaggregated test cases, improvements of 5%–20% are obtained. These results show that the combined GRP/VR method produces measurable improvements over results obtained with a fixed radius and pure greedy insertion method.



(a) Improvement in fleet size and travel time: aggregated data

(b) Improvement in fleet size and travel time: disaggregated data

Figure 9: Solution quality improvement with GRP/VR method

Table 4 presents detailed results for the aggregated test cases. The first four columns list the test cases and the number of tasks, the lower bound on fleet size obtained with the method from Smilowitz (2006) adapted for the VR method, and the minimum fleet size with the improved

GRP/VR method. The fifth column presents the solution time in minutes. The final two columns list the frequency with which the GRP/VR method finds the best known fleet-size and the frequency with which it finds a slightly inferior solution (best fleet-size + 1).

Test case	Tasks	Lower bound on fleet size	GRP/VR fleet size	Total solution time for 500 runs (mins.)	Solutions with best fleet size	Solutions with best fleet size+1
1	25	11	12	2.7	1	401
2	25	10	13	2.6	2	373
3	25	13	17	2.2	316	138
4	25	14	16	2.1	231	242
5	25	14	18	3.9	14	273
6	50	20	24	28.8	2	260
7	50	24	28	24.5	1	155
8	50	24	29	41.1	25	218
9	50	21	25	28.9	107	256
10	50	22	28	25.9	43	259
11	75	32	41	98.1	36	96
12	75	37	44	93.8	13	116
13	75	36	47	108.0	2	61
14	75	33	39	60.5	16	233
15	75	34	40	77.5	2	90
16	100	43	54	223.2	3	83
17	100	39	47	195.4	1	95
18	100	39	49	228.0	19	182
19	100	46	54	159.7	49	233
20	100	42	51	210.8	25	219

Table 4: Computational results for test cases: aggregated data

Table 5 presents similar results for the disaggregated test cases. For these instances, we impose a 6-hour time limit for the GRP/VR method. The fifth column in Table 5 lists the number of replications completed within 6 hours. As the table indicates, the number of replications depends on the number of tasks. For instance, roughly 300 replications of the GRP method are completed for the 100-task instances, and roughly 30 replications are completed for the 200-task instances. As shown in Table 5, we obtain the best known fleet-size with greater frequency in the smaller test cases. This is most likely due to limitations in computational resources: fewer replications are performed for the larger instances. Further, replications are interrupted by the solution time limit more often with larger instances. Although the solution time limit increases from 300 to 1000 seconds for larger instances, the limit still is not sufficient for some instances as the number of tasks grows. Despite these limitations, there are still significant improvements in the solution results from the GRP/VR method.

5 Conclusions

In this paper, we propose modeling and solution method improvements for the MRRP with flexible tasks. The VR method generates more effective and efficient execution choices for flexible tasks. The GRP method overcomes solution method deficiencies caused by a pure greedy heuristic in

Test case	Tasks	Lower bound on fleet size	GRP/VR fleet size	Replications in 6-hour runs	Solutions with best fleet size	Solutions with best fleet size+1
1	100	20	23	295	7	105
2	100	18	21	387	42	328
3	100	20	23	513	6	29
4	100	23	28	287	6	155
5	125	29	36	102	14	31
6	125	24	26	106	1	6
7	125	26	29	158	1	33
8	125	23	26	226	2	15
9	150	29	34	96	8	23
10	150	26	32	64	2	12
11	150	31	35	89	1	18
12	150	29	34	88	4	34
13	175*	34	44	37	3	8
14	175*	32	37	72	1	9
15	175*	33	39	44	1	6
16	175*	33	41	57	7	24
17	200*	39	45	37	2	16
18	200*	38	48	22	3	1
19	200*	39	46	27	8	3
20	200*	35	43	21	1	1

* Solution for each replication is limited to 1000 seconds; otherwise solution time limit is set to 300 seconds.

Table 5: Computational results for test cases: disaggregated data

column generation. These improvements combined improve the fleet size and travel time for test cases from industry.

We develop a nested partitioning algorithm to select the parameter Q for the VR method, as well as procedures to determine parameters for GRP method. Several techniques have been incorporated to make the method suitable for users with limited computational resources or a limitation on the amount of time that can be spent in searching for solutions. By intelligently limiting the set of flexible options through the VR method and utilizing this saved time to explore other possibilities, we obtain high quality solutions in a limited number of replications. The combination of these two techniques – imposing an intelligent limit on the set of choices, and using a randomized heuristic – results in improved solution quality and greater flexibility to solve the MRRP.

Extensions of the work will explore heuristics to solve formulation (1). The solution method could also be extended to a GRASP-like mechanism by implementing a local search mechanism to improve routing. Further extensions could include more complex control of the search with diversification and intensification phases, in a manner similar to Tabu Search mechanisms which are employed in Nanry and Barnes (2000); Combs and Moore (2004).

Acknowledgment:

This research has been supported by the National Science Foundation, grant DMI–0348622.

