A Savings Algorithm Approach to the Truck Routing and Loading Problem

Adhrit Shetty
Department of Information Technology
Bharatiya Vidya Bhavans Sardar Patel Institute of Technology
Mumbai, India
adhritshetty2908@gmail.com

Kevin Sijo Puthusseri
Department of Information Technology
Bharatiya Vidya Bhavans Sardar Patel Institute of Technology
Mumbai, India
kevinsijo@gmail.com

Akshat Shetty
Department of Information Technology
Bharatiya Vidya Bhavans Sardar Patel Institute of Technology
Mumbai, India
akshatshetty2908@gmail.com

Radha Shankarmani
Department of Information Technology
Bharatiya Vidya Bhavans Sardar Patel Institute of Technology
Mumbai, India
radha_shankarmani@spit.ac.in

***Abstract*—The Truck Routing and Loading problem is a combination of the Vehicle Routing Problem (VRP) and the Bin Packing problem. In this paper, we propose an integrated solution to the routing and loading problem, while maintaining a balance between computation time and accuracy in order to make this a practically viable solution. The aim is to deliver all the goods to their destination, covering as less distance as possible, over as few routes as possible. For the loading problem is solved using a Bin Packing algorithm, with the aim to best-fit the truck using an exhaustive iterative search to find an optimal combination of items that occupy the maximum weight and volume that the truck can possibly carry. The routing problem a solution to the Capacitated Vehicle Routing Problem (CVRP) has been proposed. It is an improved variant of the heuristics-based Clarke-Wright Savings algorithm. Computational experiments have been carried out to prove the effectiveness of the solution, including analyzing its performance for some standard CVRP benchmarks.**

***Index Terms*—Truck Routing Loading, Vehicle Routing Problem, Bin Packing, Savings Algorithm, Combinatorial Optimization, Heuristics**

I. Introduction

Transportation causes a value addition of 5% to 20% to the total cost of goods [1]. Given this impact, a lot of importance is given to the planning of routes by logistics companies. The objective has always been to supply all the goods as quickly and as safely as possible while burning as little fuel as possible. This led to the usage of automated routing softwares, in the late 1970s and early 1980s, to deploy transportation assets in a way that helped reduce costs.

Any practical solution to transportation related problems broadly deals with three categories - Routing, Scheduling and Loading. Over the past few decades significant progress has been made in the above mentioned categories, but majorly independently. Since any practical application needs to solve concerns that cut across at least two of the above mentioned categories, if not all three, we dealt with the Routing and Loading in unison. The proposed procedure is two fold - first generating optimally filled trips for each destination town separately, followed by, using heuristics to combine non-ideal trips to optimise for distance saving.

The Bin Packing problem, a widely-known NP-Hard problem in combinatorial optimization, primarily deals with the calculation of the minimum number of bins can pack a set of given input data. The problem has found application in domains like operations research, computer science, and engineering, where the items and bins of multi-dimensions are to be packed [2]. Designing practical solutions for real world applications of Bin Packing, such as the one explored in this paper, requires understanding feasibility of parameters under consideration [3]. The intended optimization must not become too expensive with respect to time or computation, rendering it infeasible for real-time usage. Analogous to the Bin Packing problem, for the Loading task, we consider a single truck with a maximum weight and volume carrying capacity, that represents the bin that needs to be filled. Filling needs to be done in such a way that both the constraints (occupied weight and volume) are maximized simultaneously. The spatial arrangement of items inside the truck and the weight of the items combined would be the ideal constraints to maximize. However, since volume and weight alone provide a fair approximation while being computationally less expensive by two orders, we chose these to be the sufficient constraints on the basis of which we fill the truck. The delivery truck in consideration is best-fit, that is, no other combination of items from the set of available items will occupy as much weight and volume as the chosen set of items will. This set of items is then used to generate consecutive trips for the delivery truck till either any one of the item-types available count is less than the count of that item-type in the (best) chosen set. Once the current best set is invalidated, a new best-set is generated with the items in the available set. This process is repeated till a list of unique trips are formulated in order to deliver all the items that a given destination town has requested.

