Uncapacitated Vehicle Routing with Time Windows

Abstract

We consider the vehicle routing problem when each customer has a time window during which they wish to take delivery of the goods. The vehicles start from a depot and the time for travel between any pair of customer locations, the depot is given. Our objective is to find the minimum number of vehicles required to serve all customers and we assume that there are no capacity restrictions on the vehicles.

In this paper we develop a new linear programming formulation for this problem and evaluate it experimentally. We find that the optimum solution provided by this linear program is within an additive 1 of the optimum solution for most instances of the Solomon set. We also implement an iterative rounding procedure using this strengthened LP formulation and obtain solutions which are either optimum or within an additive 1.

1 Introduction

The problem of routing vehicles to deliver goods to a given set of customers is well-studied in the Operations Research and the Algorithm communities. In its simplest version we are given a depot from where the vehicle(s) starts and ends, location of customers and the time for travel between the customer locations and the depot. The problem is to route one or more vehicles to service all customers. If the objective was to minimize the distance travelled, then this is the classical travelling salesman problem.

In this paper, we consider the version when each customer provides time-windows in which they are willing to take delivery of goods. Our objective is to minimize the number of vehicles needed to service every customer within their time-windows.

One of the main results of the paper is a new Integer Programming formulation of this problem which is obtained by suitably discretizing time and associating a vertex with every customer, time-index pair where time-index is a value within the customers time-window. By introducing suitable edges in this graph we capture the problem of routing vehicles as one of routing a flow subject to certain “coverage requirements”. The linear relaxation of this Integer program has a large integrality gap and we introduce additional inequalities. The strengthened LP is the second major contribution of this paper.

Our algorithm for finding the minimum number of vehicles uses iterative rounding on this strengthened LP solution by finding longest paths in the support of the LP solution. Our third major contribution is an experimental evaluation of this approach. We test our algorithm on the Solomon benchmark which is widely used for evaluating various approaches to vehicle routing. While we do not see a significant improvement in the quality of the solution obtained by the strengthened linear program over the optimum solution to the linear program, we observe that our approach for iterative rounding give optimum solutions for over 60% of the instances in the Solomon data-set. On instances where we do not hit the optimum, our solution is only 1 vehicle away, except for 1 instance when the number of vehicles is 2 more than the optimum.

Besides being a promising approach to obtain near optimum solutions for the `Vehicle routing with time windows` problem, our framework is robust enough to extend to generalizations of this problem. In the capacitated version of this problem customers have demands and the vehicles have bounded capacity and we discuss how to adapt our approach for this setting. We also discuss how other extensions like heterogeneous vehicles with different capacities, multiple depots etc can be formulated and solved using our approach.

2 Related Work

In this section of the paper we look at prior work on the `Vehicle routing with time windows` problem and focus primarily on the capacitated and uncapacitated variants of this problem.

[8] published in 1954 is one of the earliest articles on vehicle routing problems and exhaustively examined the traveling salesman problem (TSP). There is an extensive body of work on the TSP which can be viewed as a special case of the Vehicle routing problem (VRP) . The well-known ”Truck Dispatching Problem,”[9], was one of the key catalysts for the development of VRP research in 1959. The problem concerned the

designing of routes for a group of gasoline delivery vehicles between a bulk terminal and a number of service outlets supplied by the terminal. The distances between all pairs of points were given along with the demand for the products each service station ordered. The Vehicle routing problem is frequently encountered in the logistics and transportation sectors, in figuring out the best way to attend or deliver to customers, geographically distributed in the areas surrounding a centrally located depot, with the usage of trucks. The precise phrase of "vehicle routing" was introduced by [13] who also included it in the title of the paper. The early 1970s saw the emergence of various other cases of VRP, like fleet routing [15] and transportation network design [17]. [20] introduced time window constraints to traditional VRP problems and gave the well-known "Solomon Instances". In the 1990s, VRP research picked up speed and researchers and heuristics were first incorporated into research studies for routing in VRPs around this time [10].

Letchford and Eglese [14] considered an arc routing problem for a single vehicle with time windows where the arcs are divided into deadline classes. They used an integer programming formulation to solve the problem and provided an LP relaxation that yielded algorithms for identifying situations where inequality constraints were not satisfied. The paper also included results for various instances of number of customers. [21] considered the problem of routing workers in a field which was cast as a multi depot, uncapacitated vehicle routing problem. They proposed a formulation using the set covering technique and an exact solution for smaller instances. For larger instances they developed several heuristic solutions.

