Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 131450, 12 pages
http://dx.doi.org/10.1155/2014/131450

Research Article

Heuristics for Routing Heterogeneous Unmanned Vehicles with Fuel Constraints

David Levy, Kaarthik Sundar, and Sivakumar Rathinam

Mechanical Engineering, Texas A&M University, College Station, TX 77843, USA

Correspondence should be addressed to Sivakumar Rathinam; srathinam@gmail.com

Received 6 January 2014; Revised 17 February 2014; Accepted 23 February 2014; Published 13 April 2014

Academic Editor: Masoud Rokni

This paper addresses a multiple depot, multiple unmanned vehicle routing problem with fuel constraints. The objective of the problem is to find a tour for each vehicle such that all the specified targets are visited at least once by some vehicle, the tours satisfy the fuel constraints, and the total travel cost of the vehicles is a minimum. We consider a scenario where the vehicles are allowed to refuel by visiting any of the depots or fuel stations. This is a difficult optimization problem that involves partitioning the targets among the vehicles and finding a feasible tour for each vehicle. The focus of this paper is on developing fast variable neighborhood descent (VND) and variable neighborhood search (VNS) heuristics for finding good feasible solutions for large instances of the vehicle routing problem. Simulation results are presented to corroborate the performance of the proposed heuristics on a set of 23 large instances obtained from a standard library. These results show that the proposed VND heuristic, on an average, performed better than the proposed VNS heuristic for the tested instances.

1. Introduction

Small unmanned vehicles (UVs) are being used in several environmental monitoring applications [1–6] to collect information such as temperature, moisture, and humidity. These applications require UVs to visit specific target locations and monitor large swathes of land to collect data. Even though there are both economical and operational benefits [7] in using small UVs, they also come with other resource constraints due to their size and limited payload. Typically, small UVs due to their limited fuel capacity may have to revisit the depots (or fuel stations) multiple times for refueling while executing a surveillance mission. Path planning for small vehicles becomes critical in this scenario if the available resources such as fuel must be used as efficiently as possible.

This paper considers a fundamental routing problem that arises in these monitoring applications and is stated as follows: given a set of target locations, fuel stations (or depots) and UVs, find a path for each vehicle such that each target is visited at least once by a vehicle, each vehicle satisfies the fuel constraint as it traverses along its respective path and the travel cost of all the vehicles is a minimum. The travel cost we consider is the total fuel consumed by all the vehicles. To simplify the problem, we assume that the fuel consumed by a vehicle is directly proportional to the distance traveled by the vehicle. The vehicles are expected to refuel at the fuel stations as they run out of fuel. Vehicles are heterogeneous as they are allowed to carry fuel tanks with different capacities. This problem is referred to as the multiple, heterogeneous unmanned vehicle routing problem (MHUVRP).

In the absence of fuel constraints, MHUVRP is a generalization of the traveling salesman problem (TSP) and is NP-hard [8]. The difficulty in solving the TSP is further compounded when multiple vehicles are considered and even more when fuel constraints are imposed on these vehicles. Therefore, the focus of this paper is on developing heuristics that can find good solutions to the MHUVRP as quickly as possible. We accomplish this through the framework of the variable neighborhood search (VNS) and variable neighborhood descent (VND). VNS and VND are metaheuristics [9] used to solve difficult combinatorial and global optimization problems. These are iterative algorithms where in each iteration, the algorithms search through multiple neighborhoods of the current feasible solution to find a feasible solution with lower cost. The use of multiple neighborhoods allows the solution in the VNS and VND heuristics to move away from local optima as quickly as possible.

Routing for UVs has been addressed by several authors in [10–12]. The single vehicle version of the MHUVRP has been addressed by Khuller at al. [13] and Sundar and Rathinam [14]. Authors in [13] present an approximation algorithm for the symmetric version of the problem. An α-approximation algorithm is a polynomial time algorithm that produces a solution whose cost is at most α times the optimal cost for any instance of the problem. Authors in [14] present an approximation algorithm for the asymmetric version of the problem. In addition to an approximation algorithm, 2-opt and 3-opt heuristics were presented in [14] to find good feasible solutions to the single vehicle problem. Computational results [14] showed that the k-opt heuristics in combination with the approximation algorithm can produce near-optimal solutions for single vehicle instances with 25 targets within a couple of seconds of CPU time.

The MHUVRP is also closely related to routing problems with intermediate facilities [15, 16]. Other variants of the MHUVRP have also been studied in the literature. Authors in [17] consider a multiple vehicle TSP incorporating time windows and equity constraints. Approximation algorithms for a heterogeneous multiple vehicle TSP and a Hamiltonian path problem were studied in [18, 19]. Oberlin et al. [20] present a transformation of a heterogeneous multiple vehicle, multiple depot TSP into an asymmetric TSP so that algorithms for the standard TSP can be put to good use.

The contributions of this paper are as follows.

(i) We develop algorithms based on the VNS and VND heuristics proposed by Hansen and Mladenovic [9] for the MHUVRP using multiple neighborhoods.

(ii) To understand the effect of choosing different neighborhoods and initial solutions on the quality of the final solutions, we implement the proposed algorithms and test them on *large* problem instances from the multiple depot vehicle routing library by Cordeau [21]. These results give insights as to which of the neighborhoods provide substantial improvements in the search process and the order in which the neighborhoods must be selected for the VNS/VND algorithms to obtain good solutions for the MHURVP.

2. Problem Statement

Let there be K vehicles denoted by $v_1, v_2, \ldots, v_K$ with fuel capacities denoted by $L_1, \ldots, L_K$, respectively. Without loss of generality, we assume that $L_1 \leq L_2 \cdots \leq L_K$. Let T denote the set of targets to be visited, and let D denote the set of depots (or fuel stations) that are available. Let $V := T \cup D$. Each vehicle is initially located at one of the depots. The problem is formulated on a complete undirected graph $G = (V, E)$ where E is the set of edges joining any two vertices in V. Let the amount of fuel (travel cost) required to travel between any two vertices $x, y \in V$ be represented by f_{xy}. The travel costs are assumed to be symmetric and satisfy the triangle inequality; that is, for distinct vertices $x, y, z \in V$, we have $f_{xy} = f_{yx}$ and $f_{xy} + f_{yz} \geq f_{xz}$. Additionally, for any target $t \in T$, it is assumed that there is at least one vehicle v_i and a depot $d \in D$ such that $2 \cdot f_{td} \leq L_i$. This is a reasonable assumption because target t will be unreachable for any vehicle if this assumption is not true.

A tour for a vehicle is denoted by a sequence of vertices $(s, t_1, \ldots, t_p, s)$ where $t_i \in V$ for $i = 1, \ldots, p$ and $s \in D$ is the depot corresponding to the initial location of the vehicle. The cost of traveling a tour is defined as the sum of the cost of traveling all the edges in the tour. A tour satisfies the fuel constraint for its corresponding vehicle if the vehicle does not run out of fuel while traversing its tour. The objective of MHUVRP is to find K tours, one for each vehicle, such that each target is visited at least once by some vehicle, the tours satisfy the respective fuel constraints of the vehicles and the total cost of traveling all the tours is a minimum.

