Journal of Naval Science and Engineering
2016, Vol. 12, No.1, pp.19-32
Endüstri Mühendisliği/Industrial Engineering

A RICH MIN-MAX VEHICLE ROUTING PROBLEM

Ertan YAKICI

Industrial Engineering Department,
Turkish Naval Academy, Naval Sciences and Engineering Institute, Tuzla, Istanbul
eyakici@dho.edu.tr

Date of Receive: 14.01.2016 **Date of Acceptance: 28.03.2016**

ABSTRACT

We present a new variant of vehicle routing problem with a min-max objective function. The problem has different types of service demands satisfied by a heterogeneous fleet of vehicles. Unlimited service capacitated vehicles serve the demand points with multiple time windows and requirement of operation synchronization when demand is split between vehicles. A mixed integer linear programming based heuristic solution approach is proposed and a numerical study is carried out to assess the performance of the proposed method.

ÖZ

Bu çalışmada min-max amaç fonksiyonlu yeni bir çeşit araç rotalama problemi sunulmaktadır. Problemde farklı tipte araçlardan oluşan bir filo ile farklı türde hizmet talepleri karşılanmaktadır. Hizmet kapasitesi sınırsız olan araçlar, birden fazla zaman pencereli ve talebin araçlar arasında bölünerek karşılanması halinde operasyon senkronizasyonu gerektiren talep noktalarına hizmet sağlamaktadır. Problemin çözümü için tamsayılı doğrusal programlama tabanlı bir sezgisel algoritma yaklaşımı önerilmiş ve önerilen metodun persformansını değerlendirmek için sayısal bir çalışma yapılmıştır.

Keywords: *Vehicle Routing Problem; Heuristic Method; Synchronization.*
Anahtar Kelimeler: *Araç Rotalama Problemi; Sezgisel Yöntemler; Senkronizasyon.*

1. INTRODUCTION

Since the first introduction of Vehicle Routing Problem (VRP) by Dantzig and Ramser [1], a huge literature is created on VRP and its several extensions. Like different constraint environments, objective environment also may have different focuses like minimization of the maximum or average arrival time. Those objective functions is often seen in the nonprofit environments like military operations, disaster relief and humanitarian

logistics (Duran et al. [2], Yakıcı and Karasakal [3], Renkli and Duran [4]). Naval operations like logistic support of warfare fleets, mine sweeping or hunting operations is the basic motivation of this research. Type of logistic support or operation, platform eligibility and other characteristics like operation speed (or economic speed), desired support or operation periods comprise several restrictions on the problem.

In the research problem presented in this work, there is a fleet that consists of vehicles having varying operational capabilities. Some vehicles are capable of conducting one of the service types, while others can operate in multiple types of service. Similarly, vehicles have different operation and transfer speeds. It is assumed that, service demands can be satisfied by one or more vehicles, demand points have multiple time windows denoting allowed periods for vehicles to arrive and all of the service demanded by each demand location should be satisfied by vehicle/vehicles arriving at the same time window. To the best of our knowledge, although there are very close problems, our problem is never introduced before in the literature.

A closely related problem is introduced in the work of Yakıcı and Karasakal [3] where various type of customer demands are satisfied by a heterogeneous fleet and split delivery is allowed. However, there are important differences between this problem and our research problem. In the problem presented in Yakıcı and Karasakal [3], the demand points are grouped in regions and the vehicles are not allowed to change their regions once they are assigned, while there is no such restriction in our problem. The other difference which makes our problem more generalized and complicated is the existence of realistic operational limitations, namely multiple time windows and operation synchronization, which are not handled in Yakıcı and Karasakal [3].

Temporal constraints are encountered in rich VRP problems frequently. However, so far, the interdependency of vehicles attracted less attention. The interdependency between the vehicles in operation synchronization is tedious to handle, especially in developing heuristic approaches. Several authors try to overcome these issue by allowing

infeasible solutions, evaluating the cost approximately and using indirect search (Drexl [5]).

The closest solution technique to the one we adopt is presented in the research article of Bredström and Rönnqvist [6]. They introduce a combined VRP and scheduling problem with time windows and temporal constraints and propose an optimization based heuristic to solve real instances. In order to have a small branch and bound tree they restrict the problem significantly in number of variables and constraints, then they try to improve the best known feasible solution.