For the capacitated variant of VRP with time windows, many exact as well as heuristic algorithms, have been proposed. An exact algorithm, the branch cut algorithm for capacitated vehicle routing was given by [1] who proposed a formulation and inequalities to strengthened it, [1] was the first paper to successfully solve the capacitated VRP problem for 135 customers using a branch-and-cut exact algorithm. [18] proposed a branch and cut algorithm which is based on a 2 index approach. [16] put forward another exact approach and their algorithm could solve many instances that [1] was unable to solve. [3] advanced [11]'s Travelling Salesman Problem (TSP) model by introducing a 2 commodity flow technique. An exact algorithm based on this enhanced model was also detailed in [3]. The Set Partitioning Algorithm, which is yet another exact method, for capacitated vehicle routing was originally proposed by [6]. [12] proposed an exact set partitioning algorithm in which variables comprise of a collection of routes introduced by [7] and constraints are the partitioning constraints of sets. [2] tackled this problem using the set partitioning method but proposed new types of inequalities [5]. Furthermore, [2] extended this work by introducing more relaxations in [4], to solve Capacitated VRP and VRP with time windows.

A number of population-based heuristic techniques have been employed to solve the Capacitated Vehicle routing with Time-windows problem (CVRPTW). [23] gave a combination of Tabu Search (TS) and Ant Colony Optimization (ACO). For small-scale VRPTW problems, the Localized Genetic Algorithm offered superior solutions over other heuristic methods [22]. ABC-T was suggested by [19] as a solution to CVRPTW. It was an improvement over ABC (Artificial Bee Colony) algorithm with tournament selection to improve global search. The R102 instances of Solomon's data sets were used to test the hypothesized ABC-T method, and the results were compared with those obtained by earlier algorithms.

3 Formulating an LP

Let $C = \{c_i, i \in [n]\}$ be a set of customer locations and s a depot. Let $d : C \cup \{s\} \times C \cup \{s\} \to \mathbb{R}^+$ be a metric which represents the time required to travel between two points in $C \cup \{s\}$.

Let $[s_i, t_i]$ be the time window for customer $c_i \in C$. Our integer program is time-indexed and so we discretize the time windows to formulate it. Let Δ be a discretization factor; for $i \in [n]$ we replace s_i by $\lceil s_i/\Delta \rceil$ and t_i by $\lfloor t_i/\Delta \rfloor$.

3.1 Constructing a graph

To formulate the linear program, we construct a graph $G = (V, E)$ which has a vertex s for the depot and vertices $(c_i, t), s_i \leq t \leq t_i, i \in [n]$. Let $V_i = \{(c_i, t), s_i \leq t \leq t_i\}$. Then $V = \cup_i V_i \cup \{s\}$. Thus the number of vertices in G is one more than the total size of the (discretized) time windows.

We include an edge $((c_i, t), (c_j, t'))$ in E if the vehicle can start from customer location c_i at time $t\Delta$ and reach customer location c_j by time $t'\Delta$. Thus $t'\Delta \geq t\Delta + d(c_i, c_j)$. We assume the vehicle is at s at time 0 and for $i \in [n]$ include edges $(s, (c_i, t))$ if $d(s, c_i) \leq t\Delta$. We also include an edge from every vertex to s.

3.2 Reducing Edges

To reduce the size of the linear program we reduce the number of edges in G as follows.

1. Introduce an edge $((c_i, t), (c_i, t+1)), s_i \leq t < t_i, i \in [n]$. A vehicle traversing such an edge can be viewed as staying at customer location i for Δ time units. (Here t+1 is the next discrete time after t)

2. For $i, j \in [n]$ and $t \in [s_i, t_i]$, we replace the set of edges $\{((c_i, t), (c_j, t')), t' \geq t + d(c_i, c_j)/\Delta\}$ with the edge $((c_i, t), (c_j, t')), t' = \max(s_j, t + \lceil d(c_i, c_j)/\Delta \rceil)$. This edge together with the edges introduced in the previous step for c_i can fulfil the role of the edges in the set replaced.

3. For $i \in [n]$ we replace the set of edges $\{(s, (c_i, t)), t \geq d(s, c_i)/\Delta\}$ with the edge $(s, (c_i, t))$ where $t = \max(s_i, \lceil d(s, c_i)/\Delta \rceil \leq t_i)$.