3. Algorithms

Heuristics based on the variable neighborhood search (VNS) are developed in this section for solving the MHUVRP. VNS was proposed by Hansen and Mladenovic in [9, 22]. The main idea of the VNS is to perform a local search systematically using multiple neighborhoods. It explores increasingly distant neighborhoods of the current solution iteratively and jumps from its current solution in the solution space to a new one if and only if an improvement has been made.

The steps of the basic VNS are shown as in Algorithm 1. In Algorithm 1, N_κ ($\kappa = 1, \ldots, \kappa_{\max}$) denotes a finite set of neighborhood structures. The stopping conditions may include criteria based on the allowable CPU time, maximum number of iterations, or maximum number of iterations between any two improvements. One can observe that the basic VNS heuristic also contains a probabilistic component in the shaking phase. The shaking step is a characteristic feature of the VNS heuristic and it allows the algorithm to get out of a local optimum. The solution obtained from the local search phase is compared to the incumbent solution and is accepted as a new starting point if an improvement can be made; otherwise, it is rejected. Therefore, the VNS procedure is a descent, first-improvement method with randomization. This method can also be used without the randomization phase. Such a method, named as the variable neighborhood descent (VND) heuristic [9], is the same as the VNS heuristic save for the absence of the shaking phase. In the following sections, we describe our algorithms for generating the initial solutions and the improvement procedures in detail.

4. Initial Solution: Construction Phase

Construction heuristics are developed by generalizing the approach of the single vehicle algorithm in [13] to multiple vehicles. In particular, we develop two construction heuristics that provide two initial feasible solutions for the MHUVRP. In the following discussion, we summarize the main steps of these construction heuristics and give the details in the appendix.

In the first construction heuristic, we first compute a new, modified traveling cost for each vehicle that also includes the fuel constraints of the vehicle. This traveling cost will also include the extra fuel a vehicle may require

```
Initialization: Select a set of neighborhood structures denoted by $N_\kappa$ ($\kappa = 1, \ldots, \kappa_{\max}$) that will be used in
the search; find an initial solution $x$; choose a stopping condition.
Repeat the following until the stopping condition is met:
    (1) Set $\kappa \leftarrow 1$.
    (2) Repeat the following steps until $\kappa = \kappa_{\max}$:
      (a) Shaking: Generate a feasible solution $x'$ at random from the $\kappa$th neighborhood of $x$ ($x' \in N_\kappa(x)$);
      (b) Local search: Apply a local search method with $x'$ as initial solution; denote the so obtained
          local optimum as $x''$;
      (c) Move or not: If $x''$ is better than incumbent $x$, set $x \leftarrow x''$ and continue the search with $N_1$ ($\kappa \leftarrow 1$);
          otherwise set $\kappa \leftarrow \kappa + 1$;
```

Algorithm 1: Steps of the basic VNS by Hansen and Mladenovic.

to visit all the refueling stops when it traverses between any two targets. We compute this new traveling cost based on the approach presented in [13]. For completeness, we have presented the details of this approach in the appendix of this article. Suppose $l^i(x, y)$ denote the modified cost of traveling from target x to target y for the ith vehicle including the refueling stops. It is easy to verify that these traveling costs monotonically increase with the index of the vehicles since the vehicles are ordered such that the fuel capacity of the vehicles decreases, that is, $l^1(x, y) \leq l^2(x, y) \cdots \leq l^K(x, y)$. Once these traveling costs are computed, the primal-dual heuristic presented in [23] is used to assign the targets to the vehicles. After partitioning the targets to the vehicles, the best single vehicle algorithm available in [14, 24] is used to find a feasible tour for each vehicle. This construction heuristic is referred to as the *Approximate Primal Dual (APD)* algorithm.

In the second construction heuristic, each target is assigned to the nearest initial depot of the vehicle based on the Euclidean distances between the targets and depots. Once the targets are partitioned among the vehicles, the best single vehicle algorithm available in [14, 24] is used to find a feasible tour for each vehicle. This construction heuristic is referred to as the *Voronoi* algorithm. Note that the two construction heuristics primarily differ only in the partitioning of the targets.

5. Variable Neighborhood Search: Improvement Phase

This section explains the choice of neighborhoods and the main steps in the improvement phase of the VNS algorithm (Algorithm 1) that includes the *shaking* step, *local search,* and the *move or not* step.

5.1. Neighborhood Selection. We use four neighborhoods; three of these are intra-route neighborhoods (2-exchange [25], 3-exchange [25], and depot exchange [24]) and the last one is an inter-route neighborhood (relocate [26]). Any two solutions can be present in an intra-route neighborhood only if the assignment of targets to the vehicles are the same in both the solutions. On the other hand, inter-route neighborhood considers solutions where the assignment of targets to the vehicles are different.

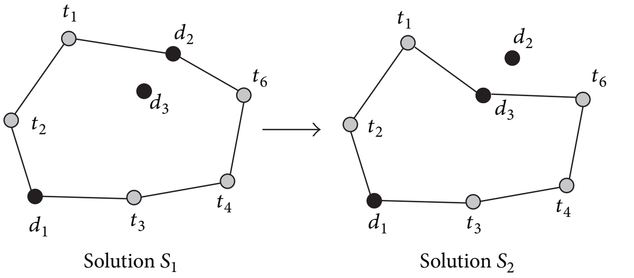


Figure 1: Depot exchange neighborhood; tour for some vehicle v_i is shown. The solution S_2 is contained in the depot exchange neighborhood of solution S_1.

We first formally define all the neighborhoods in the ensuing discussion. A solution S_2 is said to be in the k-*exchange* neighborhood of solution S_1 if there is exactly one vehicle v_i such that

(i) S_2 and S_1 may differ only in the tours of vehicle v_i, and

(ii) the tour for vehicle v_i in S_2 differs from the tour for vehicle v_i in S_1 by at most k edges.

A solution S_2 is said to be in the *depot exchange* neighborhood (Figure 1) of solution S_1 if there is exactly one vehicle v_i such that

(i) S_2 and S_1 differ only in the tours of vehicle v_i, and

(ii) the tour for vehicle v_i in S_2 differs from the tour for vehicle v_i in S_1 by exactly two edges and one depot in $D \setminus u_i$ where u_i denotes the initial depot of vehicle v_i.

A solution S_2 is said to be in the *relocate* neighborhood (Figure 2) of solution S_1 if there are exactly two distinct vehicles v_i and v_j satisfying the following conditions:

(i) S_2 and S_1 differ only in the tours of vehicles v_i, v_j,

(ii) there is exactly one target u such that u is visited by v_i in S_1 whereas u is visited by v_j in S_2.