The heuristic approach we propose is based on the idea of sequential insertion of vertices and route construction by restricted model solved with a standard ILP solver.

In the following sections, the problem is introduced with its mathematical formulation, the proposed heuristic approach is described and the results of numerical experiments are reported. Finally, conclusion is given in the last section.

2. PROBLEM DEFINITION

In this section, we present the notation and the definition of our problem. The total of transfer and service time of a vehicle leaving central vertex and visiting a given sequence of demand points is referred as “travel time”. Note that since the operation is considered to finish with the satisfaction of all service demands, time elapsed in returning to central vertex is not included in the travel time. Therefore, distance from any vertex to central vertex is considered to be zero.

The following sets and parameters are used: $K = (1,...,|K|)$: set of uncapacitated vehicles. $G = (V,E)$: complete directed graph with set of vertices $V = (1,...,n)$ and set of arcs E consisting of arcs (i,j) where i and $j \in V$. $P = (p_{ij})$: distance matrix between all pairs of vertices in V. $S = (s_k)$: vector of transfer speeds of vehicles in K. $Q = (q_i)$: vector of service

requirements demanded by vertices in *V*. *D* = (1,...,|*D*|): set of service types. *R* = (r_{kd}): matrix of service rates (delivered service in unit time) of the vehicles in *K* for type of demands in *D*. K_d: set of uncapacitated vehicles which can serve *d* type of service. V_d: set of vertices which demand *d* type of service. *Wi*: set of time windows for vertex *i*. u_{iw} : Latest time of time window *w* for any vehicle to arrive at vertex *i*. l_{iw} : Earliest time of time window *w* for any vehicle to arrive at vertex *i*. *M*: a big number. ε: a small number.

In order to formulate the problem as an Mixed Integer Program (MIP), let $\mathbb{R}$ and $\mathcal{B}$ denote the set of real numbers and {0, 1}, respectively. The decision variables are described as follows: $x_{kij} \in \mathcal{B}$ takes value 1 if edge (*i*,*j*) belongs to the k^{th} route; 0 otherwise. $y_{kij} \in \mathbb{R}$ denotes the maximum travel time. $T_k \in \mathbb{R}$ denotes the service time of vehicle k at vertex *i*. $A_{ki} \in \mathbb{R}$ denotes the arrival time of vehicle k at vertex *i*. $tw_{kiw} \in \mathcal{B}$ takes value 1 if vehicle k arrives at vertex *i* at time window *w*. $twn_{iw} \in \mathcal{B}$ takes value 1 if vertex *i* is visited at time window *w*. Only one time window is allowed for each vertex in the context of operation synchronization. $I_{ki} \in \mathbb{R}$ is a dummy variable used for subtour elimination.

$$\min y \tag{1}$$

$$A_{ki} + T_{ki} \leq y \qquad \forall k \in K \tag{2}$$

$$\sum_{k \in K_d} \frac{r_{kd}}{q_i} T_{ki} \geq 1 \qquad \forall d \in D, \forall i \in V_d \tag{3}$$

$$\sum_{i \in V} x_{k0i} \leq 1 \qquad \forall k \in K \tag{4}$$

$$\sum_{i \in V} x_{kij} = \sum_{i \in V} x_{kji} \qquad \forall k \in K, \forall j \in V \tag{5}$$

$$I_{ki} - I_k + (M+1)x_{kij} \leq M \qquad \forall k \in K, \forall i \in V \setminus (0), \forall j \in V \setminus (0) \tag{6}$$

$$\sum_{i \in V} \frac{q_i}{r_{kd}} x_{kij} \geq T_{kj} \qquad \forall d \in D, \forall k \in K_d, \forall j \in V_d \qquad (7)$$

$$A_{ki} \leq \left(1 - \sum_{i \in V} x_{kji}\right) M + u_{iw} tw_{kiw} + (1 - tw_{kiw}) M \qquad \forall k \in K, \forall i \in V \setminus (0), \forall j \in V \setminus (0), \forall w \in W_i \qquad (8)$$

$$A_{ki} \geq \left(\sum_{i \in V} x_{kji} - 1\right) M + l_{iw} tw_{kiw} + (tw_{kiw} - 1) M \qquad \forall k \in K, \forall i \in V \setminus (0), \forall j \in V \setminus (0), \forall w \in W_i \qquad (9)$$