The Routing task focuses on solving the Vehicle Routing Problem (VRP), another well-known NP-Hard problem in combinatorial optimization, specifically the Capacitated Vehicle Routing Problem (CVRP). Dantzig and Ramser first proposed the problem and because of its importance it has been under intensive research ever since [4]. Finding a set of feasible vehicle routes while optimizing the total distance travelled and/or the total number of vehicles used is the main goal in CVRP. For each route, the vehicle departs from the source depot and returns back to the source itself after completing the trip. CVRP involves a single source depot, a uniform fleet of vehicles, and a set of destination towns who require delivery of goods from the depot. Solutions designed to tackle the VRP and its variants developed into two categories: Exact and Heuristic. Exact solutions are those that solve the problem to optimality by computing the distance of every feasible solution and then choosing a solution with minimum distance, using concepts of enumeration, branch-and-bound or dynamic programming. This approach consists of algorithms such as the branch-and-bound algorithm [5], the branch-and-cut algorithm [6]–[8], and the branch-and-cut-and-price algorithm [9]. In these algorithms, CVRP instances involving more than 100 destination towns can rarely be solved to optimality owing to the extensive amount of time it requires, computationally. Given that an exact polynomial algorithm has not been found yet to overcome the problem, new methods were explored using heuristics in order to find feasible if not optimal solutions. The Savings algorithm, developed by Clark and Wright [10], is an established heuristic used to find feasible solutions to the VRP. It involves creating routes for each destination town, starting and ending at the depot. For each pair of routes the savings are computed, which is nothing, but the distance saved by merging two routes. The savings are then sorted in descending order and associated routes are merged starting from the top. The merge is valid as long as there are no common delivery locations on the two routes and no other demand (goods) constraint is violated. This merging continues until no additional savings can be achieved. This paper draws heavily from the framework provided by the Savings algorithm.

Over time many variants of the savings algorithm have been developed. One of them is mentioned in "An Improved Savings Method for Vehicle Routing Problem" [11]. Here the the rules of selecting the individual routes to add for savings are altered. In this procedure the one with the maximum savings is considered as an initial route. Then among the other routes the one with the maximum savings, and which contains either the starting or ending customer of the initial route, is added to it. This is repeated till the constraints of the truck are not crossed and then iteratively for all the remaining individual routes. The CVRP algorithm proposed in this paper builds on the above logic. It was applied to the instance mentioned in that very paper. [11] The implementation of the original Clark and Wright Savings algorithm solved the instance with a total of 3 trips and 201.49 units of distance covered. The improved algorithm they proposed did the same in 4 trips and with a total of 167.31 units covered. [11] The solution this paper proposes does the same in 3 trips and a total of 152.29 units covered.

II. Problem Definition

The important mathematical terms and symbols are specified in Table I.

TABLE I
MATHEMATICAL MODEL OF THE ALGORITHM

Symbols	Meaning
S	Depot/Source
D_i	Destination ($i\epsilon(1,N)$)
N	Number of destinations towns
n	Number of types of goods
W_{truck}	Maximum weight bearing capacity of truck
V_{truck}	Maximum volume bearing capacity of truck
$subject_{dist}$	Distance of the subject trip route
$candidates_{dist}$	Distance of the candidate trip route
Θ	Distance of new merged route
δ	Distance saved by merging the trips
W_{fact}	Denotes the consumption of the trucks weight bearing capacity as a factor of the total
V_{fact}	Denotes the consumption of the trucks volume bearing capacity as a factor of the total
α	Factor signifying the pairing capacity of the particular candidate trip
β	Factor signifying the pairing capacity of the merged trip
φ^x, φ^y	Candidate filtering parameters
$\varphi^x_{max}, \varphi^y_{max}$	Maximum values for the candidate filtering parameters
$\varphi^x_{limit}, \varphi^y_{limit}$	Minimum thresholds for the Candidate filtering parameters
$\varphi^x_{update}, \varphi^y_{update}$	Update values for φ^x_{limit}, φ^y_{limit}
λ	Factor calculated for candidate selection for merging
$\eta_1, \eta_2, \eta_3, \eta_4$	Weights for the formula of λ