5.2. Shaking. The shaking step is the randomization part of the VNS heuristic, the set of neighborhoods that are used in this shaking step form the core of the VNS. The purpose

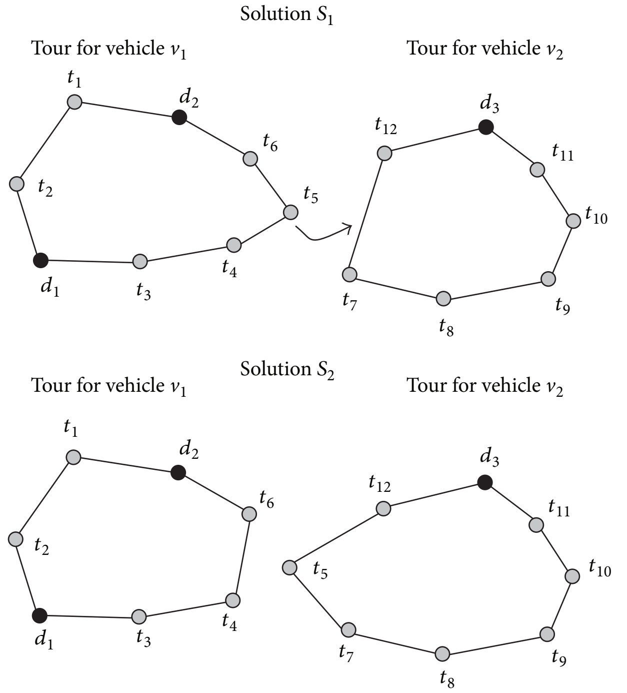


FIGURE 2: Relocate neighborhood; the solution S_2 is in the relocate neighborhood of S_1.

of these neighborhoods is to sufficiently perturb the initial solution (computed in Section 4), while ensuring that this new solution keeps certain aspects of the initial incumbent. In this step, a random feasible solution of the currently selected neighborhood of x is found and denoted as x'. This random selection enables the algorithm to avoid local optima in the local search procedure. For certain neighborhoods, it may be possible that there are no feasible neighbors; that is, the neighborhood of x is empty. In this special case, the shaking step is skipped, and x' is the same as x. Figure 3(a) shows the solution space where each solution is shown using a black dot, the neighborhood of x is denoted by a circle, and the outcome of the shaking step is denoted as x'.

5.3. Local Search. The output solution (x') from the shaking procedure is then improved using a local search procedure. The local search examines all the feasible neighbors of x' in the current neighborhood and finds the solution x'' which has the least cost. Again, it may be possible that the current neighborhood of x' is empty. If this is the case, x'' is set to be the same as x'.

5.4. Move or Not: Acceptance Decision. In this step, the cost of x'' is compared to the cost of x. The two possible outcomes of interest are when the cost of x'' is less than the cost of x and when the cost of x'' is greater than or equal to the cost of x. In the first case, x is set to be x'' (see Figure 3(b)). When the first outcome is true, the first neighborhood is set as the "current" neighborhood ($\kappa \leftarrow 1$ in Algorithm 1), and computations continue with the shaking step. When the second outcome is true, x'' is forgotten, and the next neighborhood is set as the "current" neighborhood ($\kappa \leftarrow \kappa+1$ in Algorithm 1). If there is no next neighborhood, that is, the "current" neighborhood is the last neighborhood designated, the algorithm terminates.

6. Computational Analysis

The VNS/VND heuristic proposed in Section 5 was implemented using Visual C++ and the elements of the Standard Template Library (STL). Simulations were performed on a 2.4 GHz AMD Phenom machine. The problem instances on which the simulations were performed derives from the Cordeau's benchmark instances [21] for the multiple depot vehicle routing problem. Additional information was added to these benchmark instances to satisfy the requirements of the MHUVRP as follows.

(i) The number of customers in the benchmark instances was used as the number of targets for the MHUVRP.

(ii) The depot locations in the benchmark instances were used as the initial depot locations of all the vehicles.

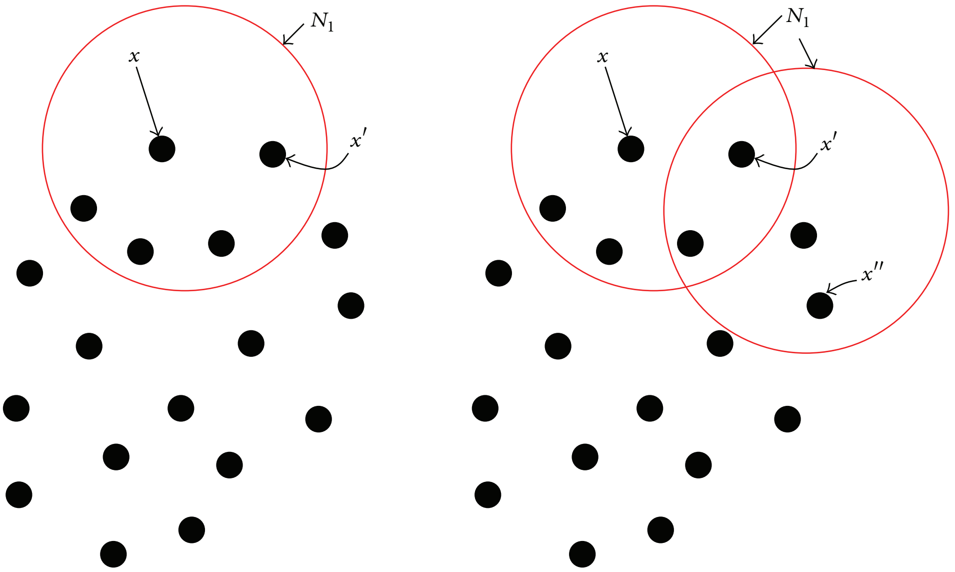
(iii) The number of vehicles in the MHUVRP was set to be equal to the number of depots in the benchmark instances.

(iv) The fuel capacity of the kth vehicle in an instance was computed using the formula $L_k := S - 15(k-1)$ where S is the size of the surveillance area. S was set to be equal to $\max_{i,j\in V}|x_i - x_j|$ where x_i denotes the x coordinate of vertex i (rounded to the nearest 100 units).

The above modifications resulted in 23 distinct instances (listed in Table 1) on which the computational study was performed. The number of targets and vehicles in these instances varied from 50 to 360 and 2 to 9, respectively. We use a maximum allowable time of 1000 secs as the stopping condition for the heuristics.

6.1. Quality of the Initial Feasible Solutions. Table 1 shows the cost of the initial feasible solutions produced by the two construction heuristics. On an average, these results show that the initial feasible solution produced by the APD algorithm was superior to the initial feasible solution produced by the Voronoi partitioning algorithm for a majority of instances.

6.2. Analysis on the Best Sequence of Neighborhoods. In this section, we first study the effect of the choice of the neighborhoods used in the VNS and VND algorithms. The % improvement in the quality of solutions using an algorithm is defined as $100 \times ((\text{Cost}_I - \text{Cost}_{\text{alg}})/\text{Cost}_I)$ where Cost_{alg} denotes the travel cost of the final solution produced by the algorithm and Cost_I denotes the travel cost of the initial solution produced by a construction heuristic. As there are two construction heuristics used, we evaluate the effect of the sequence of neighborhoods used on both the initial solutions obtained using the construction heuristics. Table 2 shows the average improvement in the quality of the solutions using a sequence of 4 neighborhoods. We only report the sequences of the neighborhoods that produced the best



(a) Shaking step (b) Move or not step

Figure 3: Shaking step and move or not step of the VNS heuristic.

