Solving Vehicle Routing Problems Using an Enhanced Clarke-Wright Algorithm: A Case Study

Buyang Cao

School of Software Engineering, Tongji University,
4800 Cao An Road, Shanghai, China 201804
buyang60@hotmail.com

Abstract. A vehicle routing problem (VRP) is an optimization problem encountered in many applications some of them even not directly related to vehicle routing. For a given fleet of vehicles (or service personnel) the goal of a VRP is to seek delivering products or services to various customer doorsteps at minimal cost (that can be represented by travel time, distances, or some customized ones) while satisfying the imposed business rules such as the vehicle capacities, the route length traversed by a vehicle, the working hours (schedules) of a driver or service person. It is known that the VRP is a difficult problem to be solved to its global optimality within a reasonable computational time. In order to solve VRPs from the real world more effectively, many algorithms, particularly heuristics, were designed and implemented to tackle this type of problems. Recently the well-known savings approach of Clarke and Wright was re-considered and some enhanced versions were proposed aiming to achieve improved solutions for the VRP. The goal of this paper is to present a business scenario requiring VRP solutions, and to propose an enhanced Clarke and Wright algorithm in the spirit of those proposed recently to solve the problems of this case study. Furthermore, the proposed algorithm aims at eliminating the human interventions such as parameter setting and tuning during the problem solving procedures. Computational results demonstrate that the new algorithm addresses the business needs better in the real applications and the results obtained by the algorithm are preferred by the end users.

Keywords: Vehicle Routing Problem, Case Study, Decision Support, Savings Algorithm.

1 Introduction

Since the vehicle routing problem (VRP) was first introduced by Dantzig and Ramser (1959), VRPs have drawn great attention from academic researchers and practitioners due to their values in theoretical research and real applications. From a practical point of view, the goal of a VRP attempts to keep distribution costs as low as possible while these distribution costs may account for a major portion of the total logistics operation costs of a company. Today a company wants to offer the best service (delivering products or services) to its customers with reduced logistics costs whenever it is possible, which raises a challenging problem due to the rapid increase in energy and labor

H. Hu et al. (Eds.): ICCL 2012, LNCS 7555, pp. 190–205, 2012.

costs. The effective solution of a VRP might produce significant economic benefits for a company. From the academic perspective, the VRP has been shown to be an NP-hard problem (Lenstra and Rinnooy Kan, 1981), which means a large VRP from the real world cannot be solved to its optimality within an acceptable computational time. Because of the value of time, a fast solution procedure for a VRP is always favorable, which attracts many scholars to invent algorithms seeking solving this difficult optimization problem efficiently.

The goal of a VRP is to decide how to deploy a fleet of vehicles or service personnel to distribute products or provide services to customers at the lowest costs where travel time, travel distance, or service quality may impact the costs of such operations. The fleet of vehicles or service personnel will start from a depot or central office and return to it after the products or services have been delivered. Some business logic to be considered include the vehicle capacities and the driver or service person's working schedule (lunch or break times). Although an extended version of a VRP might include time windows referred as *vehicle routing problems with time windows* (VRPTW), we will focus on solving the VRP in this paper.

Even though small instances of the VRP or its extended versions could be solved by exact algorithms such as branch-and-bound or branch-and-cut approaches (e.g., Ropke and Cordeau (2009), see also Bodin et al. (1983) and Braysy and Gendreau (2005) for more complete surveys on exact VRP algorithms), heuristics are preferred solution methods in solving VRPs in practice due to their capability of obtaining reasonable solutions within acceptable computational times. Therefore, almost every piece of commercial software for solving VRPs uses heuristics to obtain solutions with reasonably good quality within a short period of time.

Clarke and Wright (1964) presented a greedy heuristic that is widely known as the Clarke-Wright savings algorithm that will be named as *C-W algorithm* in the following discussion. The C-W algorithm starts with an individual route for each customer (or stop) and merges routes iteratively as long as the combination can reduce cost and meet the constraints until no further merge is possible. The C-W algorithm is relatively easy to implement and proven to be an effective approach for solving real VRPs, though the C-W algorithm is not quite suitable to solve VRPTWs. As a matter of fact, the C-W algorithm is the most popular VRP solver and has become the basis for many VRP algorithms particularly in finding reasonable good initial solutions for VRPs. Since the C-W algorithm was proposed, a lot of C-W algorithm based or enhanced methodologies have been presented in order obtain solutions with better qualities. Juan et al. (2011) proposed several solution strategies to improve the Clarke-Wright savings heuristic. In order to consider the "route shape" during the route building procedure, Paessens (1988) implemented a parameterized savings algorithm that takes the asymmetry between customer locations with respect to their distances to the depot. However, the parameters included in the savings formula need to be tuned to yield satisfactory results.

Recently, some researchers revisited C-W algorithm and attempted to invent more efficient VRP solvers. Altinel and Oncan (2005) noticed that at the late phases of the route merge of the C-W algorithm or its enhancements, the customer demands will have a greater impact on the quality of a VRP solution. They added a new item in the

cost savings formula reflecting the customer demands with an associated parameter. Unfortunately, the more parameters in the savings formula, the more time needed to tune these parameters. Furthermore, the tuned parameters may be good for one dataset but could be very bad for another one because of different characteristics (geography, vehicle capacities, working schedules, etc.). In order to facilitate the parameter tuning procedures for the parametric C-W algorithm and its enhancements, Battarra et al. (2008) proposed a genetic algorithm based methodology that is able to tune the parameters more efficiently upon the computational experiments. Subsequently Corominas et al. (2010) developed a fine-tuning procedure for a parametric C-W algorithm and its enhancement, which appears to be an effective procedure and is able to achieve desired parameters in shorter time frames. The computational results reported in the paper validate the fitness of those tuned parameters.