A. Statement

Consider a static VRP with one depot and multiple destinations where S represents the depot and D_i represents the destination. N represents the number of towns. Each destination can demand any number of goods of varying weights and volumes. There are no time windows for delivery. The number of types of goods, discriminated based on weight and volume, is represented by n. Each truck has 2 constraints maximum weight bearing capacity represented by W_{truck} and maximum volume bearing capacity represented by V_{truck}. There is only one type of truck considered, hence the values of these constraints are constant.

B. Objective

The objective is to deliver all the goods to the appropriate destinations, optimizing the total distance covered and the total number of trips required.

C. Constraints

The depot is the only source of goods, so each town other than it is a point of drop and not pickup. Once a trip is completed the truck (vehicle) has to return to the depot to load up for the next trip. While solving the Bin Packing problem only the weights and volumes of the goods are considered as factors and not their dimensions.

III. Truck Routing-Loading Algorithm

The entire truck routing and loading procedure is broken into two major phases:

1) Initial Loading - It involves the sections, destination specific loading of trucks and creation of a routing table.
2) Trip Merging - The sections involved in this phase are sorting and elimination and merging of routes.

There also is a section called Optimum Weight Values which discusses the weights used in the algorithm.

A. Destination Specific Loading of trucks

The primary solution revolves around the capability to generate all unique combinations of the variety of items in the inventory and exhaustively search for the one combination that is fitting the best. In order to generate unique combinations, the simple logical model is an n-level nested loop (where, n = number of types of items) such that each level ranges from 0 to the maximum number of items of that type the truck can carry. The challenge lies in the fact that the number of types of items varies based on the types requested by a given destination. To explain the more generic solution, we resort to the notion of n-tuples of mixed radices [12]. An n-tuple is an ordered list of items, that is, each position in the list coveys a specific meaning. Furthermore, if every position in the list is ranged from 0 to q, it would be called an n-tuple of radix q. Clearly, a list wherein each position ranges from 0 to it's own limit, say q_i, would be called an n-tuple with mixed radices. Thus, we can generate all unique combinations in which n types of items can be chosen, by generating all n-tuples of their respective mixed radices. For example,

Let there be 3 types of items with total count of 1, 2 and 3 respectively. Thus, the list of all 24 n-tuples with mixed radices of 1,2,3 will be: [1,2,3] [1,2,2] [1,2,1] [1,2,0] [1,1,3] [1,1,2] [1,1,1] . . [0,0,0]. Using the generated list of all n-tuples, since we know the volume and weight each type occupies, we can iteratively check for the n-tuple that will result in maximum packing of weight and volume. This also allows us to set thresholds for fraction of total weight and volume occupied which may not chose the most ideal tuple but satisfies realistic scenarios. This flexibility helps in expanding the usage of such a Truck Loading procedure into scenario-specific ones very easily.

B. Creation of Routing table

For merging of trips, a routing table is created. It has 8 columns - trip number, S, list of D_i in order of traversal, list of goods of each type, V_{fact}, W_{fact}, total distance of the trip and a viability factor. Each row in the routing table signifies a different trip and has data related to that particular trip. The routing table is shown in Table II.

C. Sorting and Elimination

To understand the purpose of the sorting and elimination process, the difference between workable and redundant trips will have to be known.

TABLE II
STRUCTURE FOR ROUTING TABLE

Trip Number	S	List of D_i	List of Goods	V_{fact}	W_{fact}	Viability	Total Distance

- **Workable trips**: The trips which have the possibility of merging with another trip to reduce overall distance travelled and the total number of trips are called workable trips.
- **Redundant trips**: The trips which do not have the possibility of merging with another trip due to a certain constraint(s) not being satisfied are called redundant trips.