4. For $i \in [n]$ we replace the set of edges $\{((c_i, t), s), s_i \leq t \leq t_i\}$ with the edge $(c_i, t_i), s)$.

3.3 Integer Program Let x_e, $e = ((c_i, t), (c_j, t'))$ be an indicator variable which is 1 if the vehicle starts from c_i at time $t\Delta$ and reaches c_j at or before time $t'\Delta$. For $e = ((c_i, t), (c_i, t + 1))$, $x_e = 1$ corresponds to the truck waiting from time $t\Delta$ to $(t + 1)\Delta$ at location c_i. Since the truck is at s at time 0, $x_e = 1$ for $e = (s, (c_i, t))$, corresponds to truck arriving at c_i from s by time $t\Delta$ while $x_e = 1$ for $e = ((c_i, t), s)$, corresponds to truck returning to s from c_i.

If a truck arrives (or has been waiting) at c_i at time $t\Delta$ then it should also leave (or continue waiting at) c_i at time $t\Delta$. Variable x_e can be viewed as a flow on the edge e and we have flow conservation at every node in V. Flow is also conserved at s since the number of trucks leaving s equals the number returning back. Thus,

$$(3.1) \qquad \forall v \in V, \quad \sum_u x_{(u,v)} = \sum_u x_{(v,u)}.$$

Since every customer must be serviced by a truck we have

$$(3.2) \qquad \forall i \in [n], \quad \sum_{v \in V_i} \sum_{u: u \notin V_i} x_{(u,v)} = 1.$$

Enforcing integrality constraints $x_e \in \{0, 1\}, e \in E$ gives an integer program which we denote by `VTIP`. The objective of this integer program is to minimize $\sum_{u:(s,u) \in E} x_{(s,u)}$(number of outgoing trucks from the source).

Lemma 3.1. *An optimum solution to* `VTIP` *gives the minimum number of trucks needed to service all customers within their time windows.*

Proof. Let x be an integer solution to `VTIP`. We first argue that x yields a solution to the `Vehicle routing with time windows` instance. Let $E' \subseteq E$ be the edges with $x_e = 1$. Constraint 3.1 ensures that E' is a set of cycles each of which contain s. Constraint 3.2 ensures that for all $i \in [n]$ the cycles include at least one vertex from V_i. A cycle gives

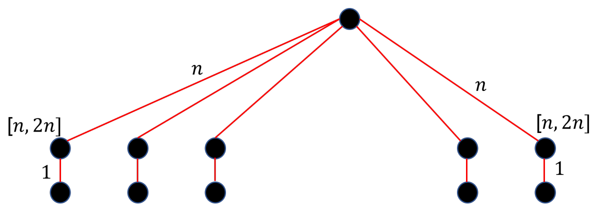

Figure 1: An instance which establishes an integrality gap of $n/2$.

a path followed by a truck and hence every customer is serviced by a truck in her time-window. Thus the number of trucks is the number of cycles which is the value of the solution x.

For the converse observe that a solution to `Vehicle routing with time windows` yields a set of cycles in G each of which includes the depot s. Assigning $x_e = 1$ to edges on these cycles and $x_e = 0$ to the remaining edges gives a solution to `VTIP` of value equal to the number of cycles which in turn equals the number of trucks.

This equivalence between solutions of `VTIP` and solution to the `Vehicle routing with time windows` problem implies that the optimum solution to the `VTIP` gives the minimum number of trucks needed to service all customers within their time-windows. □

3.4 The LP Relaxation and a Bad Example Replacing the integrality constraints in `VTIP` with the constraint $0 \leq x_e \leq 1, e \in E$, yields a linear program which we refer to as `VTLP`.

The Linear program `VTLP` has a large integrality gap; there exist instances where the optimum solution to `VTLP` is much smaller than the optimum solution to `VTIP`. One such instance is shown in Figure 3.4.

The n customer locations are distributed over $n/2$ pairs and each pair is n time-units away from the depot. It takes 1 unit of time to travel between the two locations in a pair and the time window for each customer is $[n, 2n]$. It is easy to see that a vehicle starting from the depot can service only one pair and hence we require $n/2$ vehicles to serve all customers.

On the other hand it is easy to construct a solution to `VTLP` of value 1. Let c_i, c_j be customer locations in a pair. For $\Delta = 1$ the graph G has vertices (c_i, t) and (c_j, t) for $n \leq t \leq 2n$ and edges $((c_i, t), (c_j, t + 1))$ and $((c_i, t), (c_j, t + 1))$ for $n \leq t < 2n$. G also has an edge $(s, (c_i, n))$ and an edge $((c_j, 2n), s)$ (see Figure 3.4). Thus for odd n, G has a cycle $P = s, (c_i, n), (c_j, n+1), (c_i, n+2), \ldots, (c_i, 2n-1), (c_j, 2n), s$. The cycle P visits $n/2$ vertices each from sets V_i, V_j. Thus, if we assign $x_e = 2/n$ to each edge on P, we satisfy Constraint 3.2 for customers i, j. Doing this for all pairs of customer locations yields a feasible solution to VTLP of value $n/2 \times 2/n = 1$.