However, based upon our project experiences in solving real VRPs, it is hard to obtain an "ideal" parameter set for a parametric C-W algorithm and its enhancements not to mention applying this tuned parameter set universally to all VRP problems and expecting overall better results. Furthermore, the end users of VRP software usually do not have any background in Operations Research; it is very difficult for them to tune the algorithm parameters; not to mention that the parameter tuning is very tedious. We believe it might be more crucial to enhance the C-W algorithm structurally, i.e., adding new steps and logic, rather than merely parameter tuning. In this paper we present an algorithm that is capable to solve VPRs relatively efficiently without too much user intervention and a case study for it, which are the motivation of this paper.

This paper is organized as follows. In the next section, we provide a background of VRPs encountered and to be solved. Then we present the details of the enhanced C-W algorithm. Section 4 contains computational experiments using real application datasets to demonstrate the effectiveness of the proposed algorithm. The paper ends with concluding remarks and directions for future studies.

2 Background

The operations research literature documents rich applications of VRPs. Some examples include Weigel and Cao (1999) for delivering furniture, appliances, and related services to customer homes; Kim et al. (2006) for solving VRPs in a waste collection operation and Spada et al. (2005) for scheduling student bus service. The interested readers are referred to Bozkaya et al. (2011) for more details on these and other interesting VRP applications. The following real problem needs to be solved.

The logistics department of a large consumer product company is responsible for delivering their products daily to their authorized stores where the products will be sold. Each store owner will make his order online according to the expected sales. The day before the delivery is conducted; the logistics department gathers the following information in order to make the delivery decision for the next day's operation:

- the available vehicles and their capacities, and they may not be homogenous
- the drivers' working schedule
- the locations of those customers to be delivered and the demands
- the service (delivery) time at each customer location, which varies upon the demand

Based upon the business needs, the objective of the VRP to be solved is to minimize the total travel time of all routes. On average the logistics department needs to deliver around 1000 customers with approximately 30 vehicles every day. In addition, the logistics department must take the following business logic into account while building the delivery routes:

- Provide delivery truck drivers with consistent routes.
- Stores on the same street segment should be delivered by the same truck whenever it is possible.
- The side of a street to unload products needs to be considered.
- Whenever a truck encounters a store of its route on the way out, it should deliver that store immediately if possible to reduce the weight loaded on the truck; it is referred to as "first-see first-serve" rule.
- Larger trucks should be utilized first.
- Whenever it is possible, one area where several stores are located should be delivered by the same vehicle.

For a more comprehensive mathematical model of VRPs, interested readers are referred to Cao and Bozkaya (2012). It is interesting to note that some of these business rules are very difficult to be presented "mathematically". Besides the business rules listed above, the geographic characteristic may be challenging as well. One of the delivery or service areas is a mountain area; Figure 1 illustrates the portion of the service area and some customer locations.

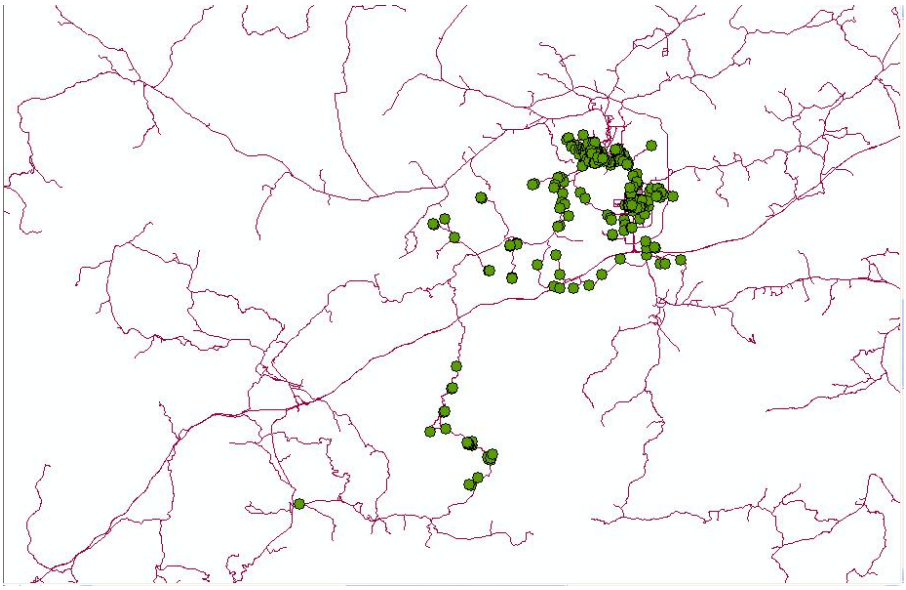

Fig. 1. Delivery area

From Figure 1 it is not difficult to recognize that the street network structure is similar to a "tree" structure because of the nature of mountain roads. Furthermore, some stores are located at some branches of this street network. In this case, there are not so many alternative paths from a depot to a store that is located at a branch. It is conceivable that if a store at one branch is serviced by a vehicle other than the one serving nearby branches, then the majority of the time these two vehicles will traverse the same road segments. If one area is serviced by more than one vehicle, then some vehicle might have to drive extra miles to service the store located on a branch of the street network. We name this scenario as *route crossing over*. From the operational perspective route crossing over is not efficient and that cannot satisfy the business rule listed above stating one area should be serviced by a single vehicle whenever it is possible. Figure 2 presents some route crossing over, which should be avoided whenever it is possible (the number next to each stop is the corresponding route ID).