The objective is to separate the redundant trips from the workable trips. First the trips in the routing table are sorted in ascending order according to its W_{fact}. Each trip in the routing table is compared with the first trip in the routing table, the one consuming the least of the vehicles weight bearing capacity. If the addition of the W_{fact} values of the two trips is greater than 1 then it is a redundant trip, else it is registered as a workable trip. The redundant trips are removed from the routing table. The same procedure of sorting and eliminating is executed with the V_{fact} being considered as the factor for elimination.

At any point in the algorithm only the workable trips are to be considered for merging, hence the sorting and elimination procedure is necessary to make sure that the routing table does not contain redundant trips.

D. Merging of Routes

From the list of workable trips, pairs are found to be merged into one trip. This involves the following four steps implemented sequentially - subject and candidate search, distance viability and factor generation, candidate filtering and candidate selection. These steps are repeated indefinitely until the number of workable trips becomes less than or equal to one. The explanation of these steps is given below.

- **Subject and Candidate search:** Subject is the trip being considered for a merge, while a candidate is one of the many trips being considered to merge with the subject. The subject is set to the last trip in the routing table unless it has been updated in the previous iteration. The specifics of this exception are explained later in the candidate filtering section. Since the routing table has been sorted earlier, the last trip is the one which will have the highest V_{fact}. Selecting it as the subject of each iteration is to ensure that trips even with high W_{fact} or V_{fact} values have a good chance of merging with other trips to form completely filled trucks. The candidates are found by adding the W_{fact} and V_{fact} of the subject trip to the W_{fact} and V_{fact} of each workable trip respectively, to see if the weight and volume bearing constraints of the truck are satisified. If they are then it is considered as a valid candidate.
- **Distance Viability and Factor generation:** Once the weight and volume constraints of the candidate-subject

pairs have been checked the distance validity of these trips have to be considered. The distance validity of the trips are checked by considering the towns involved in the route of the subject trip and a particular candidate trip together and finding the shortest path to traverse all these towns and the distance covered in doing so. This distance is represented by Θ. The algorithm used in this solution for shortest path finding follows the approach of dynamic programming. However, the algorithm itself is not of importance here and any algorithm which provides decent results in a suitably short period of time can be used.

$$\delta = (subject_{dist} + candidates_{dist}) - \Theta \tag{1}$$

δ is calculated using equation 1 to check if the merging of routes creates any distance saving. If δ is greater than 0 then that particular candidate trip is distance viable, otherwise it is removed from the candidate list. At this stage if no trips are left in the candidates list, then the viability attribute of that particular subject is made false and it is registered as a redundant trip. The control goes to the next iteration and a new subject is found. However, if the candidates list is not empty, then a set of factors are generated for each subject-candidate pair which will be used later to select one of them for merging. The factors used are Θ, δ, V_{fact}, W_{fact}, α and β. α is calculated by counting the number of trips in the routing table the particular candidate trip can be a candidate for, other than the current subject. While β is calculated by temporarily merging the subject trip and the candidate trip, and counting the number of possible candidates for the new merged trip. While finding candidates to calculate α and β we perform candidate search and distance viability check as explained above.

- **Candidate filtering:** The viable candidates are further filtered based on 2 parameters, φ^x and φ^y, the values of which are calculated using equations 2 and 3. φ^x denotes the distance saving of the merged trip as a fraction of the total distance of the trip. It is used as a parameter for filtering as it represents the key element of route merging, which is distance saving. φ^y is β which gives us an idea of the potential of the newly merged trip to merge again. It is important to merge routes in such a way that leaves scope for future mergers. Hence φ^y is used as the second parameter. The purpose of this filtering is to ensure that only the most optimum merges occur while everything else is eliminated.