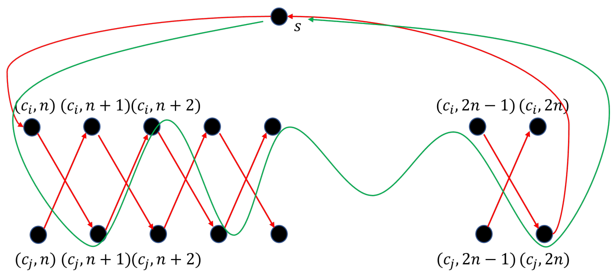
s
(c_i, n) $(c_i, n+1)$ $(c_i, n+2)$
$(c_i, 2n-1)$ $(c_i, 2n)$
(c_j, n) $(c_j, n+1)$ $(c_j, n+2)$
$(c_j, 2n-1)$ $(c_j, 2n)$

Figure 2: Part of the graph G for the instance in Figure 3.4.The cycle P is shown in green.

3.5 A strengthened LP

To address the integrality gap of VTLP we strengthen it with additional constraints. In particular, given an assignment of x_e values to edges $e \in E$ we would like to better capture the extent to which a customer i is serviced. Note that this is the "fractional connectivity" of the set V_i from depot s. In other words the extent to which customer i is serviced is the capacity of the minimum cut separating V_i from s where the capacity of a cut is the sum of the x_e values of the edges in the cut.

To obtain a succinct LP formulation, for all $i \in [n]$ we introduce a commodity i which corresponds to customer c_i. We modify G by introducing a sink r_i for commodity i and adding edges from every vertex in V_i to r_i. Let f_e^i be a variable denoting the flow of commodity i in edge e. Then

$$(3.3) \qquad \forall e \in E, i \in [n], \quad 0 \leq f_e^i \leq x_e.$$

The flow of commodity i is conserved at all nodes except s and r_i. Thus,

$$(3.4) \forall v \in V \setminus \{s, r_i\}, \quad \sum_{u \in V} f_{(u,v)}^i = \sum_{u \in V} f_{(v,u)}^i.$$

Finally the total flow of commodity i reaching r_i should be 1 and hence

$$(3.5) \qquad \forall i \in [n], \quad \sum_u f_{(u,r_i)}^i = 1.$$

The constraints 3.1,3.2, 3.3, 3.4, 3.5 together with the objective of minimizing $\sum_u x_{s,u}$ form our strengthened Linear program, SLP.

4 The Algorithm

Our algorithm begins by finding an optimum solution, x^* to the strengthened linear program, SLP. Let x_e^* be the value assigned to edge $e \in E$ and let $E' = \{e \in E, x_e^* > 0\}$ be the support of x^*. The flows of various commodities on edge e, $f_e^i, i \in [n]$ were used only to strengthen the LP and will have no further role to play; they are thus ignored.

Recall (see Constraint 3.1) that for every node $v \in V$, the sum of the x^* values on the incoming edges and the outgoing edges are equal. Our algorithm proceeds in iterations and in each iteration we,

1. find all cycles containing s which have $x_e^* = 1$ for every edge of the cycle,
2. find the longest cycle containing s in $G' = (V, E')$ which does not use any edge of a cycle found in Step 1, and
3. remove all customers on cycles found in Steps 1,2, and reformulate and re-solve SLP for this reduced instance.

The algorithm ends when all customers have been removed. The cycles picked in the various iterations form the routes for the vehicles and the number of vehicles needed is the number of cycles. Note that the number of iterations is at most the number of vehicles needed since in each iteration we pick at least one cycle.

Step 1 which picks cycles containing s all of whose edges are at 1 in x^* identifies the "integral part" of the solution x^*.

For Step 2, we note that finding the longest cycle in a graph is NP-hard. However, the graph $G = (V, E)$ is acyclic if we ignore the edges entering s. This is because every vertex, (c_i, t) has a time-index t (s can be viewed as having a time-index 0) and edges (other than those entering s) only go from vertices with a lower time to vertices with a higher time.