Appendix

The Greedy Randomized Adaptive Search Procedure (GRASP) is a metaheuristic introduced by Feo and Resende (1989). We refer the reader to Feo and Resende (1989) for details on the general GRASP metaheuristic and to Kontoravdis and Bard (1995) for details on GRASP as it relates to vehicle routing problems. The general GRASP algorithm consists of two main phases: *construction* in which an initial solution is created using randomization and *local search* in which the current solution is updated from candidates within a neighborhood of solutions. The algorithm is replicated until a maximum number of replications is reached. The algorithm is summarized as follows.

Step 0: Initialization

(1) Determine the maximum replication number K

(2) Initialize the seed for randomization

(3) Set current count $k = 1$

Step 1: Construction Phase

(1) Let $Solution = \emptyset$

(2) Build a candidate list of components for the solution

(3) Update *Solution* by:

(a) randomly select one component from the candidate list to become part of the solution

(b) use greedy criteria to insert current component into *Solution*

(c) update the candidate list

(4) If candidate list is empty, go to Step 2

Else, return to (3)

Step 2: Local Search Phase

(1) Build a complete neighborhood list of current *Solution*

(2) Calculate the objective value for every candidate solution within the neighborhood

(3) Update *Solution* as the one with the best objective, which is local optimal

Step 3: Replicate

(1) If *Solution* is better than *BestSolution*, update $BestSolution = Solution$

(2) Increase current replication count k by one

(3) If stopping criteria are reached, STOP

Else, go to Step 1

References

Ball, M., Golden, B., Assad, A., and Bodin, L. (1983). Planning for truck fleet size in the presence of common-carrier options. *Decision Sciences*, **14**(1), 103–120.

Bodin, L., Mingozzi, A., Baldacci, R., and Ball, M. (2000). The roll-on, roll-off vehicle routing problem. *Transportation Science*, **34**(3), 271–288.

Bräysy, O. and Gendreau, M. (2000). Vehicle routing problem with time windows, Part ii: Metaheuristics. *Transportation Science*, **39**(1), 119–139.

Campbell, A. and Savelsbergh, M. (2004). Efficiently handling practical complexities in insertion heuristics. *Transportation Science*, **38**(3), 369–378.

Carreto, C. and Baker, B. (1999). A grasp interactive approach to the vehicle routing problem with backhauls. *In Proc. of the Third Metaheuristics International Conference*, pages 125–128.

Combs, T. and Moore, J. (2004). A hybrid tabu search/ set partitioning approach to tanker crew scheduling. *Military Operations*, **9**(1), 43–56.

Corinescu, E. (2003). Drayage data files from 2002. Hub City Terminals, Inc.

Cullen, F., Jarvis, J., and Ratliff, H. (1981). Set partitioning based heuristics for interactive routing. *Networks*, **11**, 125–143.

Dahnke, B. (2003). Drayage data files from 1998. Laser Trucking, personal communication.

De Meulemeester, L., Laporte, G., Louveaux, F., and Semet, F. (1997). Optimal sequencing of skip collections. *Journal of Operational Research Society*, **48**, 57–64.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, **40**(2), 342–354.

Dror, M. (1994). Note on the complexity of the shortest path problem with resource constraints for column generation in VRPTW. *Operations Research*, **42**, 977–978.

Dror, M., Fortin, D., and Roucairol, C. (2000). Complexity issues for a redistribution problem. In D. Gardy and A. Mokkadem, editors, *Mathematics and Computer Science: Algorithms, Trees, Combinatorics, and Probabilities*, pages 165–176. Birkhauser.

Dror, M., Fortin, D., Parent, M., and Roucairol, C. (2001). Split pickup and delivery: the management of a fleet of free usage electric cars. *Preprint submitted to Elsevier Preprint*.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, **54**, 7–22.

Feo, T. and Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, **8**, 67–71.

Feo, T., Venkatraman, K., and Bard, J. (1991). A GRASP for a difficult single machine scheduling problem. *Computers and Operations Research*, **18**, 635–643.

Grosz, J. (2003). Personal communications. Cushing Transportation , Inc.

Kontoravdis, G. and Bard, J. (1995). A GRASP for the vehicle routing problem with time windows. *ORSA J. on Computing*, **7**, 10–23.

Matejcik, F. and Nelson, B. (1995). Two-stage multiple comparisons with the best for computer simulation. *Operations Research*, **43**, 633–640.

Morlok, E. K. and Spasovic, L. N. (1994). Approaches for improving drayage in rail-truck intermodal service. Research Report, University of Pennsylvania.

Nanry, W. and Barnes, J. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research B*, **34**, 107–121.

Ólafsson, S. and Kim, J. (2002). Simulation optimization. *In Proceedings of the 2002 Winter Simulation Conference*, pages 79–84.

Savelsbergh, M. and Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, **29**, 17–29.

Savelsbergh, M. and Sol, M. (1998). DRIVE: Dynamic routing of independent vehicles. *Operations Research*, **46**, 474–490.

Shi, L. and Ólafsson, S. (1997). An integrated framework for deterministic and stochastic optimization. *In Proceedings of the 1997 Winter Simulation Conference*, pages 358–365.

Sinha, P. and Zoltners, A. (1979). The multiple choice knapsack problem. *Operations Research*, **27**, 503–515.

Smilowitz, K. (2006). Multi-resource routing with flexible tasks: an application in drayage operations. *IIE Transactions*. Forthcoming.

Xu, H., Chen, Z.-L., Rajagopal, S., and Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, **37**(3), 347– 364.