$$\varphi^x = \delta / \Theta \tag{2}$$

$$\varphi^y = \beta \tag{3}$$

For each trip in the candidates list these 2 parameter values are calculated. The maximum values, φ^x_{max} and φ^y_{max}, for these 2 parameters for a particular list of candidates are found. These are important for setting the values of the parameter limits φ^x_{limit} and φ^y_{limit}, which serve as minimum thresholds for the filtering. If φ^x is less than φ^x_{limit} or φ^y is less than φ^y_{limit}, then the candidate trip is removed from the candidate list. The values of φ^x_{limit} and φ^y_{limit} are initially zero and then set via an update strategy. Here we need to introduce 2 new values called the limit update values, φ^x_{update} and φ^y_{update}. The update statements for the parameter limits are mentioned in equations 4 and 5. The values of φ^x_{update} and φ^y_{update} are discussed in the optimum weight values section.

$$\varphi^x_{limit} = \varphi^x_{max} * \varphi^x_{update} \tag{4}$$

$$\varphi^y_{limit} = \varphi^y_{max} * \varphi^y_{update} \tag{5}$$

The parameter limits are not necessarily updated for every new candidate list being considered. The parameter limits set for a certain set of candidates continues for the next one. It is possible that the parameter limit values are greater than even the maximum parameter values for the current set of candidates, but that does not lead to an update in them. The reason for this is that we are not finding the best candidates for a particular subject, but the best pair of subject and candidate across the table for merging. Hence it is optimum if some subjects are left with no candidates after the filtering due to high parameter limit values. The parameter limits will be updated only in the following two cases -

1) *If $\varphi^x_{limit} < \varphi^x_{max} * \varphi^x_{update}$ or $\varphi^y_{limit} < \varphi^y_{max} * \varphi^y_{update}$:* Here the filtering occuring is not adequate, as the values are less than required and hence the parameter limits have to be updated.
2) *If the parameter limits are too high for any merging to occur across the routing table:* Here the parameter limits have to be updated for further merging to occur. This case has been explained in detail ahead.

In this way if no candidates are left after this final filtering stage then one of the following two actions may be undertaken based on the value of the subject -

1) *If the subject is any trip with position greater than one in the routing table:* Now the subject of the merging is updated. The new subject is the trip in the routing table at an index position of one lesser. The purpose of this is to continue the merging process with a new subject instead of decreasing the strictness of the filtering parameters.
2) *If the subject is the first trip in the routing table even though the routing table contains other workable trips:* The subject cannot be updated any more and the parameter limits have to be updated for any merge to occur across the routing table. Hence the subject is set to the last trip in the routing table and the parameter limits are updated to accommodate the φ^x_{max} and φ^y_{max} values of the current candidates list, using the previously mentioned update state-

ment. This is the second case in which the parameter limits are updated.

- **Candidate selection:** This involves selecting one trip for merging with the subject trip out of all those in the candidates list. For this purpose a value λ is calculated for each trip using four factors, as shown in equation 6. The trip having the largest value is selected for merging.

$$\lambda = \eta_1 * \delta - \eta_2 * \Theta - \eta_3 * \alpha + \eta_4 * \beta \tag{6}$$

The greater δ is the more useful the merger is to the overall solution. Hence it has a positive sign attached to it. Similarly for β, the greater this value is the better it is for the merging process. If two candidates have equally good distance savings, the one with the lesser Θ value should be chosen. This will help achieve a decrease in overall distance covered and additionally give preference to mergers between closely situated destinations before far flung destinations. Thus, leading to more sensible mergers. Because of this Θ has a negative sign attached to it. The greater α is, greater is the possibility of merging with other trips. If there are two equally good candidates, such that one has a lesser possibility of merging with other trips, then the former should be given preference over the latter for merging with this particular subject, as denying it now may lead to it never getting merged. However, denying the other candidate will not have that bad an impact on our solution as it has other trips it can merge with. Hence Θ has a negative sign attached to it. The values of η_1, η_2, η_3 and η_4 are discussed in the optimum weight values section.
After a candidate trip has merged with the subject trip it is placed as an entry in the routing table. The entries in the routing table for the subject and chosen candidate are removed as they no longer exist as separate trips. On this updated table the sorting and elimination procedure explained previously is carried out. This is to remove the redundant trips from the updated table. The control flows to the next iteration and a new subject is found.