Table 1: Comparison of the initial feasible solutions.

Instance	Targets	Vehicles	APD Cost (Cost_{apd})	Voronoi Cost (Cost_v)	$\text{Cost}_v/\text{Cost}_{\text{apd}}$ (%)
p01	50	4	2483.04	2474.2	99.64
p03	75	5	2170.27	3666.57	168.95
p04	100	2	2569.91	4354.53	169.44
p05	100	2	2588.21	2645.91	102.23
p06	100	3	5089.49	4158.92	81.72
p07	100	4	5759.59	5798.38	100.67
p08	249	2	40044.1	59003.2	147.35
p09	249	3	37783.8	51802.7	137.10
p10	249	4	34990.6	50382.6	143.99
p11	249	5	34610.9	71234.3	205.81
p12	80	2	5917.04	12618	213.25
p15	160	4	23619.6	9628.57	40.77
p21	360	9	77354.6	19131.9	24.73
pr01	48	4	6964.58	9940.97	142.74
pr02	96	4	7625.7	7854.82	103.00
pr03	144	4	21303.9	29336.8	137.71
pr04	192	4	14659.2	19382	132.22
pr05	240	4	9780.66	24196.3	247.39
pr06	288	4	24778.5	9144.85	36.91
pr07	72	6	4026.09	10006.3	248.54
pr08	144	6	9340.81	14959.8	160.16
pr09	216	6	12207.2	16921.7	138.62
pr10	288	6	9119.63	30491.9	334.35

improvements. In general, the order of the neighborhoods did not significantly affect the improvement in the quality of the solutions.

Table 3 shows the average improvement in the quality of the solutions using a sequence of 3 neighborhoods. These results show that the VND heuristic provides more improvement in the quality of the solutions, on an average, compared to the VNS heuristic. We also note that majority of the best sequences has 3-opt neighborhood as the last search neighborhood. In general, we found that the addition of the

Table 2: Influence of 4 neighborhoods sequences on improving quality.

Scheme	Construction heuristic	N_1	N_2	N_3	N_4	Average improvement (%)	Average computation time (sec)
VND	APD	2-opt	Relocate	3-opt	Depot	33.92	78.04881
VND	APD	2-opt	Relocate	Depot	3-opt	33.79	77.33881
VND	APD	2-opt	3-opt	Relocate	Depot	33.76	197.7543
VNS	APD	Relocate	2-opt	3-opt	Depot	27.52	70.8876
VNS	APD	Relocate	2-opt	Depot	3-opt	27.39	75.12725
VNS	APD	Relocate	Depot	2-opt	3-opt	26.99	75.44715
VND	Voronoi	Depot	Relocate	3-opt	2-opt	38.08	310.7845
VND	Voronoi	Relocate	3-opt	Depot	2-opt	37.94	304.7469
VND	Voronoi	Depot	Relocate	2-opt	3-opt	36.85	203.9593
VNS	Voronoi	Relocate	Depot	2-opt	3-opt	42.22	221.0236
VNS	Voronoi	Relocate	Depot	3-opt	2-opt	38.23	741.5726
VNS	Voronoi	Depot	Relocate	3-opt	2-opt	37.52	799.8579

3-opt neighborhood produced high quality solutions for the problem at the expense of computation time.

Table 4 shows the average improvement in the quality of the solutions using a sequence of 2 neighborhoods. This table provides a clear picture of the overall improvement capability of the different neighborhoods. These results show that the relocate and 3-opt neighborhoods appear most frequently followed by the 2-opt, indicating that the neighborhood combinations containing 3-opt or relocate are the most effective, even though they take an order or two of magnitude longer to run than the other neighborhood combinations. They also provide the best improvement in the quality of solutions on an average. Finally, Table 5 shows the effect of choosing a single neighborhood in the improvement of the quality of the solutions. Results in Table 5 shows a clear trade-off in the quality of the solutions produced by an algorithm versus the computation time the algorithm requires. In terms of solution quality, algorithms using the 3-opt neighborhood performed the best while using the maximum amount of computation time. On the other hand, the algorithms using only the relocate neighborhood ran quickly while producing an improvement of up to 17.56%.

6.3. Effect of the Construction Heuristics on the Final Cost. Table 6 shows the effect of the two construction heuristics on the quality of the final solutions obtained by the VNS and VND algorithms. In this table, Cost_v^f and $\text{Cost}_{\text{apd}}^f$ denote the cost of the final solution obtained by the algorithms using the Voronoi and the APD heuristic, respectively. The table shows the average values of $((\text{Cost}_v^f - \text{Cost}_{\text{apd}}^f)/\text{Cost}_{\text{apd}}^f) \times 100$ for some of the best combination of neighborhoods. For a majority of instances, these results show that the choice of the construction heuristic, on an average, does not significantly affect the quality of the final solutions produced by the algorithms. Figures 4 and 5 show the initial and the best solution obtained using the proposed algorithms for the instance *pr05*.

6.4. The Best Algorithm with the Neighborhood Combinations. In this subsection, we specify the best combination of the neighborhoods obtained for the instances and the corresponding cost of the final solutions obtained by the algorithms. For a single neighborhood, the *VND heuristic with 3-opt* (Table 7) performed the best with the initial solution provided by the APD heuristic. This heuristic improved the quality of the solutions by 22.96% on an average. For two neighborhoods, the *VND heuristic* with *relocate* as the first neighborhood and *3-opt* as the second neighborhood (Table 8) performed the best with the initial solution provided by the Voronoi heuristic. This heuristic improved the quality of the solutions by 39.42% on an average. We found that the relocate neighborhood in combination with either 2-opt or 3-opt were effective in improving the quality of the solutions substantially.

For three neighborhoods, the *VND heuristic* with *relocate*, *depot exchange, and 3-opt neighborhoods* (in the given sequence) performed the best with the initial solution provided by the Voronoi heuristic. In particular, they improved the quality of the solutions by 42.59% as shown in Table 9. For four neighborhoods, the *VNS heuristic with relocate, depot exchange, 2-opt, and 3-opt neighborhoods* (in the given sequence) performed the best with the initial solution provided by the Voronoi heuristic. This heuristic improved the quality of the solutions by 42.22% as shown in Table 10. In general, using four neighborhoods as compared to three neighborhoods only increased the computation time while not providing any substantial improvements in the solution quality.