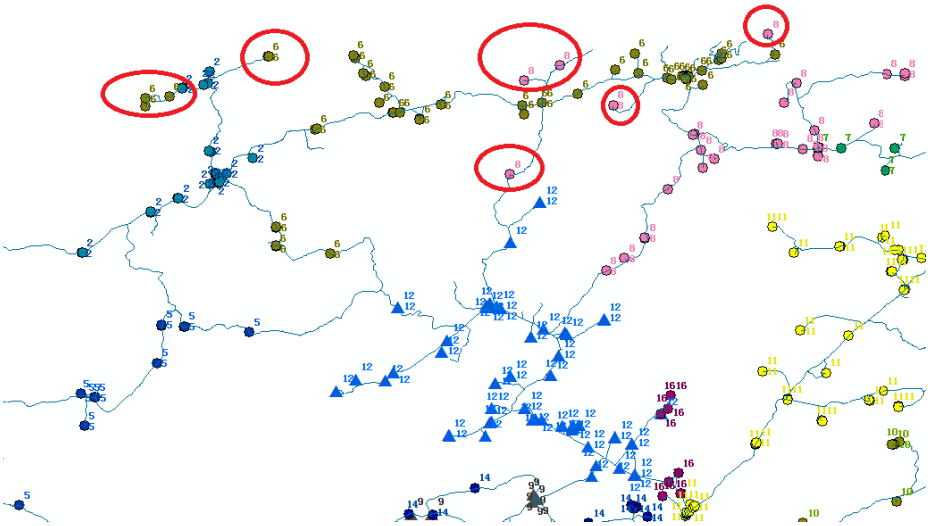

Fig. 2. Route Crossing Over

Those stops inside of red circles are serviced by vehicles that are different from the nearby ones. In order to service these stops more than one vehicle has to travel the same street segments, and some of them have to travel extra time and distance in order to reach those stops. If those stops are far away from the depot, the negative economic impact is severe. This result usually encounters the resistance at the field during the deployment because of its relatively poor performance in the execution.

In order to eliminate the route crossing over as much as possible, we need to pay attention during the route construction phase. If the route crossing over exists after the route construction, it would be very hard to be removed no matter how the improvement steps are applied according to our practical experience. Because of multiple constraints such as time, capacities, etc. it is nearly impossible to utilize improvement steps like transfer/exchange moves to get rid of route crossing over even when metaheuristics are employed. We need to point out that the system we attempted to deploy

for this customer has been used by other clients, and the core solver is a meta-heuristics based one which has been evolving over years.

According to the paper of Braysy and Gendreau (2005), there are basically two types of route construction frameworks though there are a number of variants. One is the C-W algorithm that will be discussed further in the following sections and the other is the insertion based algorithm proposed by Solomon (1987). The insertion algorithm starts with a route including a "seed" stop, and sequentially inserts unrouted stops into the route. It restarts a new route when the current route is full (either in terms of time limit of capacity limit). Each time the algorithm inserts a stop with the lowest insertion cost, where the meaning of a cost can be general enough to consider travel time/distance, time window violation, etc. The procedure will repeat until all stops are routed or no route is available for routing. It is not hard to recognize that the insertion framework is able to handle temporal dimension fairly effectively, it has been widely applied to solve VRPTWs (e.g., Bozkaya et.al (2011), Kim et.al (2006), Weigel and Cao (1999)). Nevertheless, the insertion algorithm emphasizes the insertion cost impacted mainly by two neighboring stops at the moment of an unrouted stop to be evaluated for insertion. Unlike the C-W algorithm, the insertion algorithm takes less consideration of the penalty of missing an insertion opportunity for an unrouted stop. Thus, another vehicle may have to travel long time/distance to service this unrouted stop eventually. Actually we experienced some cases where route crossing over occurred at the beginning of the project of building the routing application for this consumer product company as we employed the insertion framework to construct routes.

After careful analyses of the problems, it turns out that we do not need to consider time windows because the stores do not have the imposed time window and usually they can receive the delivery within the business hour of the company. We can take advantage of the C-W algorithm that not only considers the neighboring travel time/distance but also the travel time/distance between unrouted stop and depot during the route construction. We are going to discuss in detail an enhanced C-W algorithm that is able to eliminate the route crossing over at great degree and create more satisfactory results in the next section where the time window is no more a consideration.

3 Enhanced C-W Algorithm

For the purpose of completeness, we discuss the C-W algorithm and its variants briefly here. In a VRP there is a depot d and a set of customer locations $L = (1,\ldots,n)$, where the vehicles start from d and service a subset of L and return to d. The C-W algorithm start with each location or stop being serviced by a vehicle and gradually merges routes until there is no possibility of merging. If we want to use a single vehicle to serve two stops, say i and j, on a single trip, then total travel time/distance is reduced by:

$$s_{ij} = (c_{di} + c_{id} + c_{dj} + c_{jd}) - (c_{id} + c_{ij} + c_{dj}) = (c_{id} + c_{dj} - c_{ij}) \quad (1)$$

where c_{ij} represents the travel distance/time from location i to location j. It is desirable to service two locations that have the larger savings value defined by (1) under the

condition that no imposed constraints (capacity, working hour, etc.) will be violated due to the merge. At the beginning of the solution procedure, the list containing savings values is created and sorted in the non-increasing order. At each iteration a pair of stops *(i,j)*, actually a link, is picked at the top of the list, and it considers:

- If both stops are not routed (assigned), then a new route serving these two stops is created. Or,
- If one of the stops is routed, and it is a non-interior stop (a routed stop is called non-interior stop if it is adjacent to the depot in the route), then the unrouted stop may be inserted to the same route if no constraints are violated. Or,
- Both stops are routed, and they are all non-interior stops, then these two routes may be combined if no constraints are violated.