$$A_{kj} \geq A_{ki} + \frac{p_{ij}}{s_k} x_{kij} + \left(x_{kij} - 1\right) M + T_{ki} \qquad \forall d \in D, \forall k \in K_d, \forall j \in V_d, \forall i \in V \qquad (10)$$

$$\sum_{w \in W_i} tw_{kiw} \geq \sum_{j \in V} x_{kji} \qquad \forall k \in K, \forall i \in V \setminus (0) \qquad (11)$$

$$twn_{iw} M \geq \sum_{v \in K} tw_{kiw} \qquad \forall i \in V \setminus (0), \forall w \in W_i \qquad (12)$$

$$\sum_{w \in W_i} twn_{iw} \leq 1 \qquad \forall i \in V \setminus (0) \qquad (13)$$

The objective function (1) minimizes the maximum travel time. Constraint (2) enforces the objective function value to be equal or greater than the largest travel time. Constraint set (3) ensures that all of the demands are satisfied. Constraints (4-6) ensure that each vehicle is assigned to at most only one route and each route starts and ends at central vertex without multiple visits to vertices. Constraint (7) connects the decision variables related to routing and servicing by ensuring that the vehicles should visit the vertices where they serve. Constraints (8) and (9) enforce the arrivals to occur within time windows. Constraint (10) enforce each arrival satisfies the elapsed time by previous travel. Constraint (11) connects the decision variables related to routing and time windows. Constraint (12) and (13) serve to ensure the operation synchronization by enforcing each vertex can be visited by the vehicles in only one of its time windows.

Note that still we can reduce the size of feasible space by demand type cuts (14) which prevents the vehicles from visiting the demand points that they cannot serve.

$$\sum_{k\in(K\setminus K_d),i\in V} x_{kij} = 0 \qquad \forall d \in D, \forall j \in V_d \qquad (14)$$

3. HEURISTIC SOLUTION APPROACH

The heuristic approach is based on the idea of restriction of the MIP model. Instead of solving the problem as a whole, we solve the problem in steps. At each step, a group of vertices are added to the model until all of the vertices are covered. The restricted problem is given a certain period of time for each step. The traversed arcs of the best solution found is carried to the next step. Each vehicle can break at most a limited number of arcs from its own route where some (or all) of the carried arcs are allowed to be broken in order to visit the vertices added. Another restriction is applied once in a while after a certain number of steps are taken in between. This restriction fixes all of the previously traversed arcs and service that are already processed in the visited vertices. Therefore, the problem restarts again with the remaining vertices. We call these steps "fixing steps". After a fixing step, a vehicle starts from the vertex it visits just before returning to the central vertex and from the time it finishes its service in that last visited vertex. In order to apply the restrictions, the following constraints are added to the model.

$$x_{kij} \geq f_{kij} - e_{kij} \qquad \forall k \in K, \forall i,j \in V \qquad (15)$$

$$\sum_{i,j\in V} e_{kij} \leq 1 \qquad \forall k \in K \qquad (16)$$

$$e_{kij} \leq a_{kij} \qquad \forall k \in K, \forall i,j \in V \qquad (17)$$

$$A_{k0} \leq A_{k\ level} \qquad \forall k \in K \qquad (18)$$

The binary parameters f_{kij}, a_{kij} and nonnegative parameter $A_{k\ level}$ respectively denote whether the arc is traversed at the previous step (= 1),

whether it is allowed to be broken (= 1) at the current step and the time when vehicle k completes its job just before returning to central vertex in the last fixing step. The binary decision variable e_{kij} takes positive value if the arc (i,j) is broken by vehicle *k*. The last constraint (18) forces vehicle *k* to start at the time given by $A_{k\ level}$ after last fixing step. Note that, distance parameter in the model has to take an additional index indicating the vehicle it belongs to, and distances from new starting vertices of vehicles to all of the vertices not covered yet, p_{k0j}, have to be recalculated at fixing steps.