7. Conclusion

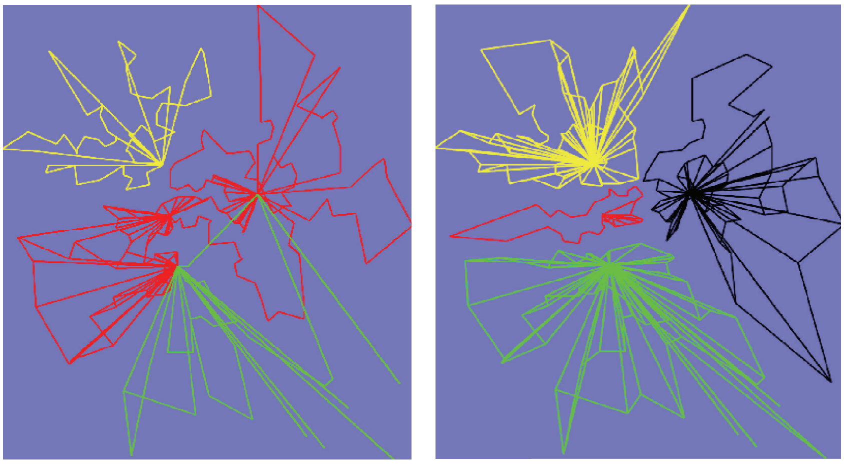
The effects of different neighborhoods and partitioning heuristics on benchmark instances for the MHUVRP were examined. The neighborhoods included the 2-opt, 3-opt, depot exchange, and the relocate neighborhoods. Simulation results showed that, on an average, the construction

Table 3: Influence of 3 neighborhoods sequences on improving quality.

Scheme	Construction heuristic	N_1	N_2	N_3	Average improvement (%)	Average computation time (sec)
VND	APD	2-opt	Relocate	3-opt	33.99	483.1837
VND	APD	2-opt	3-opt	Relocate	33.86	595.6045
VND	APD	Relocate	3-opt	2-opt	33.49	89.5769
VNS	APD	Relocate	2-opt	3-opt	27.52	71.3513
VNS	APD	Relocate	3-opt	2-opt	26.77	178.1168
VNS	APD	2-opt	Relocate	3-opt	25.87	79.254
VND	Voronoi	Relocate	Depot	3-opt	42.59	323.1872
VND	Voronoi	Relocate	2-opt	3-opt	36.24	225.4593
VND	Voronoi	Depot	Relocate	3-opt	35.97	891.1418
VNS	Voronoi	Relocate	2-opt	3-opt	37.37	241.9253
VNS	Voronoi	Relocate	3-opt	Depot	37.18	298.7079
VNS	Voronoi	Depot	Relocate	3-opt	36.93	795.2278

Table 4: Influence of 2 neighborhoods sequences on improving quality.

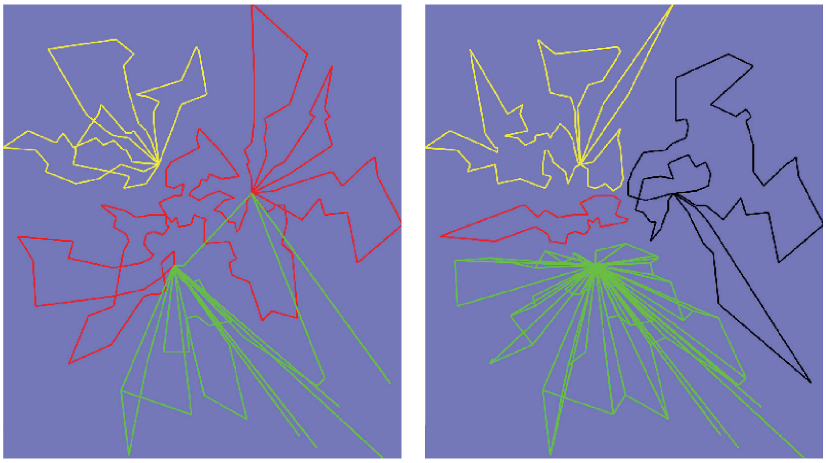
Scheme	Construction heuristic	N_1	N_2	Average improvement (%)	Average computation time (sec)
VND	APD	Relocate	2-opt	32.82	417.9984
VND	APD	2-opt	Relocate	32.15	422.4113
VND	APD	Relocate	3-opt	31.99	495.1695
VNS	APD	Relocate	3-opt	22.76	170.846
VNS	APD	3-opt	Relocate	21.24	173.4163
VNS	APD	3-opt	2-opt	20.38	121.7078
VND	Voronoi	Relocate	3-opt	39.42	753.4336
VND	Voronoi	3-opt	Relocate	33.73	1109.158
VND	Voronoi	Relocate	2-opt	30.32	11.05305
VNS	Voronoi	Relocate	3-opt	33.93	503.4305
VNS	Voronoi	3-opt	Relocate	29.22	479.8828
VNS	Voronoi	Relocate	2-opt	27.92	7.714526



Approximate primal dual

Voronoi

Figure 4: Initial solutions for the instance *pr*05 using the APD and Voronoi partitioning heuristics.



Approximate primal dual

Voronoi

Figure 5: The best solution obtained for the instance *pr*05 using the proposed heuristics.

Table 5: Influence of one neighborhood on improving quality.

Scheme	Construction heuristic	N_1	Average improvement (%)	Average computation time (sec)
VND	APD	3-opt	22.96	1462.62
VND	APD	2-opt	21.96	420.1842
VND	APD	Relocate	14.08	1.017391
VNS	APD	3-opt	17.32	109.5311
VNS	APD	2-opt	10.36	0.75705
VNS	APD	Relocate	8.13	0.30115
VND	Voronoi	3-opt	21.29	835.2096
VND	Voronoi	Relocate	17.56	1.6659
VND	Voronoi	2-opt	15.56	4.2504
VNS	Voronoi	3-opt	19.92	459.5533
VNS	Voronoi	2-opt	13.19	3.118789
VNS	Voronoi	Relocate	8.87	0.399421

heuristics did not affect the quality of the final solutions obtained by the algorithms despite the fact that the primal-dual heuristic produced better initial solutions compared to the Voronoi heuristic. Overall, the VND heuristic produced better solutions compared to the VNS heuristic. This result shows that including the shaking step in the VNS may not always yield better results. In general, heuristics with the 3-opt neighborhood provided the best solutions for majority of the instances. However, 3-opt neighborhoods can be replaced with 2-opt neighborhoods whenever possible, to obtain solutions of similar quality while reducing the computation time.

This paper mainly focused on developing heuristics for the MHUVRP. However, there are currently no algorithms in the literature that can find optimal solutions to the problem relatively quickly for the large instances considered in this paper. Therefore, future work for the MHUVRP can focus on finding good lower bounds and optimal solutions for these instances.