Since G' is a subgraph of G, it too is acyclic and we can find the longest path from s to any vertex in $V \setminus \{r_1, r_2, \ldots, r_n\}$ in G'. Such a path will end at a vertex (c_i, t_i) and by adding the edge $((c_i, t_i), s)$ to this path we obtain the longest cycle in G' containing s. Note that this cycle may contain more than one vertex of a set V_i and hence it only serves as a proxy for finding a route which maximizes the number of customers serviced.

Step 2 could have been implemented in other ways. We could have picked a "fattest cycle" - a cycle which maximises the minimum x_e^* for edges e on the cycle. Or we could have picked a cycle which is both "fat" and "long" and maximises the product of the length of the cycle and the minimum x_e^* for edges e on the cycle. Our initial experiments - which are not reported in this paper - suggested the longest cycle as the best choice amongst these three options.

5 Experimental Results

This section presents the quality of solutions obtained for the Integer Program (VTIP), the Linear Programming relaxation of the Integer Program (VTLP) and the strengthened linear program (SLP). The experiments were performed on Solomon's benchmarks that consist of six different data-sets. These data-sets are grouped in six instance classes: R1,C1,RC1,R2,C2,and RC2.

Each instance has 100 customers distributed in a 100 x 100 square. Each class is defined by how the customers are distributed within the square - R stands for randomly, C for clustered and RC contains both random and clustered. These are further classified based on the average width time window in comparison to the average travel time - 1 stands for narrow time windows and 2 for time windows that are wide. In our paper we have experimented on narrow time windows only. In Solomon's data-sets x-y coordinates for each customer location are provided and the time required to travel between location c_i and location c_j is given by $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

5.1 Our Experimental Setup Our algorithm is coded in python and uses Gurobi to solve the linear and integer programs. The code runs on a machine with Intel(R) Core i7-7500U CPU , 2.7 GHz/8 GB of RAM. The discretization factor Δ was set to 10 minutes(since a delay of 10 min is reasonable in real life). We ran our model for a few instances of the Solomon data-set and tabulated the results.

5.2 Results and Comparisons The tables 1, 2 and 3 show our results when the number of customer locations are 25, 50 and 100 respectively. The tables show a comparison between the optimum value of the Integer Program (VTIP), the optimum value of the linear relaxation to the integer program (VTLP), and the optimum value to the Strengthened Linear program (SLP). We report the improvement in the optimum value of SLP over the optimum value of VTLP and observe that for most instances the improvement is only very minor. We also report the number of vehicles used in the solution returned by our algorithm in the column labeled "Algorithm" and report the difference from the optimum solution of VTLP in the column labeled "Error". Finally, we report the number of iterations required by our algorithm. This measure informs us of the number of times the strengthened Linear Program is solved by our algorithm

As expected we observe that the optimum solution to VTLP is no more than the optimum solution to SLP which in turn is at most the optimum solution to VTIP. Interestingly, for many instances the optimum solution to the SLP equals the optimum solution of the Integer Program VTIP.

We ran our algorithm on 29 instances with 25 customers each. Of these 29 instances, our algorithm returned the optimum solution for 24, and for the remaining 5 instances the solution obtained by our algorithm was one more than the optimum. We ran our algorithm on 24 instances with 50 customers each. Of these 24 instances, our algorithm returned the optimum solution for 23, and for the remaining 11 instances the solution obtained by our algorithm was one more than the optimum. For instances having 100 customers, of the 9 instances on which we ran the algorithm, we obtained the optimum solution for 5. For 3 of thee instances our solution was 1 more than the optimum while for 1 instance (RC106) we were off by 2 vehicles. The code and paths obtained are available at `https://bit.ly/3A5u842`

6 Extensions: Vehicle Routing with Time Windows and Vehicle Capacities

Our approach to solve the `Vehicle routing with time windows` problem can be extended to the setting when the vehicles have capacities, say U, and each customer i has a demand $d_i \in [U]$. The graph we construct now has a vertex for every triple (c_i, t, u) where $i \in [n], t \in [s_i, t_i]$ and $u \in [U]$. We include an edge from (c_i, t, u) to (c_j, t', u') if $t' = \max(s_j, t + \lceil d(i,j)/\Delta \rceil)$ and $u' = u + d_j$. Viewing the depot s as the tuple $(s, 0, 0)$, we add an edge from s to (c_i, t, u) if $t = \max(s_i, t + \lceil d(s,i)/\Delta \rceil)$ and $u = d_i$. With these definitions it follows that for any path P from s to (c_i, t, u) the total demand of customers on P is u and the time required to traverse P is t. Note that if P visits multiple vertices associated with customer j, d_j would be included in u with the corresponding multiplicity.