E. Optimal Weight Values

The proposed algorithm includes the some weights whose values are highly integral to the merging of routes. These weights are φ^x_{update}, φ^y_{update}, η_1, η_2, η_3 and η_4. Their values have been decided after observing the results over multiple computational experiments. φ^x_{update} and φ^y_{update} decide the strictness of the candidate filtering procedure. Too small a value would lead to minimalistic filtering and unfavourable mergers while too large a value may lead to missing on some important mergers. Of the two parameters φ^x is more fundamental to the filtering procedure, hence $\varphi^x_{update} = 0.8$, while $\varphi^y_{update} \epsilon (0.4, 0.7)$. Values η_1, η_2, η_3 and η_4 decide the importance of the different factors in the candidate selection formula. $\eta_1 \epsilon (1.4, 1.5), \eta_2 = 1, \eta_3 = 1$ and $\eta_4 = 1$. η_1 has a higher value than η_2 to give greater importance to distance saved than total distance. Even though α and β have lower values as compared to δ and Θ, η_3 and η_4 have small values to limit their importance in candidate selection to a tie-breaker in most cases. φ^y_{update} and η_1 have a range of values and for this purpose a weight tweaking shell is used which is responsible for incrementing and feeding their values to the merging process. The increment value for φ^y_{update} is 0.05 and for η_1 is 0.05. The iteration with least total distance is selected as the final output.

IV. Pseudocode

The pseudo code of the merging of routes is shown in Algorithm 1.

Algorithm 1 Merging of Routes

```
Input: routing table, $\varphi^y_{update}$, $\eta_1$
Initialization: workable trips list, candidates list
Perform sorting and elimination to obtain workable trips;
while $numberofworkabletrips > 1$ do
    Set subject considering update in previous iteration;
    Perform candidates search on workable trips;
    while $j \leq numberofcandidates$ do
        Find $\Theta$ for the subject candidate pair;
        Calculate $\delta$ using formula (1);
        if $\delta < 0$ then
            Eliminate from candidates list
        else
            Find $V_{fact}$, $W_{fact}$, $\alpha$ and $\beta$ for the candidate;
        end if
    end while
    if $numberofcandidates > 0$ then
        while $j \leq numberofcandidates$ do
            Calculate $\varphi^x$ and $\varphi^y$ using formulas (2) and (3);
        end while
        Calculate $\varphi^x_{max}$ and $\varphi^y_{max}$;
        Conditional limit update using formulas (4) and (5);
        while $j \leq numberofcandidates$ do
            Candidate filtering using $\varphi^x_{limit}$ and $\varphi^y_{limit}$;
        end while
        if $numberofcandidates > 0$ then
            while $j \leq numberofcandidates$ do
                Calculate $\lambda$ using formula (6);
            end while
            Select the candidate trip with maximum $\lambda$;
            Merge the selected candidate trip with the subject;
            Perform sorting and elimination on updated trips;
        else
            if $positionofsubject = 1$ then
                Update filtering parameter limits;
            else
                Update subject to trip of lesser index;
            end if
        end if
    else
        Eliminate subject from workable trips list;
    end if
end while
```

V. Computational Experimentation

The proposed algorithm was programmed in Java 8 and implemented on an Intel Core i3 machine with 4 GB of memory.

A. Case Data

A logistics company has multiple identical trucks, having a volume bearing capacity of 8600 units and a weight bearing capacity of 6250 units. It has to deliver goods to 10 customers (town 1-10) from its depot (town 0). The goods to be delivered are of 4 types. The weight and volume specifications of these goods are mentioned in Table III. The co-ordinates and demand of each town are mentioned in Table IV.