Appendix

The purpose of this computation is to find a path for every vehicle v_k to travel from any target $x \in T$ to any other target $y \in T$ such that the path can be a part of the feasible tour for v_k, satisfies all the refueling constraints for v_k, and is of minimum travel cost. Given a vehicle v_k, we note that the maximum amount of fuel available for the vehicle when it reaches target x in any tour is $L^k - \min_d f_{dx}$. Also, in any feasible tour, there must be at least $\min_d f_{yd}$ units of fuel left when the vehicle reaches target y so that the vehicle can continue to visit other vertices along its tour. Define $F_x := \min_d f_{dx}$ (also $F_x = \min_d f_{xd}$ since the travel costs are symmetric) for any $x \in T$. The first step of the algorithm finds a feasible path of least cost (also referred as the shortest path) such that the vehicle starts at target x with at most $L - F_x$ units of fuel and ends at target y with at least F_y units of fuel. If there is enough fuel available for the vehicle v_k to travel from x to y (or, if $L^k - F_x - F_y \geq f_{xy}$), the vehicle can directly reach y from x while respecting the fuel constraints. In this case, we say that the vehicle v_k can *directly* travel from x to y and the shortest path (also referred to as the *direct* path) is denoted by $\text{PATH}^k(x, y) := (x, y)$. The cost of traveling this shortest path is just f_{xy}. If the vehicle v_k *cannot directly* travel from x to y (if $L^k - F_x - F_y < f_{xy}$), the vehicle must visit some of the depots on the way before reaching target y. In this case, we find a shortest path using an auxiliary graph, (V', E'), defined on all the depots and the targets x, y, that is, $V' = D \cup \{x, y\}$ (illustrated in Figure 6).

Table 6: Comparison between the final solutions produced by the algorithms.

Scheme	N_1	N_2	N_3	N_4	$(\mathrm{Cost}^{f}_{\mathrm{apd}} - \mathrm{Cost}^{f}_{v})/\mathrm{Cost}^{f}_{\mathrm{apd}}$ (%)
VNS	Relocate	2-opt	Depot	3-opt	9.90
VND	Depot	2-opt	Relocate	3-opt	4.42
VND	Depot	2-opt	Relocate	3-opt	4.42
VNS	Relocate	3-opt	Depot		−18.64
VND	3-opt	Relocate	Depot		1.80
VNS	Relocate	2-opt	3-opt		8.73
VND	Relocate	2-opt	3-opt		5.85
VNS	Relocate	2-opt			−1.52
VND	Relocate	2-opt			6.47
VNS	3-opt	2-opt			−35.45
VND	Relocate	3-opt			2.10
VNS	Relocate				−51.36
VND	2-opt				−3.04
VND	Relocate				−34.98

Table 7: Results using the VND heuristic with 3-opt.

Instance	Initial cost (Cost_i)	Final cost (Cost_f)	$(\mathrm{Cost}_i - \mathrm{Cost}_f)/\mathrm{Cost}_i$ (%)	Computation time (sec)
p01	2483.04	1942.48	21.77	7.368
p03	2170.27	1172.44	45.98	4.822
p04	2569.91	1301.3	49.36	15.816
p05	2588.21	1468.28	43.27	30.267
p06	5089.49	2904.23	42.94	180.685
p07	5759.59	4456.7	22.62	195.286
p08	40044.1	40044.1	0.00	192.995
p09	37783.8	37783.8	0.00	142.009
p10	34990.6	34990.6	0.00	374.885
p11	34610.9	34610.9	0.00	30.816
p12	5917.04	2064.31	65.11	25.047
p15	23619.6	3105.85	86.85	5842.052
p21	77354.6	76936.9	0.54	9000
pr01	6964.58	6885.05	1.14	11.20039
pr02	7625.7	7625.7	0.00	9.709453
pr03	21303.9	21303.9	0.00	29.87566
pr04	14659.2	13465.5	8.14	943.9597
pr05	9780.66	8137.73	16.80	286.9558
pr06	24778.5	14108.8	43.06	32975.92
pr07	4026.09	2151.59	46.56	2.17765
pr08	9340.81	7385.68	20.93	178.9951
pr09	12207.2	9294.9	23.86	480.1001
pr10	9119.63	9119.63	0.00	50.35585

An edge is present in this graph only if traveling the edge can satisfy the fuel constraint. For example, as the vehicle v_k has at most $L^k - F_x$ units of fuel to start with, the vehicle can reach a depot d from x only if $f_{xd} \le L^k - F_x$. Therefore, E' contains an edge (x, d) if the constraint $f_{xd} \le L^k - F_x$ is satisfied. Similarly, the vehicle can travel from a depot d to target y only if there are at least F_y units of fuel remaining after the vehicle reaches y. Therefore, E' contains an edge (d, y) if the constraint $f_{dy} \le L^k - F_y$ is satisfied. In summary, the following are the edges present in E':

$$E' := \begin{cases} \left\{(x, d) : \forall d \in D, f_{xd} \le L^k - F_x\right\} \\ \cup \left\{(d_1, d_2) : \forall d_1, d_2 \in D, f_{d_1 d_2} \le L^k\right\} \\ \cup \left\{(d, y) : \forall d \in D, f_{dy} \le L^k - F_y\right\}. \end{cases} \tag{A.1}$$

Table 8: Results using the VND heuristic with relocate and 3-opt.

Instance	Initial Cost (Cost_i)	Final Cost (Cost_f)	($\text{Cost}_i - \text{Cost}_f$)/$\text{Cost}_i$ (%)	Computation time (sec)
p01	2474.2	1096.8	55.67	1.012
p03	3666.57	1253.25	65.82	16.18
p04	4354.53	1762.64	59.52	206.734
p05	2645.91	1465.84	44.60	38.875
p06	4158.92	1829.36	56.01	22.65
p07	5798.38	2224.81	61.63	48.702
p08	59003.2	59003.2	0.00	518.078
p09	51802.7	51802.7	0.00	557.311
p10	50382.6	50382.6	0.00	479.043
p11	71234.3	67496.8	5.25	9570.828
p12	12618	1573.43	87.53	8.731
p15	9628.57	2398.17	75.09	23.964
p21	19131.9	5543.1	71.03	58.40544
pr01	9940.97	9940.97	0.00	2.577926
pr02	7854.82	2425.53	69.12	16.12673
pr03	29336.8	22436	23.52	4844.833
pr04	19382	18776.1	3.13	508.4298
pr05	24196.3	17398.5	28.09	4531.525
pr06	9144.85	5088.55	44.36	252.208
pr07	10006.3	5294.35	47.09	2.493464
pr08	14959.8	8907.27	40.46	142.704
pr09	16921.7	5272.41	68.84	100.4253
pr10	30491.9	30491.9	0.00	46.63218

Table 9: Results using the VND heuristic with relocate, depot exchange and 3-opt.

Instance	Initial Cost (Cost_i)	Final Cost (Cost_f)	($\text{Cost}_i - \text{Cost}_f$)/$\text{Cost}_i$ (%)	Computation time (sec)
p01	2474.2	1096.8	55.67	0.962
p03	3666.57	1253.25	65.82	16.196
p04	4354.53	1634.81	62.46	186.685
p05	2645.91	1465.84	44.60	38.83
p06	4158.92	1829.36	56.01	22.633
p07	5798.38	2224.81	61.63	48.9
p08	59003.2	59003.2	0.00	515.802
p09	51802.7	51802.7	0.00	554.835
p10	50382.6	50382.6	0.00	476.858
p11	19327.5	4599.05	76.20	212.3943
p12	12618	1573.43	87.53	8.829
p15	9628.57	2398.17	75.09	24.106
p21	19131.9	5391.13	71.82	89.22671
pr01	9940.97	9940.97	0.00	2.634846
pr02	7854.82	2435.17	69.00	14.46503
pr03	29336.8	22475.2	23.39	4365.047
pr04	19382	18776.1	3.13	541.5995
pr05	24196.3	17403.7	28.07	4190.062
pr06	9144.85	5165.15	43.52	235.299
pr07	10006.3	5510.2	44.93	2.12073
pr08	14959.8	8689.09	41.92	179.7975
pr09	16921.7	5289.71	68.74	94.15305
pr10	30491.9	30491.9	0.00	46.17499

Table 10: Results using the VNS heuristic with relocate, depot exchange, 2-opt and 3-opt.