We define two criteria to choose vertices to be added. Given the worst scenario, the vehicles do not break any arc and continue from the latest visited vertex just before returning to the central vertex, one of them is the minimum required time to begin to serve newly added vertex, $min_{RT}(i,k)$, and the other is the maximum time that can be spent before making an attempt (leave the current vertex) to visit newly added vertex, $max_{TS}(i,k)$, where $i \in V$, $k \in K_i$ and K_i is the set of eligible vehicle set for vertex *i*. These values are affected by both of the time windows of the vertex to be added and the time required by vehicle to reach to that vertex. For a vertex to take priority in addition to the model, it is better to have less value from both of these criteria. Note that each of the criteria can be defined for each of the combinations made by a vertex and a vehicle that are eligible to visit that vertex. Therefore, while considering all of those combinations, it is also useful to consider the combinations that give the extreme values: the minimum value coming from the first criterion and the maximum value coming from the second one, namely, $\min_{k \in K_i} minRT(i,k)$ and $\max_{k \in K_i} maxTS(i,k)$. We define a simple function that determines the relative priorities of the vertices to be added to the model. There are four components in the function $f(i,K_i)$: the summation of values for all eligible vehicles (for first and second criteria) and the extreme values which are defined in the previous paragraph (for first and second criteria). Those four values for each vertex are separately scaled between 0 and 1. Therefore, each vertex has four values between 0 and 1. The summation of those four values, equally or differently weighted, defines the priority of vertex.

In the selection of the arcs that are allowed to be broken, the number of consecutive steps that an arc remain in the solutions is taken as the basic parameter. If this parameter is set to "0", all traversed arcs are allowed to be broken in the next step. As the parameter increases, the number of arcs allowed to be broken decreases. In all cases, the last arc of all vehicles which is returning to central vertex is kept breakable. We accept that it is hard to give a specific comment on the best combination for the remaining parameters which are the number of added vertices at each step, the number of steps between fixing steps and the time allowed for solving the restricted problem. These parameters can be determined with respect to the problem size and the time given to have a feasible solution.

Since we solve the problem in a stepwise manner and establish a priority value for the vertices for being added to the model, it is possible to have infeasibility in subproblems because of time window restrictions of the vertices. In order to handle this case, we add penalty component to the objective function. In order the penalty mechanism to work, a new decision variable is introduced into the model. The objective function (1) and the constraint (3) is changed as indicated in the expressions (19) and (20).

$$\min\left(y + \sum_{i \in (V \setminus 0)} penalty_i M\right) \tag{19}$$

$$\sum_{k \in K_d} \frac{r_{kd}}{q_i} T_{ki} \geq 1 - penalty_i \qquad \forall d \in D, \forall i \in V_d \tag{20}$$

The new binary decision variable $penalty_i$ takes positive value if adding vertex i is infeasible or it takes much time than allowed to find a solution including that vertex at the current step. If there is a penalty, the algorithm turns back to the last fixing step to repair the solution to have it become feasible. In repairing, fixed solution is released in order to have the penalized vertices be introduced to the model, where all traversed arcs are kept fixed but allowing that limited number of arcs from each route can be broken just as it is applied in constructing steps. The formal ***algorithm of construction heuristic*** is given below. Since repairing procedure is

straightforward, we do not denote repairing steps separately in the algorithm.

• Step 1. Set heuristic parameters and construction model (objective function 19 subject to constraints 2, 4-13, 15-18, 20), calculate vertex priority values and create an ordered list of vertices L w.r.t. addition-to-model priority. Go to Step 2.

• Step 2. If L is not empty, add first n_{step} vertices (or add all vertices if there are less than n_{step}) from L to model and update L, excluding the vertices added to model, go to Step 3. Otherwise stop.

• Step 3. Solve model in given time period,

– Step 3.1. If this is not a fixing step and all vertices are served without penalty, set the values f_{kij} to 1 for all traversed arcs $(i,j) \in E$ and $k \in K$ where x_{kij} is equal to 1. Set a_{kij} to 1 w.r.t. given setting in order to determine the arcs that can be broken, go to Step 2.

– Step 3.2. If this is a fixing step and all vertices are served without penalty, go to Step 4.

– Step 3.3. If all vertices cannot be served, turn back to the last fixing step and apply repair procedure. If solution cannot be repaired go to previous fixing steps to apply repair procedure until the penalized vertices are added to the model. Update L, go to Step 2.