In addition to the above edges we include in the graph, edges $(c_i, t, u), (c_i, t+1, u)$ traversing which implies the vehicle waits at customer location c_i for Δ time. We also have edges from $(c_i, t_i, u), i \in [n], u \in [U]$ to s and these allow the vehicle to return to its depot. To reduce the size of the graph constructed one can introduce a discretization factor Γ, for the capacities/demands. Thus the vehicle capacity would be replaced by $\lfloor U/\Gamma \rfloor$ and demands d_i by $\lceil d_i/\Gamma \rceil$, $i \in [n]$.

For the remainder we follow the approach of this paper. We formulate an integer program and introduce additional commodities to strengthen the Linear Programming relaxation. Solving the strengthened LP optimally and using iterative rounding on the fractional solution will give us a set of vehicle routes which would service all customers while respecting the time-window

Results for Uncapacitated

Table 1: Instances with 25 customers

Model	VTLP	SLP	Improvement(%)	VTIP	Algorithm	Number of iterations	Error
R101	8	8	0	8	8	2	0
R102	6.5	6.5	0	7	7	5	0
R103	4	4	0	4	4	1	0
R104	4	4	0	4	4	4	0
R105	5.5714	5.5714	0	6	6	2	0
R106	4.80	4.80	0	5	5	3	0
R107	3.7549	3.78148	0.7079	4	4	3	0
R108	3.5405	3.6133	2.0562	4	4	4	0
R109	4.2707	4.5	5.3691	5	5	4	0
R110	3.8495	3.965	3.0003	4	5	5	1
R111	3.6720	3.75	2.1242	4	5	4	1
R112	3.384	3.48101	2.8667	4	4	4	0
C101	3	3	0	3	3	3	0
C102	2.33	2.33	0	3	3	3	0
C103	2.25	2.25	0	3	4	4	1
C104	2.1885	2.1899	0.0640	3	3	3	0
C105	3	3	0	3	3	2	0
C106	3	3	0	3	3	2	0
C107	3	3	0	3	3	3	0
C108	2.60	2.60	0	3	3	3	0
C109	2.25	2.25	0	3	3	3	0
RC101	3.9166	3.9166	0	5	5	3	0
RC102	3.325	3.325	0	4	4	3	0
RC103	3.0318	3.05	0.6003	4	4	3	0
RC104	3	3	0	4	4	4	0
RC105	4.5	4.5	0	5	5	5	0
RC106	3.332	3.4428	3.3253	4	5	5	1
RC107	2.9839	3.08	3.22062	4	4	3	0
RC108	2.888	2.944	1.9391	3	4	3	1

Table 2: Instances with 50 customers

Model	`VTLP`	`SLP`	Improvement(%)	`VTIP`	Algorithm	Number of iterations	Error
R101	12	12	0	12	12	5	0
R102	10	10	0	10	10	6	0
R103	8	8	0	8	9	8	1
R104	5.7337	5.77	0.6331	6	7	7	1
R105	8.333	8.333	0	9	9	5	0
R106	7.058	7.066	0.1133	8	8	5	0
R107	6.334	6.401129	1.0598	7	7	6	0
R108	5.6434	5.6987	0.9799	6	7	6	1
R109	6.9576	7.1325	2.5138	8	8	7	0
R110	6.4441	6.623	2.7762	7	8	6	1
R111	6.176	6.3108	2.1826	7	7	5	0
R112	5.66117	5.8615	3.5386	6	7	6	1
C101	5.8	5.8	0	6	6	6	0
C105	5.25	5.25	0	6	6	5	0
C106	5.6	5.6	0	6	6	6	0
C107	5	5	0	5	6	6	1
RC101	7.953	7.953	0	9	10	5	1
RC102	6.708	6.708	0	8	9	6	1
RC103	6.0909	6.144	0.8718	7	8	7	1
RC104	5.74948	5.77	0.3569	7	7	7	0
RC105	7.3265	7.3265	0	8	8	4	0
RC106	6.7071	6.788	1.2061	8	8	6	0
RC107	6.2182	6.2182	0	7	8	7	1
RC108	6.168	6.168	0	7	8	6	1

Table 3: Instances with 100 customers

Model	`VTLP`	`SLP`	Improvement(%)	`VTIP`	Algorithm	Number of iterations	Error
R101	19.25	19.25	0	20	20	9	0
R102	17.25	17.25	0	18	18	9	0
R105	14.166	14.166	0	15	16	12	1
R106	12.321	12.321	0	13	13	10	0
C101	11.4285	11.4285	0	12	12	9	0
RC101	15.2083	15.2083	0	16	17	11	1
RC102	12.8753	12.8784	0.024	14	14	10	0
RC105	13.2399	13.337	0.733	14	15	13	1
RC106	12.5182	12.71288	1.555	13	15	14	2

and vehicle-capacity constraints. Note that in this formulation and the solution obtained, each customer is serviced by only one vehicle.