TABLE III
Types Of Goods

Sr. No.	Volume	Weight
1	100	72
2	275	300
3	780	1200
4	920	870

TABLE IV
Town Co-ordinates and Demand

Town	X-Coordinate	Y-Coordinate	Demand			
			Type 1	Type 2	Type 3	Type 4
0	70	40	-	-	-	-
1	75	350	10	0	3	0
2	80	500	5	6	0	3
3	-150	200	0	7	0	0
4	10	650	3	2	2	5
5	60	-450	0	0	4	0
6	1000	45	0	17	0	0
7	-40	-80	20	0	0	10
8	-5	340	0	22	0	0
9	35	-35	11	0	13	0
10	95	-220	12	20	10	5
Total Demand of each Type			61	74	32	23

B. Results and Comparison

Figure 1 depicts the subject and candidates during one of the iteratons of the route merging process. The routes in grey and pink, route A and route B, are the potential routes after the candidates, A and B respectively, have merged with the subject. From the figure it is visible that candidate A is closer to the subject while candidate B is closer to the source. The values of the different parameters for these routes are -

- **Route A:** δ = 907, Θ = 1238, α = 5, β = 0, λ = +27
- **Route B:** δ = 588, Θ = 950, α = 10, β = 0, λ = -137

Route A has a higher δ but has a higher Θ as well. The difference in α is 5 in favour of B, but that is negligible as compared to the other values. There is no difference in β. Hence route A ends up having a greater λ due to the greater weight assigned to δ. Candidate A is selected for merging with the subject.

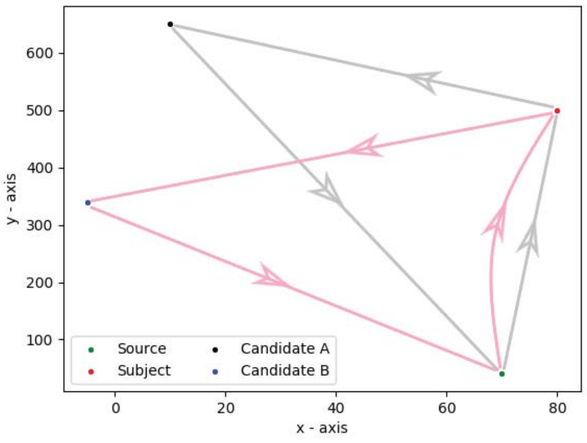

Fig. 1. Subject and Candidate Trip Routes.

The results of the above-mentioned case is mentioned in Table V. This includes the route to be followed by the truck starting and ending at the depot, the goods being carried by it, V_{fact}, W_{fact} and the total distance covered by it. For most of the routes either V_{truck} or W_{truck} is completely satisfied, confirming the optimization in total number of trips and total distance covered.

The performance of the algorithm was tested by applying it to the instances from Eilon et al [13]. As mentioned in Table VI, the algorithm performs well for these instances and gives a result which is either the best-known result of the official records or is in close proximity of it. The reason for this is the number of factors that are considered while merging the routes other than just distance saved. The combination of these factors helps us choose a suitable pair of routes for merging repeatedly until no more merging is possible. Other than approaching the best-known results, the algorithms also manages to do so in a very short period of time. This is because of its savings algorithm approach which requires minimum number of runs of the problem to reach a solution. This reduced computation time makes this algorithm a potential solution for real time truck routing and loading.

VI. Conclusion and Future Work

The proposed algorithm is a solution to the truck routing and loading problem. It solves the Bin Packing problem by exhaustively searching for the best-fit n-tuple of items each time in an optimal timespan. A savings algorithm approach has been used to solve the VRP. This algorithm introduces factors to signify the pairing capacity of the routes before and after merging to assist in the decision-making process while merging the routes. The algorithm produces good results when applied to standard CVRP benchmarks.