Instance	Initial Cost (Cost_i)	Final Cost (Cost_f)	$(\text{Cost}_i - \text{Cost}_f)/\text{Cost}_i$ (%)	Computation time (sec)
p01	2474.2	694.271	71.94	0.232
p03	3666.57	1143.62	68.81	4.915
p04	4354.53	1551.03	64.38	58.451
p05	2645.91	1476.75	44.19	19.571
p06	4158.92	1474.53	64.55	12.117
p07	5798.38	1464.33	74.75	26.84
p08	59003.2	59003.2	0.00	519.381
p09	51802.7	51802.7	0.00	559.957
p10	50382.6	50346.4	0.07	470.61
p11	71234.3	65498.1	8.05	1910.843
p12	12618	2385.06	81.10	53.939
p15	9628.57	2409.39	74.98	11.693
p21	19131.9	5492.56	71.29	78.08324
pr01	9940.97	9940.97	0.00	2.691766
pr02	7854.82	2107.41	73.17	14.26489
pr03	29336.8	22494.6	23.32	974.1806
pr04	19382	18603.2	4.02	213.7126
pr05	15198.4	9801.09	35.51	484.938
pr06	9144.85	5240.83	42.69	98.146
pr07	10006.3	4827.39	51.76	1.632319
pr08	14959.8	8357.23	44.14	64.22047
pr09	16921.7	4676.42	72.36	168.083
pr10	30491.9	30473.4	0.06	47.34827

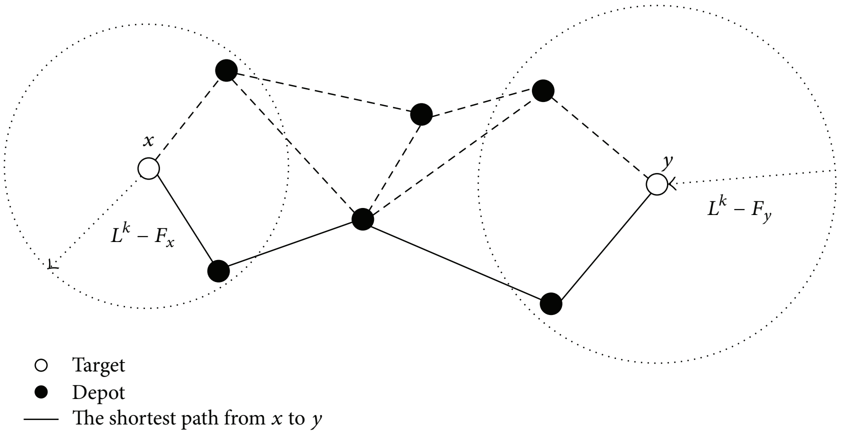


Figure 6: The first step of the algorithm: the solid edges represent the shortest path $\text{PATH}^k(x, y)$ for the vehicle v_k, to travel from target x to target y, and the cost of traveling this path is denoted by $l^k(x, y)$.

Any path starting at x and ending at y in this auxiliary graph will require the vehicle v_k to carry at most $L^k - F_x$ units of fuel at target x, satisfy all the fuel constraints, and reach target y with at least F_y units of fuel left. Also, we let the cost of traveling any edge $(i, j) \in E'$ to be equal to f_{ij} (as defined in Section 2). Now, we use Dijkstra's algorithm [27] to find a shortest path to travel from x to y. This shortest path (also referred to as the *indirect* path using intermediate depots) can be represented as $\text{PATH}^k(x, y) := (x, d_1, d_2, \ldots, y)$. Let the length of this path be represented by $l^k(x, y)$. This new cost function is computed for every vehicle between every pair of targets and is denoted by l^k where $k = 1, \ldots, n$.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant no. 1015066 and the Air Force Office of Scientific Research under Grant no. FA9550-10-1-0392.

References

[1] K. Geiser, D. Slack, E. Allred, and K. Stange, "Irrigation scheduling using crop canopy-air temperature differences," *Transactions of the American Society of Agricultural and Biological Engineers*, vol. 25, no. 3, pp. 689–694, 1982.

[2] R. D. Jackson, S. B. Idso, R. J. Reginato, and P. J. Pinter Jr., "Canopy temperature as a crop water stress indicator," *Water Resources Research*, vol. 17, no. 4, pp. 1133–1138, 1981.

[3] J. A. Curry, J. Maslanik, G. Holland, and J. Pinto, "Applications of aerosondes in the arctic," *Bulletin of the American Meteorological Society*, vol. 85, no. 12, pp. 1855–1861, 2004.

[4] "NOAA and partners conduct first successful unmanned aircraft hurricane observation by ying through Ophelia," NOAA News Online, 2005, http://www.noaanews.noaa.gov/stories2005/s2508.htm

[5] C. E. Corrigan, G. C. Roberts, M. V. Ramana, D. Kim, and V. Ramanathan, "Capturing vertical profiles of aerosols and black carbon over the Indian Ocean using autonomous unmanned aerial vehicles," *Atmospheric Chemistry and Physics*, vol. 8, no. 3, pp. 737–747, 2008.

[6] T. Zajkowski, S. Dunagan, and J. Eilers, "Small UAS communications mission," in *Proceedings of the 11th Biennial USDA Forest Service Remote Sensing Applications Conference*, Salt Lake City, Utah, USA, 2006.

[7] M. Lt. Comeaux, "Soldiers train with Raven UAV's," United States Army, Online Web Article, 2007, http://www.army.mil/article/5644/soldiers-train-with-raven-uavs/.

[8] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and its Variations*, vol. 12 of *Combinatorial Optimization*, Kluwer Academic, Dordrecht, The Netherlands, 2002.

[9] P. Hansen and N. Mladenovic, *Variable Neighborhood Search*, 1997.

[10] S. Kim, P. Silson, A. Tsourdos, and M. Shanmugavel, "Dubins path planning of multiple unmanned airborne vehicles for communication relay," *Proceedings of the Institution of Mechanical Engineers G: Journal of Aerospace Engineering*, vol. 225, no. 1, pp. 12–25, 2011.

[11] M. Innocenti, L. Pollini, and A. Bracci, "Cooperative path planning and task assignment for unmanned air vehicles," *Proceedings of the Institution of Mechanical Engineers G: Journal of Aerospace Engineering*, vol. 224, no. 2, pp. 121–131, 2010.

[12] J. Karimi and S. H. Pourtakdoust, "Integrated motion planning and trajectory control system for unmanned air vehicles," *Proceedings of the Institution of Mechanical Engineers G: Journal of Aerospace Engineering*, vol. 227, no. 1, pp. 3–18, 2013.