Our framework is robust enough to handle other extensions. The setting of multiple depots can be addressed by introducing multiple starting vertices, one for each depot and minimizing the total flow out of these starting vertices. If vehicles are heterogeneous and have different capacities, we can create a depot for each vehicle capacity. The depot corresponding to vehicle capacity U_i will have edges from all vertices $(c_j, t_j, u), j \in [n], u \in [U_i]$. Constraints on numbers of vehicles of a certain capacity can be enforced by bounding the sum of the x_e values on edges leaving the corresponding depot.

6.1 Results and Comparisons Tables 4, 5 and 6 shows results for capacitated version when the number of customer locations are 25, 50 and 100 respectively.

The Tables show a comparison between the optimum value of the Integer Program (), the optimum value of the linear relaxation to the integer program (). We also report the number of vehicles used in the solution returned by our algorithm in the column labeled "Algorithm" and report the difference from the optimum solution of in the column labeled "Error". Finally, we report the number of iterations required by our algorithm. This measure informs us of the number of times the strengthened Linear Program is solved by our algorithm

We ran our algorithm on 29 instances for 25 customers each. Of these 29 instances, our algorithm returned the optimum solution for 26, and for the remaining 3 instances the solution obtained by our algorithm was one more than the optimum.

We ran our algorithm on 29 instances with 50 customers each. Of these 15 instances, our algorithm returned the optimum solution for 23, and for the 11 instances the solution obtained by our algorithm was one more than the optimum , and for remaining 3 our solution was two more than the optimum.

We ran our algorithm on 20 instances with 100 customers each. Of these 20 instances, our algorithm returned the optimum solution for 7, and for the 8 instances the solution obtained by our algorithm was one more than the optimum.Our algorithm exceeded optimum solution by 2 and 3 for three and one instances respectively. We were not able to solve integer program for some cases due to time constraint(18000 seconds).

`https://bit.ly/3A5u842`

7 Conclusions

The paper provides a new approach to obtain near optimal; solutions for various versions of the vehicle routing problem. The experiments we have conducted for the `Vehicle routing with time windows` problem serve to demonstrate the promise of this approach.

The strengthened Linear program we have formulated has a large number of variables and constraints and these are likely to increase further when we include capacity and other constraints. However the `SLP` is a positive LP and so combinatorial algorithms for solving such LPs can potentially be used for finding fast approximate solution to the `SLP`. The rounding algorithms we propose is linear time and hence the bottleneck is the time required to solve the `SLP`.

References

[1] Philippe Augerat, D Naddef, JM Belenguer, E Benavent, A Corberan, and Giovanni Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. 1995.

[2] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

[3] Roberto Baldacci, Eleni Hadjiconstantinou, and Aristide Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations research*, 52(5):723–738, 2004.

[4] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations research*, 59(5):1269–1283, 2011.

[5] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.

[6] Michel L Balinski and Richard E Quandt. On an integer program for a delivery problem. *Operations research*, 12(2):300–304, 1964.

[7] Nicos Christofides, Aristide Mingozzi, and Paolo Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282, 1981.

[8] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

[9] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

Results for Capacitated Version

Table 4: Instances with 25 customers

Model	LP	First Iteration	IP	Error(LP-IP)	Iteration
R101	8	8	8	0	2
R102	7	6.5	7	0	3
R103	4	4	4	0	1
R104	4	4	4	0	2
R105	6	5.5714	6	0	4
R106	5	4.800	5	0	5
R107	4	3.7549	4	0	3
R108	4	3.5405	4	0	4
R109	5	4.2707	5	0	4
R110	5	3.8495	4	1	4
R111	5	3.6720	4	1	5
R112	4	3.3848	4	0	4
C101	3	3	3	0	2
C102	3	2.3333	3	0	3
C103	3	2.3000	3	0	3
C104	3	2.3000	3	0	3
C105	3	3	3	0	3
C106	3	3	3	0	3
C107	3	3	3	0	3
C108	3	2.6	3	0	3
C109	3	2.3	3	0	3
RC101	5	3.9166	5	0	4
RC102	4	3.325	4	0	4
RC103	4	3.0318	4	0	4
RC104	4	3	4	0	4
RC105	5	4.5	5	0	5
RC106	4	3.3319	4	0	4
RC107	4	2.9839	4	0	3
RC108	4	2.8888	3	1	4