If the saving list is exhausted, the algorithm terminates. Note that at the end some routes may contain a single stop.

Altinel and Oncan (2005) believed that the customer demands impact the overall solution quality, and they proposed the following new savings value:

$$s_{ij} = c_{id} + c_{dj} - \lambda c_{ij} + \mu \left| c_{di} - c_{jd} \right| + \nu \bar{r} \tag{2}$$

where $\bar{r}$ is defined by: $\frac{r_i + r_j}{r'}$

Furthermore, r_i is the demand for customer i, r' is the average demand. The customer spatial distributions ($\left| c_{di} - c_{jd} \right|$) are taken into account to a certain degree. There are some parameters in the savings value. It is inevitable that these parameters must be tuned in order to achieve satisfactory results. Upon the results reported in the paper, the savings value (2) does obtain better solutions than the native C-W algorithm does after the parameters are carefully tuned.

As we pointed out earlier, based upon our real project experience merely the parameter tuning at times is unable to produce desired results (e.g., route crossing over still occurs) and the parameter tuning is a tedious task that cannot be performed by the end users. We are going to enhance C-W algorithm structurally by adding solution steps to be applied to any VRP without tuning parameters and conduct a case study in which this proposed algorithm is applied. Because in practice there are traversing restrictions for street networks including one-way, divided roads, we assume that the underlying street network is not symmetric, i.e.: we may have $c_{ij} \neq c_{ji}$. The following subsections describe the enhancements for the native C-W algorithm.

3.1 Enhancement on Stop Insertion

In the native C-W algorithm, if one of the pair of stops is not routed or assigned, then this unrouted stop can be inserted to the same route where the other stop is located only if it is a non-interior stop of that route. However, we observed that if this condition must be held to decide if an unrouted stop can be inserted to an existing route, we might miss some opportunities to produce relatively good results. Therefore, when a pair of stops is being evaluated for route assignment, an unrouted stop can be inserted right after (or before) the other stop that has been assigned to the route as long as no constraint is violated. Specifically:

Let $p = (d,\ldots,h,i,k,\ldots,d)$ be an existing route and (i,j) be the pair of stops being evaluated for possible assignment (insertion). If there is no constraint being violated, then the resultant route will be: $p = (d,\ldots,h,i,j,k,\ldots,d)$.

After this strategy is implemented, we find that there are very few routes containing only one real stop comparing to the native C-W algorithm. In this case, we can save the number of vehicles to be used, which in turn saves the operational costs. The combination of native C-W algorithm and this step is called *stop assignment procedure* in the following discussions.

3.2 Enhancement on Route Improvements

The stop assignment procedure presented above alone cannot generate very satisfactory results. The route improvement steps are necessary to bring us solutions with better qualities. We are also going to employ the savings value list or the information gathered during the creation of the savings value list to guide the improvement process. The following route improvement steps are proposed, which are similar to those inter-route improvements presented in Bozkaya et al. (2011) with additional updates:

a) Stop transfer move: although the application of the C-W algorithm is able to avoid some route crossing over mentioned in the previous section, the results at times contain route crossing over. Regarding the business practice, a route crossing over is usually inefficient and it is desirable to remove any route crossing over whenever it is possible. To this purpose, we create a neighboring list holding neighboring stops for each stop while the saving value list is being created. This neighboring list contains certain number of closest stops (in terms of real travel times/distances) to the underlying one and is sorted in non-decreasing order of travel time/distance. With this neighboring list it is relatively easy to identify the possible route crossing over or the inefficiency of a solution. Fig. 3 illustrates the scenario.

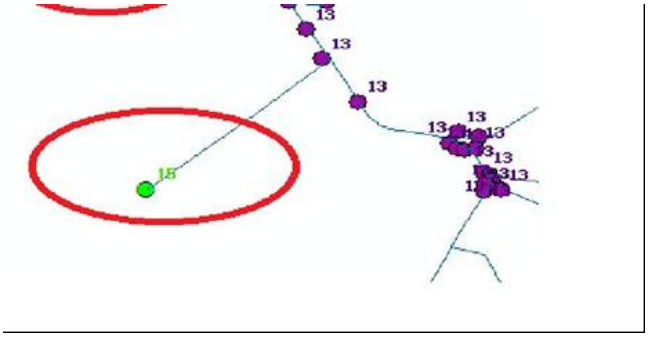

Fig. 3. Possible route crossing over

In Fig. 3 the numbers besides stops represent the route ID. It is not difficult to see, that routes 13 and 18 are crossing since the neighbors of a stop in route 18 are all in route 13. Route 18 has to traverse a lot of street segments

that are also traveled by route 13. It is conceivable that it could be more efficient if this stop can be serviced by the route where its neighbors are located. In this example, route 13 will be an ideal route to service the stop currently serviced by route 18. Furthermore, even if the route crossing over cannot be removed completely, we want a stop and its neighboring stops be serviced by the same route whenever it is possible. Here we will perform the improvement move (step) called stop transfer to reach this goal. In order to speed up the entire process, we consider performing the possible stop transfer move for a stop if its first and second neighbors are not at the same route as it does, and the first and second neighbors are serviced by the same route.
Definitions:

- Transfer candidate: a stop that may be transferred
- Source route: the route services a transfer candidate
- Target route: the route can potentially service a transfer candidate, where the first and second neighbors of the transfer candidate are located

Let i be the stop index for the transfer candidate, $ps = (d,\ldots,i-1,i,i+1,\ldots,d)$ be the source route, and $pt = (d,\ldots,j-1,j,\ldots,d)$ be the target route, respectively. The cost for the potential transfer move is determined by (here we will insert the candidate only right after or directly before its first neighbor denoted by $j-1$in the target route):

$$\Delta_t = c_{i-1,i+1} + c_{j-1,i} + c_{i,i+1} - c_{i-1,i} - c_{i,i+1} - c_{j-1,j} \tag{3}$$

This is the scenario for the candidate being inserted right after its first neighbor; and we can have the similar equation for the case where the candidate is inserted directly before its first neighbor. For each candidate both possible moves are evaluated, and the move with the lowest transfer cost is picked, which is called possible transfer move. Furthermore, for a possible transfer move, the constraints will be checked. If a possible transfer move meets all the constraints and possesses a negative transfer cost, it is a valid transfer move.

At each iteration of the stop transfer move, it picks the valid transfer move that has the lowest transfer cost among all valid ones and updates the corresponding routes after the move is carried out.

The stop transfer repeats these steps described above until no further valid transfer move is found.

b) Route interchange move: it is often that a transfer move cannot be carried out due to the constraint (time, capacity, etc.) violation of a target route. It is not enough to apply merely transfer move for better solutions because the capability of exploring solution space of a stop transfer move is limited. Here we propose a new improvement step called route interchange move, which is described as follows.

An element (a pair of stops) in the saving value list with positive value is called an interchange candidate if they are serviced by two different routes.

According to the principle of the C-W algorithm, it could be beneficial to service two stops of an interchange candidate by the same route. Let *(i, j)* be the link or stop pair of an interchange candidate, *p1 = (d…,i–1,i,i+1,…, d)* and *p2 = (d,…,j–1, j, j+1,…,d)* be the routes serving stop *i* and stop *j*, respectively. After the route interchange move is performed, these two routes become: *p1* = (d…,i–1,i,j,j+1,…, d)* and *p2* = (d,…,j–1,i+1,…,d)*, and Figure 4 illustrates the move.

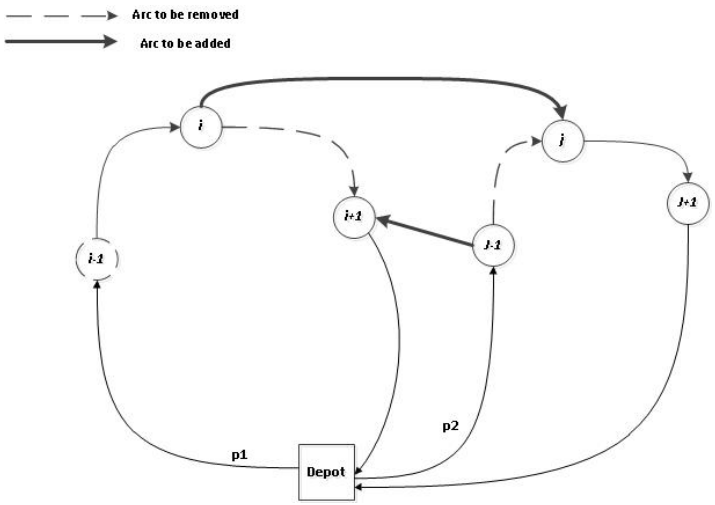

Fig. 4. Route Interchange move

Obviously for each interchange candidate, the corresponding interchange cost is determined by:

$$\Delta_{ri} = c_{ij} + c_{j\text{-}1,i+1} - c_{i,i+1} - c_{j\text{-}1,j} \tag{4}$$

If an interchange candidate has the negative interchange cost and it meets all imposed constraints, then it is called a valid route interchange move. Unlike the stop transfer move, the move will be carried out immediately as soon as a valid route interchange move is found and two corresponding routes are updated. Because of using the savings value list, the solution space to be searched is narrowed and the savings value list provides the promising candidates for the route interchange move.

The route interchange move scans the savings value list until no valid candidate is found.

3.3 Enhancement on Route Balance

One of the weaknesses of the C-W algorithm is the route balancing consideration. It is hard to apply the native algorithm to consider the route balancing issue explicitly. Unlike the insertion based framework for the route construction where the routes

usually are full and can be built relatively well balanced, the results obtained by the native C-W algorithm contain unbalanced routes, some routes may work until the end of the day while some routes have only a couple of working hours. The solution with unbalanced routes is very difficult to deploy in practice, and it should be fixed. The route balancing procedure described below is invented to reduce the imbalance in resultant routes.

The basic idea of the route balancing procedure is to remove those half-full routes and reassign the stops serviced by these half-full routes to other ones. It employs the methodology called "destroy and re-build" proposed by Cao and Bozkaya (2012), which is also similar to the methods suggested by Glover (2000) and Laporte et al. (2010). All the stops removed from those half-full routes will be treated as unrouted stops and they will be reassigned to certain routes. Based on the solution, we will empty all routes that are working only 40% of their stated working hours and all stops currently serviced by these routes will become unrouted. The stop assignment and route improvement procedures discussed above will be applied again to reassign these unrouted stops.