TABLE V
RESULTS FOR CASE DATA

Sr.No.	Route	Goods Carried				V_{fact}	W_{fact}	Distance Covered
		Type 1	Type 2	Type 3	Type 4			
1	0-7-3-0	6	7	0	4	0.72	0.96	735.65
2	0-10-0	0	0	5	0	0.45	0.96	522.4
3	0-9-0	0	0	5	0	0.45	0.96	165.53
4	0-10-0	0	16	1	0	0.6	0.96	522.4
5	0-8-0	0	20	0	0	0.64	0.96	618.47
6	0-4-0	1	2	1	5	0.7	1.0	1225.89
7	0-10-0	12	3	0	5	0.77	0.98	522.4
8	0-7-0	14	0	0	6	0.8	1.0	325.58
9	0-2-4-0	7	6	1	3	0.68	0.98	1238.58
10	0-5-0	0	0	4	0	0.36	0.77	980.2
11	0-9-0	0	0	4	0	0.36	0.77	165.53
12	0-10-0	0	1	4	0	0.39	0.82	522.4
13	0-1-8-0	10	2	3	0	0.45	0.79	699.9
14	0-9-0	11	0	4	0	0.49	0.89	165.53
15	0-6-0	0	17	0	0	0.54	0.82	1860.03
Total Demand		61	74	32	24	Total Distance		10270.47

TABLE VI
RESULTS FOR STANDARD PROBLEMS

Data Set	Algorithm Solution			Optimal Solution	
	Total Distance	Total Trips	Time (seconds)	Total Distance	Total Trips
eil22	375.27	4	2	375	4
eil23	569.74	3	16	569	3
eil30	525.44	4	21	510	3
eil33	857.82	4	6	835	4
eil51	554.41	6	98	521	5

The algorithm can be improved by adding more factors to govern the trip merging process. More research can be conducted to improve equation 6 which is responsible for candidate selection. The algorithm can be further developed to include additional features such as dimension specific Bin Packing and Multiple Depot Vehicle Routing.

REFERENCES

[1] Toth, P., Vigo, D, *The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications.*, SIAM, Philadelphia,2001.

[2] Coleman N., Wang P., "Bin-Packing" in *Gass S.I., Fu M.C. (eds) Encyclopedia of Operations Research and Management Science*, Springer, Boston, MA,2013.

[3] Q. Zhang et al., "A Review on the Bin Packing Capacitated Vehicle Routing Problem", *Advanced Materials Research*, vol. 853, pp. 668-673, 2014.

[4] Dantzig, G. B., and J. H. Ramser. "The Truck Dispatching Problem." in *Management Science 6*, no. 1, pp. 80-91,1959.

[5] Christofides, Nicos et al. "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations", *Mathematical Programmming*, vol. 20, no. 1, pp. 255-282, 1981.

[6] L. Fisher, Marshall. "Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees.", *Operations Research*, no. 4, vol. 42, pp. 626-642, 1994.

[7] Augerat P, Belenguer J, Benavent E, Corbern A, Naddef D, Rinaldi G (1995) Computational results with a branch and cut code for the capacitated vehicle routing problem, Research Report 949-M, Universite Joseph Fourier, Grenoble, France.

[8] Lysgaard J, Letchford AN, Eglese RW (2004) A new branch-and-cut algorithm for the capacitated vehicle routing problem. Math Program 100, 42345.

[9] Fukasawa R, Longo H, Lysgaard J, Poggi de Araga o M, Reis M, Uchoa E, Werneck RF (2006) Robust branch- and-cut-and-price for the capacitated vehicle routing problem. Math Program 106, 491511.

[10] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, pp. 568581, 1964.

[11] Wang Xing and Zhao Shu-Zhi and Chu Hao and Li Yan, "An improved savings method for vehicle routing problem," in *Proceedings of 2nd International Conference on Control Science and Systems Engineering (ICCSSE)*, 2016, pp. 1-4.

[12] Donald Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Third Edition. Addison-Wesley,1997.

[13] Christofides, N., and Eilon, S.,"An algorithm for the vehicle dispatching problem", *Operational Research Quarterly*, vol. 20, no. 3, pp. 309-318,1969.