Results for Capacitated Version

Table 5: Instances with 50 customers

Model	LP	First Iteration	IP	Error(LP-IP)	Iteration
R101	12	12	12	0	2
R102	10	10	10	0	4
R103	8	8	8	0	7
R104	7	5.7337	6	1	7
R105	9	8.3333	9	0	5
R106	8	7.058	8	0	6
R107	7	6.3343	7	0	6
R108	7	5.6434	6	1	7
R109	8	6.9576	8	0	8
R110	9	6.4441	7	2	8
R111	7	6.1760	7	0	6
R112	8	5.6611	6	2	8
C101	6	5.8	6	0	5
C102	5	5	5	0	4
C103	5	4.3646	5	0	5
C104	5	4.3	5	0	5
C105	6	5.25	6	0	6
C106	6	5.6	6	0	6
C107	6	5	5	1	6
C108	5	5	5	0	5
C109	6	4.4444	5	1	6
RC101	10	7.9531	9	1	6
RC102	9	6.7081	8	1	6
RC103	8	6.0909	7	1	8
RC104	8	5.7494	7	1	8
RC105	9	7.3265	8	1	6
RC106	9	6.7071	8	1	8
RC107	8	6.2182	7	1	7
RC108	9	6.1686	7	2	9

[10] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009.

[11] Gerd Finke. A two-commodity network flow approach to the traveling salesman problem. *Congresses Numeration*, 41:167–178, 1984.

[12] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006.

[13] Bruce L Golden, Thomas L Magnanti, and Hien Q Nguyen. Implementing vehicle routing algorithms. *Networks*, 7(2):113–148, 1977.

[14] A.N. Letchford and R.W. Eglese. The rural postman problem with deadline classes. *European Journal of Operational Research*, 105(3):390–400, 1998.

[15] Amos Levin. Scheduling and fleet routing models for transportation systems. *Transportation Science*, 5(3):232–255, 1971.

[16] Jens Lysgaard, Adam N Letchford, and Richard W Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[17] Arthur D O'Connor and CA De Wald. A sequential deletion algorithm for the design of optimal transportation networks 37th national meeting of the operations research society of america. *Bulletin of the Operations Research Society of America*, 18(1), 1970.

[18] Ted K Ralphs, Leonid Kopman, William R Pulleyblank, and Leslie E Trotter. On the capacitated vehicle routing problem. *Mathematical programming*, 94(2):343–359, 2003.

[19] Yan-Jun Shi, Fan-Wei Meng, and Guo-Jiang Shen. A modified artificial bee colony algorithm for vehicle routing problems with time windows. *Information Technology Journal*, 11(10):1490, 2012.

[20] Marius Mihai Solomon. Vehicle routing and scheduling with time window constraints: Models and algorithms. Technical report, 1984.

[21] Fabien Tricoire, Nathalie Bostel, Pierre Dejax, and Pierre Guez. Exact and hybrid methods for the multiperiod field service routing problem. *Central European Journal of Operations Research*, 21(2):359–377, 2013.

[22] Ziauddin Ursani, Daryl Essam, David Cornforth, and Robert Stocker. Localized genetic algorithm for vehicle routing problem with time windows. *Applied Soft Computing*, 11(8):5375–5390, 2011.

[23] Yang Yu. Yao, 2011 yu b., yang z., yao b. *A hybrid algorithm for vehicle routing problem with time windows, Expert Systems with Applications*, 38(1):435–441, 2011.

Results for Capacitated Version

Table 6: Instances with 100 customers

Model	LP	First Iteration	IP	LP-IP	Iteration
R101	20	19.25	20	0	8
R102	18	17.25	18	0	11
R103	14	13	13	1	13
R104	12	10.3612	11	1	12
R105	15	14.166	15	0	10
R106	14	12.3212	13	1	12
R107	13	10.8608	12	1	12
R108	14	10.1643	11	3	14
R109	14	11.6858	12	2	13
R110	13	10.9643	12	1	13
R111	12	10.6995	11	1	11
R112	14	10.1750	11	3	14
C105	11	10.3750	11	0	9
C106	11	10.3082	11	0	9
C107	10	10	10	0	8
RC101	18	15.2083	16	2	14
RC102	14	12.8753	14	0	8
RC103	13	11.3561	12	1	11
RC105	15	13.2399	14	1	13
RC106	15	12.5182	13	2	13