Furthermore, we have the knowledge of the number of vehicles to handle the customer demand; therefore we can limit the number of vehicles to be used for building new routes (according to C-W algorithms, if a pair of unrouted stops picked from the savings value list, then a new route will be opened to service these two stops). The purpose of this consideration is to build all routes as full as possible and to eliminate the route imbalance. The number (n) of vehicles used for opening new routes after those half-full routes are emptied is defined by:

$$n = \begin{cases} 1; if\ the\ number\ of\ emptied\ route\ is\ 1 \\ \lceil (number\ of\ emptied\ routes)/2 \rceil; otherwise \end{cases} \quad (5)$$

where $\lceil x \rceil$ is the integer that is less or equal to number x.

If we cannot find any route to be emptied, then we believe that the current solution is relatively balanced or most routes are packed, and this step is not necessary to be carried out.

Summary of the enhanced C-W algorithm:

Based on the above discussions we now summarize the enhanced routing algorithm as follows.

Initialization:

Building OD matrix and saving value list.

Main procedure of the algorithm:

Done = false;

Iter (recording the number of iterations) = 0;

While not Done

- Perform stop assignment procedure;
- Perform stop transfer;
- Perform route interchange;
- Iter = Iter + 1;
- If Iter > predefined number of iterations to be performed then
  Done = true;

Else

Get number of routes to be emptied, *n'*. If *n' > 0*, set all stops serviced by these routes to be unrouted.

End if

End While

Sequencing all routes:

For each route in route set do

Perform intra-route improvement (Bozkaya et al. (2011));

End For

Although the stop assignment procedure is included in the loop, it will not be performed if all stops are assigned. If some routes are emptied because they are half-full, certain stops are unrouted. In this case, the stop assignment procedure attempts to reassign these unrouted stops to proper routes with the limited number of vehicles for opening new routes. The number of iterations for the loop in the algorithm is set between 10 and 20 upon the size of a problem.

4 Computational Experiments

We use real datasets to conduct the computational experiment for this case study. The purposes of this experiment are:

- Comparing to the basic C-W algorithm to demonstrate the capability of producing better solutions of the proposed algorithm in this paper, and
- Comparing to the insertion-based algorithm to demonstrate the ability of creating solutions containing non- or nearly non-crossing over routes. As mentioned above, this insertion-based algorithm is not a simple implementation, it has been used by various customers and proven to achieve economic benefits in the past (see Weigel and Cao (1999), Bozkaya et al. (2011) Cao and Bozkaya (2012)).

The data were collected from the real applications, and the delivery area is shown in Figure 1 that contains about 12000 street segments. All computational experiments were conducted on a laptop whose configuration is: CPU - Intel Core 2 Duo P8400 2.26GHz, memory size - 4G. The operating system is Windows 7. The problem sizes vary from 200 stops (customers to be serviced) to 1600 stops. The computational results are listed in Table 1.

In Table 1 we list the results obtained by three algorithms including the algorithm presented in the paper (Our Solver), the basic C-W algorithm (Basic C-W Savings Solver), and the insertion based algorithm (Insertion Based Solver). The results yielded by these three algorithms contain the total travel time in minutes traversed by all routes, the total travel distance in kilometers traversed by all routes, the number of routes to be used, and the CPU time in seconds required to solve a problem. Figure 5 depicts the results (total travel time for each problem).

Table 1. Computational Results (#R: # Routes)

prob-lem size	Our Solver				Basic C-W Savings Solver				Insertion based Solver			
	CPU time (sec.)	#R	total length (km.)	total travel time (min.)	CPU time (sec.)	#R	total length (km.)	total travel time (min.)	CPU time (sec.)	#R	total length (km.)	total travel time (min.)
200	1	5	667.8	1124	<1	5	683.3	1153	2	5	722.0	1227
280	1	6	832.1	1460	1	7	874.71	1513	3	6	884.8	1535
350	2	8	1155.9	1879	1	8	1056.5	1883	5	8	1215.8	2142
420	3	9	1094.1	1957	3	9	1140	2008	8	9	1300.1	2272
490	4	10	1210.3	2187	3	10	1252.2	2233	11	10	1415.3	2548
560	4	10	1305.2	2357	2	11	1326.9	2400	13	10	1472.6	2678
630	6	11	1405.5	2544	5	13	1450.7	2596	16	11	1509.6	2738
700	8	13	1574.9	2815	5	13	1555.2	2791	26	12	1603.1	2928
780	10	14	1679.6	2998	5	15	1692.7	3021	25	14	1761.5	3186
850	9	15	1798.4	3183	4	15	1816.6	3224	24	15	1929.6	3403
920	11	16	1940.9	3421	6	19	2113.2	3672	29	16	2114.0	3731
1000	12	17	2133.1	3692	5	21	2311.5	3969	32	17	2168.5	3798
1200	13	19	2257.5	3912	10	21	2330.7	4025	47	19	2359.2	4087
1400	15	23	2446.3	4219	10	25	2675.4	4548	63	20	2418.6	4224
1600	28	23	2467.0	4278	12	27	2735.9	4719	81	24	2558.8	4442