• Step 4. Exclude all added vertices from model. For all excluded vertices; keep the traversed arcs, served time windows and serving duration in the vertices. Update the distances from central vertex to other vertices: For all vehicles $k \in K$, set p_{k0j} to p_{kcj} where c indicates the last visited vertex of k. Set the current vertices to "0", and $A_{k\,level}$ to $A_{kc} + T_{kc}$ for all vehicles $k \in K$. Go to Step 2.

The solution constructed as it is explained above may have still room to be improved. One way is improvement within each route and the other is combining more than one routes to have an opportunity to improve. Before combining the routes, we choose to check if there is an improvement room within each route applying a simplified version of the model with considered vehicle and vertices it has served. Then, we expand this check by adding two routes, one from the set of longest routes and one from the set of shortest routes, to the model. In both improvement phases, the traversed arcs are all allowed to be broken, which means the exclusion of the constraints (15-17) from the model. Also allowed time window of each vertex that are served by other vehicles (the vehicles belong to the routes that are not considered at the current improvement step) are fixed by adding constraint (21) into the improvement model.

$$twn_{iw} \geq twn_{iw\ level} \qquad \forall i \in V \setminus (0), \forall w \in W_i \qquad (21)$$

The binary parameter $twn_{iw\ level}$ is set to positive value, if the vertex i is served in time window w by any other vehicle that is not considered at the current improvement step. While choosing the longest and shortest route, the combination, giving the greatest number of vertices in longest route that can be served by the vehicle of shortest route, is given priority. While the set of longest routes consist of only the routes that have the maximum time length, the set of shortest routes are populated starting from the route(s) having shortest time length and adding one by one until the set satisfies that the vehicles included are eligible to serve all vertices served by the set of longest routes. To complete the improvement phase within reasonable time period, each improvement iteration is given a certain period of time and after allowing a certain number of steps (or time), when there is a deterioration or no-improvement in the objective function, the algorithm is stopped. The steps of the procedure applied for the ***improvement phase of the heuristic approach*** is given below.

• Step 1. Set improvement model (objective function 1 subject to constraints 2-13, 21). Solve separately in given time periods for each vehicle $k \in K$. Update the solution. Go to Step 2.

• Step 2. Update the set of longest routes R_l and set of shortest routes R_s such that R_l covers all r_l longest routes having equal values, and R_s covers the shortest r_s routes of which vehicles can serve all of the vertices covered in R_l. Go to Step 3.

• Step 3. If the stopping criterion does not hold, solve improvement model with the combination of one route each from R_l and R_s, such that the number of vertices in the longest route that can be served by the vehicle of the short route be maximized (if there is a tie, choose the combination with shorter route, if the tie is not broken, break randomly). Update the solution, and go to Step 2. Otherwise stop.

4. EXPERIMENTS

Since there is no test instances that can serve as benchmark either as a lower bound or optimal solution to our problem, we choose our test problems from the test problems given in Yakıcı and Karasakal [3]. We also applied the same vehicle parameters as they are applied in Yakıcı and Karasakal [3]. While keeping all of the data of test problems given in Yakıcı and Karasakal [3] as they are, since it is required in our problem, we add parameters for two time windows for each vertex. Three different time window pairs, (0-15, 20-35; 5-20, 25-40; 10-25, 30-45), are determined and one of these pairs is assigned to each one of vertices except central vertex. In time window assignment to vertices, we follow the order of given numbers to vertices and the order of time windows given in the previous sentence.

Performance of the proposed method is compared to exact solution method provided by CPLEX 12.6.2.0 ILP solver. In addition, since even the small instances cannot be solved by exact method, in order to have another benchmark, we relaxed the time windows such that the first time window of each node has no upper bound. The lower bounds and best solutions achieved in twelve hours are reported along with the heuristic methods performance. In application of heuristic method, at most two minutes (and 10% relative gap criterion, if it is reached earlier) is given for solving the

restricted MIP at each step and maximum ten minutes is given in improvement phase. Except 31-node instance, all instances used up ten-minute improvement period. Improvement phase is only applied after the completion of construction and stepwise approach is not used in this phase. In the construction phase, five nodes are added to the model at each step using equal weight priority function values and at each step only one arc from each route is allowed to be broken. When not all arcs, but only older arcs (the arcs which is traversed in the same route consequently at more than one in the last steps) are defined as breakable, the last arc returning to central vertex is always allowed to be broken.