[13] S. Khuller, A. Malekian, and J. Mestre, "To filll or not to fill: The gas station problem," in *Algorithms - ESA 2007*, L. Arge, M. Ho mann, and E. Welzl, Eds., vol. 4698 of *Lecture Notes in Computer Science*, pp. 534–545, Springer, Berlin, Germany, 2007.

[14] K. Sundar and S. Rathinam, "Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 287–294, 2014.

[15] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno, "Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length restrictions," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 3, pp. 209–223, 2004.

[16] M. Polacek, K. F. Doerner, R. F. Hartl, and V. Maniezzo, "A variable neighborhood search for the capacitated arc routing problem with intermediate facilities," *Journal of Heuristics*, vol. 14, no. 5, pp. 405–423, 2008.

[17] R. F. Dell, R. Batta, and M. H. Karwan, "The multiple vehicle TSP with time windows and equity constraints over a multiple day horizon," *Transportation Science*, vol. 30, no. 2, pp. 120–133, 1996.

[18] S. Yadlapalli, S. Rathinam, and S. Darbha, "3-approximation algorithm for a two depot, heterogeneous traveling salesman problem," *Optimization Letters*, vol. 6, no. 1, pp. 141–152, 2012.

[19] R. Doshi, S. Yadlapalli, S. Rathinam, and S. Darbha, "Approximation algorithms and heuristics for a 2-depot, heterogeneous Hamiltonian path problem ," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1434–1451, 2011.

[20] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesmen problem: heterogeneous, multiple depot, multiple UAV routing problem," *IEEE Robotics and Automation Magazine*, vol. 17, no. 4, pp. 70–77, 2010.

[21] J. F. Cordeau, "The multiple depot vehicle routing problem benchmark instances," http://www.bernabe.dorronsoro.es/vrp/.

[22] P. Hansen and N. Mladenović, "Variable neighborhood search: principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.

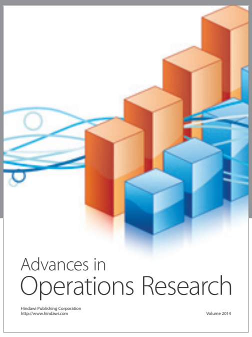
[23] J. Bae, *Algorithms for multiple vehicle routing problems [Ph.D. thesis]*, Texas A & M University, 2014.

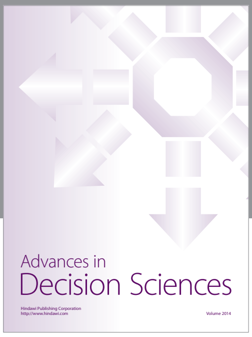
[24] K. Sundar and S. Rathinam, "Route planning algorithms for unmanned aerial vehicles with refueling constraints," in *Proceedings of the American Control Conference (ACC '12)*, pp. 3266–3271, 2012.

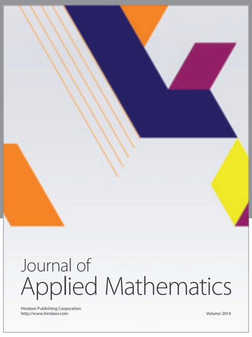
[25] S. Lin, "Computer solutions of the traveling salesman problem," *The Bell System Technical Journal*, vol. 44, pp. 2245–2269, 1965.

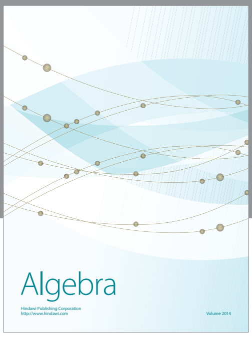
[26] M. Savelsbergh, "The vehicle routing problem with time windows: minimizing route duration," *ORSA Journal on Computing*, vol. 4, no. 2, pp. 146–154, 1992.

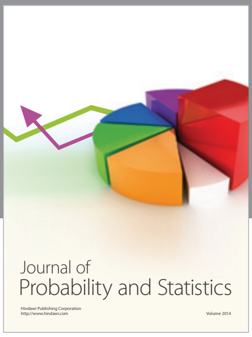
[27] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.


Advances in
Operations Research

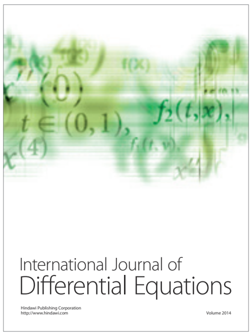

Advances in
Decision Sciences


Journal of
Applied Mathematics

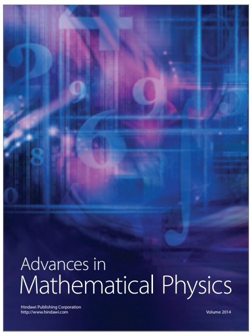

Algebra

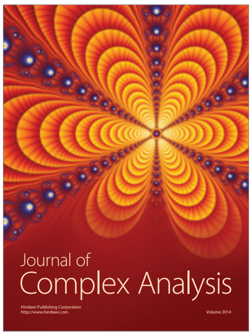

Journal of
Probability and Statistics

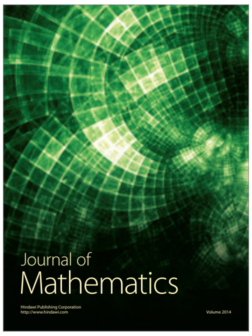
The Scientific
World Journal

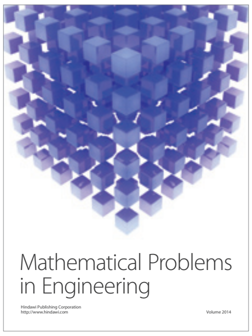

International Journal of
Differential Equations

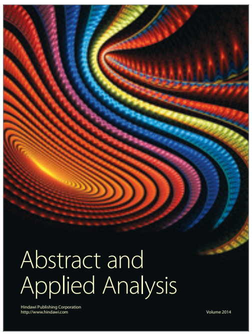
International Journal of
Combinatorics

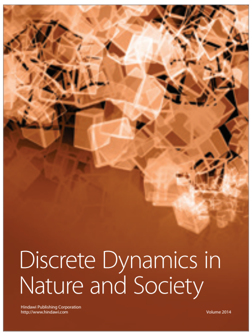

Advances in
Mathematical Physics

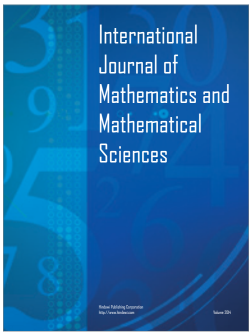

Journal of
Complex Analysis


Journal of
Mathematics


Mathematical Problems
in Engineering

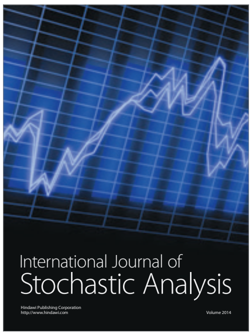

Abstract and
Applied Analysis


Discrete Dynamics in
Nature and Society


International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces


International Journal of
Stochastic Analysis

Journal of
Optimization