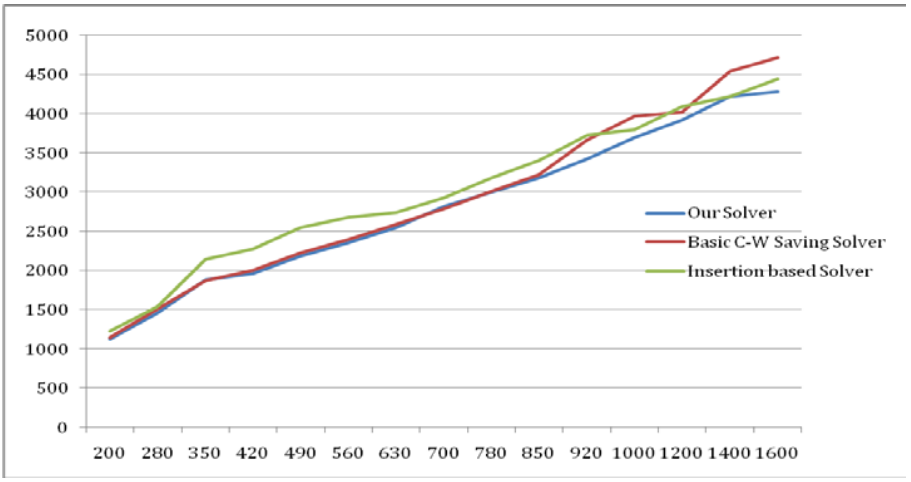

Fig. 5. Computational Results (abscissa: problem size; ordinate: total travel time)

It is not difficult to recognize that the algorithm presented in this paper obtains overall best solutions upon the business logic (minimum total travel time). It is possible that the travel time is shorter but the travel distance is longer. In reality, traveling on a freeway takes shorter time but it may traverse longer in distance compared to traveling on local streets. There is no need to tune the parameters for the algorithm at all. The native C-W algorithm produces consistently more routes than the other two algorithms, because it won't merge two routes if one stop of a pair of stops in the savings value list is not a non-interior node of a route. The enhanced algorithm has the additional consideration to overcome this problem while the insertion based algorithm always makes a vehicle full before a new vehicle is used. Actually using fewer vehicles to handle the same amount of products to be delivered will not only save money but also work more efficiently. Compared to the insertion based algorithm the proposed algorithm is able to solve the problems in shorter computational times and obtain better solutions in this case study. The insertion based algorithm needs longer computational times since it needs to re-evaluate the insertion position of an unrouted stop at each iteration during the assignment procedure (though some implementation strategies can be taken to shorten the time a bit). However, the proposed algorithm scans the sorted list and does not need to perform a complicated evaluation procedure. Furthermore, as we mentioned above, there is no parameter for the solver, and we do not need to conduct comprehensive parameter tuning process. In this case the solver is more suitable for the real applications since an end user usually does not have any idea how to tune the algorithm parameters to fit his problems. The proposed algorithm in this case study provides a very solid alternative for solving VRPs in practice.

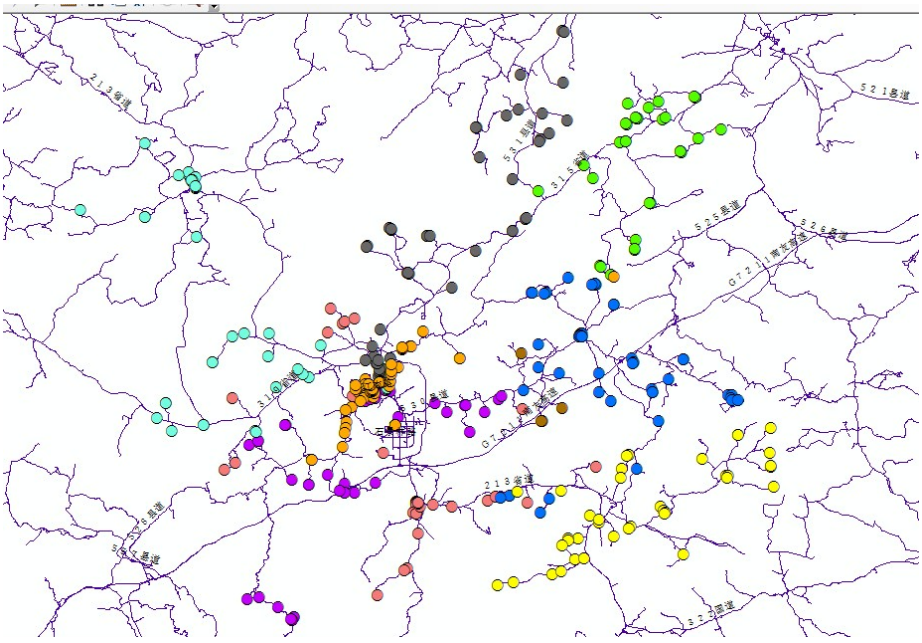

Fig. 6. Results obtained by the insertion based algorithm

Figures 6 and 7 present the results obtained by the insertion based algorithm and the proposed algorithm, respectively. Obviously the routes created by the proposed algorithm have clean route boundaries and have little crossing over while the routes generated by the insertion based algorithm have more crossing over. The end users favor the results created by the proposed algorithm.