Experiments are conducted with a personal computer with 2.6 GHz CPU and 8 GB RAM. The result of the experiments is reported in Table 1, where instance identity (ID) is given in the first column. In instance ID, the first section before the letter “k” denotes the test problem group and number of vertices as denoted in Yakıcı and Karasakal [3]. Vehicle quantities from each type are indicated after the letter “k”. Next four columns are related to the solutions of MIP and relaxed MIP. MIP refers to the model defined by objective and constraints (1-13), while Relaxed MIP refers to the same model where only first time windows are applied without upper bound. For MIP and Relaxed MIP, the lower bound and the incumbent solution from exact solution method is reported after twelve-hour period is elapsed. In the columns under the “Heuristic” title, solution for the problem with original time windows (not relaxed), time elapsed in the construction phase of solution process (in minutes and seconds) and parameter settings are given. In the “Settings” column, first field (before slash) denotes the number of steps that an arc should remain in the solution to be breakable and second field denotes the required number of vertices to be covered to apply fixing steps. As presented in this column, we start fixing when the step with 15 vertices is solved. Then, at every increment of 10 more vertices in the problem, fixing step is applied.

Table 1. Experiment Results

	MIP		Relaxed MIP		Heuristic		
Instance ID	LB	Sol'n	LB	Sol'n	Sol'n	Time	Setting
E031-k1/1/2/4	2.33	-	6.49	31.53	35.28	7'23"	0/15-25
E031-k1/1/2/4	2.33	-	6.49	31.53	36.53	6'57"	1/15-25
P-n51-k1/1/2/4	2.29	-	16.99	-	54.28	10'30"	0/15-25-…-45
P-n51-k1/1/2/4	2.29	-	16.99	-	58.10	10'22"	1/15-25-…-45
E-n76-k1/1/2/4	2.18	-	13.74	-	89.05	15'04"	0/15-25-…-65
E-n76-k1/1/2/4	2.18	-	13.74	-	91.04	14'15"	1/15-25-…-65

Observing Table 1 yields that we have rather poor lower bounds and no solutions at all from commercial solver. When the problem is relaxed and solved by the same solver, we can find better lower bounds which serves as lower bounds to the original problem (MIP). Since the solver does not perform well in finding solutions, we believe these lower bounds are still far away from optimal solutions. Therefore, while we cannot determine the quality of heuristic solution, we can claim that the heuristic is useful in this case when there is no other algorithm that can find better solutions.

5. CONCLUSION

We introduce a rich VRP in which there is a heterogeneous fleet having varying operational capabilities and it is assumed that, service demand of a demand point can be satisfied by one or more vehicles in one of assigned multiple time windows.

An ILP formulation of the problem is given and an ILP based heuristic is developed for solving the problem. We have employed and tailored three small test problems from an earlier study [3] about a similar problem having no time window and synchronization restrictions. Even with these small instances, the commercial solver cannot find any feasible solution in twelve hours while heuristic solution is able to find solutions in less than half of an hour for the largest instance. Also lower bounds found by the exact method solver in twelve hours are very poor. Therefore, we do

not have an idea about the solution quality, however we consider that the proposed method is useful in the absence of a better solution.

REFERENCES

[1] Dantzig G.B., and Ramser J,H., (1959). *The truck dispatching problem,* Management Science, vol. 6(1), pp. 80–91.
[2] Duran S., Gutierrez M.A., and Keskinocak P., (2011). *Pre-positioning of emergency items for care international*, Interfaces, vol. 41(3), pp. 223–237.
[3] Yakıcı E., and Karasakal, O., (2013). *A minmax vehicle routing problem with split delivery and heterogeneous demand,* Optimization Letters, vol. 7(7), pp. 1611–1625.
[4] Renkli Ç., and Duran S., (2015). *Pre-positioning disaster response facilities and relief items,* Human and Ecological Risk Assessment: An International Journal, vol. 21(5), pp. 1169–1185,.
[5] Drexl M., (2012). *Synchronization in vehicle routing - a survey of VRPs with multiple synchronization constraints,* Transportation Science, vol. 46(3), pp. 297–316.
[6] Bredström D., and Rönnqvist M., (2008). *Combined vehicle routing and scheduling with temporal precedence and synchronization constraints,* European Journal of Operational Research, vol. 191(1), pp. 19–31.