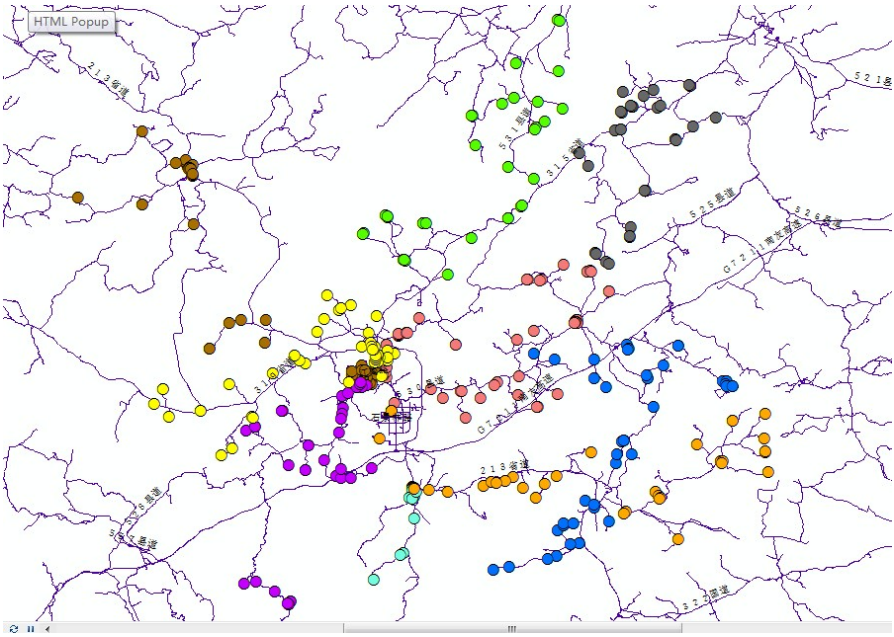

Fig. 7. Results obtained by the proposed algorithm

5 Conclusions

In this paper, we have presented an enhanced Clarke-Wright algorithm. By introducing enhanced steps in route building, stop assignments, and route balancing, the algorithm is able to obtain superior results compared to the basic C-W algorithm and an insertion based algorithm. Unlike parametric C-W algorithms for which parameter tuning is necessary, we do not require any parameter tuning, which facilitates the algorithm deployment for real applications. The algorithm is used in several application systems to solve VRPs for different clients as reflected in a case study.

In our future work we are planning to enhance the algorithm in order to deal with VRPs where time windows are imposed so that the proposed algorithm can be applied to additional applications. Furthermore, we plan to incorporate the approach into some meta-heuristics framework.

Acknowledgements. The author would like to express his gratitude to three anonymous referees for their critics and very useful suggestions to improve the manuscript.

References

1. Altinel, I.K., Oncan, T.: A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. Journal of the Operational Research Society 56, 954–961 (2005)
2. Battarra, M., Golden, B., Vigo, D.: Tuning a parametric Clarke-Wright heuristic via a genetic algorithm. Journal of the Operational Research Society 59, 1568–1572 (2008)
3. Bodin, L., Golden, B., Assad, A., Ball, M.: Routing and scheduling of vehicles and crews: the state of the art. Computers & Operations Research 10, 63–212 (1983)
4. Bozkaya, B., Cao, B., Aktolug, K.: Routing solutions for the service industry. In: Montoya-Torres, J.R., Juan, A.A., Huatuco, L.H., Rodriguez-Verjan, G.L. (eds.) Hybrid Algorithms for Service, Computing and Manufacturing Systems: Routing and Scheduling Solutions, pp. 46–78. IGI-Global Publishing (2011), doi:10.4018/978-1-61350-086-6
5. Braysy, O., Gendreau, M.: Vehicle routing problem with time windows, Part I: Routing construction and local search algorithms. Transportation Science 39(1), 104–118 (2005)
6. Braysy, O., Gendreau, M.: Vehicle routing problem with time windows, Part II: Metaheuristics. Transportation Science 39(1), 119–139 (2005)
7. Cao, B., Bozkaya, B.: Vehicle routing in service industry using decision support systems. In: Faulin, J., Juan, A.A., Grasman, S.E., Fry, M.J. (eds.) To Appear in Decision Making in Service Industries: A Practical Approach. Taylor & Francis (2012)
8. Clarke, G., Wright, J.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581 (1964)
9. Corominas, A., Garcia-Villoria, A., Pastor, R.: Fine-tuning a parametric Clarke and Wright heuristic by means of EAGH (empirically adjusted greedy heuristics). Journal of Operational Research Society 61, 1309–1314 (2010)
10. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management Science 6, 80–91 (1959)
11. Glover, F.: Multi-start and strategic oscillation methods – Principles to exploit adaptive memory. In: Laguna, M., Gonzales-Valarde, J.L. (eds.) Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, pp. 1–24. Kluwer (2000)
12. Juan, A., Faulin, J., Jorba, J., Riera, D., Masip, D., Barrios, B.: On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright savings heuristics. Journal of the Operational Research Society 62, 1085–1097 (2011)
13. Kim, B.-I., Kim, S., Sahoo, S.: Waste collection vehicle routing problem with time windows. Computers & Operations Research 33, 3624–3642 (2006)
14. Laporte, G., Musmanno, R., Vocaturo, F.: An adaptive large neighborhood search heuristic for the capacitated arc-routing problem with stochastic demands. Transportation Science 44(1), 125–135 (2010)
15. Lenstra, J., Rinnooy Kan, R.: Complexity of vehicle routing and scheduling problems. Networks 11, 221–227 (1981)
16. Paessens, H.: The savings algorithm for the vehicle routing problem. European Journal of Operational Research 34, 336–344 (1988)
17. Ropke, S., Cordeau, J.-F.: Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science 43, 267–286 (2009)
18. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265 (1987)
19. Weigel, D., Cao, B.: Applying GIS and OR techniques to solve Sears technician dispatching and home delivery problems. Interfaces 29(1), 112–130 (1999)