# A Task Based Approach for A Real-World Commodity Routing Problem

Jianjun Chen*, Ruibin Bai†, Rong Qu‡ and Graham Kendall§
* Division of Computer Science, University of Nottingham Ningbo, China
Email: jianjun.chen@nottingham.edu.cn
† Division of Computer Science, University of Nottingham Ningbo, China
Email: ruibin.bai@nottingham.edu.cn
‡ School of Computer Science, University of Nottingham
Email: rxq@cs.nott.ac.uk
§ School of Computer Science, University of Nottingham, Nottingham, UK and Semenyih, Malaysia
Email: graham.kendall@nottingham.ac.uk

***Abstract*—In this paper, a real world short-haul commodity routing problem is presented. This problem shares several similarities with vehicle routing problem with time windows (VRPTW) and the service network design problem (SNDP), but also has its own specific structures that do not exist in VRPTW or SNDP. A task based formulation is developed for this problem and a variable neighbourhood search metaheuristic approach is proposed, resulting in a visible improvement over the original routing plans according to experimental tests over three real-life instances. Apart from introducing a new real-world commodity routing problem, another main contribution of this paper is a task based formulation that allows commodity flows being considered as nodes in a routing network. Thus algorithms that were designed for VRPTW or SDVRP can also possibly be adapted to solve this commodity flow problem.**

## I. Introduction

Freight transportation is an important aspect of business for transportation firms. The efficiency and quality of service is especially crucial for their survival and profitability in a competitive market. For the problems in this area, the objective is usually described as providing high standard services while maintaining a consistent performance or running at the lowest cost at a specified service level. As traffic in urban areas, or even rural areas, becomes increasingly demanding and more complicated, companies are facing more challenges than ever before to maintain or improve the level of freight services due to traffic uncertainties. Another important issue is the environmental impact from modern transportation. High transport efficiency could help to reduce the use of fossil fuel. Such perspectives have attracted many researchers to explore this area by using many different approaches.

In this paper, a real world short-haul routing problem is introduced. This new problem shares several characteristics of vehicle routing problem with time windows (VRPTW) [1] and service network design problem (SNDP) [2]. All these problems involve transporting commodities or services over a network with geographically distributed nodes. However, there are some features that differentiate this problem from the classic VRPTW and SNDP. The first main difference is the "servicing time" due to the short-haul nature of this new problem. The commodity loading/unloading time in this problem is comparable to the transportation time and hence cannot be ignored or combined in a similar way as in the service network design problem or VRPTW. Secondly, the commodity flows in SNDP can be continuous but in this new problem they are integers (measured by the number of containers). Finally, transportation demands in this new problem can be between any ports but no consolidation is generally permitted since the volume of unit commodity is comparable to the capacity of vehicles. This is very different from conventional VRP where commodities are shipped from one or more depots to customers. It is also different from SNDP in which path sharing and consolidation are widely used mechanisms to improve efficiency.

The paper is structured as follows: Section II reviews relevant work, mainly focusing on vehicle routing problems and service network design problems. In section III, the new commodity routing problem is presented and a task based formulation is provided. The proposed solution method for this problem is then given in section IV and its computational results are discussed in section V. Finally section VI concludes this paper.

## II. Literature Review

### A. Vehicle Routing Problem

The vehicle routing problem (VRP) involves designing a vehicle routing plan to service all customers over a network. These customers, located in geographically different locations, are visited exactly once. The routes typically start from a special node called *depot*, pass through a set of arcs then return back to the depot after all assigned customers are serviced. Each route is associated with one vehicle schedule. The vehicle routing problem was originally proposed as the truck dispatching problem [3]. Later, several variants of VRP were introduced and the topic as a whole has been intensively researched over the past few decades. Reviews and recent developments of VRP can be found in [4], [5] and [6]. Related algorithms are also covered by [7] in detail. There are many VRP variants proposed later. For some common variants, one

can read the book by Toth and Vigo [1] for more information. Among these variants, the VRP with split delivery (SDVRP) and VRP with time windows (VRPTW) share most similarities with our problem.

VRPTW is an extension of VRP with additional time window constraints. Solomon [8] proposed several classic heuristics for VRPTW. However, the results from these heuristics are inferior when compared with modern approaches in terms of solution quality. This is understandable as the computational power was low at that time and iterative metaheuristic approaches are computationally more expensive. Taillard et al. [9] introduced an efficient tabu search for the problem. An adaptive memory is used to improve the quality of solutions. The adaptive memory contains the routes of the best solutions that have previously been explored. Chiang and Russel [10] proposed a reactive tabu search that adjusts the tabu list during the search process. It increases the size of the tabu list based on whether the same solution is visited multiple times during a period. If no feasible solution is found during several iterations, the size of the tabu list is decreased. Several local search algorithms and neighbourhood functions are covered in [11]. El-Sherbeny [12] also reviewed recent algorithms for VRPTW.

SDVRP was introduced in 1989 by Moshe Dror and Pierre Trudeau [13]. The idea is that transportation cost can be reduced by splitting the commodity such that they can be transported by several vehicles. Dror et al. [14] proposed an integer programming model and a branch and bound algorithm for the problem. Cordeau et al. [15] reviewed SDVRP and some related problems with related algorithms. Recent development of SDVRP can be found in [16].

### B. Service Network Design Problem

The service network design problem is usually used to solve problems at strategic and tactical levels. The service network design problem differs from the fixed-charge multi-commodity routing problem in that service assets are required to stay balanced during every period of the planning horizon. In a service network design problem, each node in the problem has a predictable service demand in each period. Thus, generating a cost-effective, asset-balanced routing plan that can cope with demand across the entire planning horizon is important. Early models and algorithmic frameworks can be found in [17], [18] and [19]. Crainic [2] presented an excellent review of this problem, especially with the modelling and mathematical programming development efforts. The review focuses on long-haul transportation. Andersen et al. [20] discussed the coordination of multiple fleets in the service network design problem. Various modelling approaches are also compared. A time-dependent, real-world sized SNDP with stochastic demand was investigated by [21]. Bai et al. [22] explored several guided local search (GLS) approaches for SNDP, an investigation was carried out to find out the mechanisms that can potentially speed up the GLS algorithm, resulting in a more efficient tabu assisted multi-start GLS algorithm.

## III. Problem Definition and Model

This section describes the problem addressed in this paper. It is an increasingly challenging problem faced by a large international port in Ningbo, China.

### A. Description of the Problem

The Ningbo Port is the 5th largest port in the world. The problem concerned in this paper is regarding to the operations of container transshipments between nine different ports managed by the company. Figure 1 shows the locations of these ports. The company currently has a fleet of 30 trucks, each having the same configuration. There is a central depot for all trucks that is quite close to port "BCLT2". The drivers are split into two shifts per day. Each shift lasts at most 12 hours. The day shift starts at 8am and the night shift starts at 8pm. Before the end of the shift, all trucks are required to return to the depot to prepare for the next shift. On average, there are about 200-300 containers to be relayed by the fleet every day. The commodity information is stored in digital declaration forms. They define the source port, destination port, available time, deadline, how many containers to be transported and their sizes. Historical data shows that most commodities' deadlines are during the day shift.


Fig. 1. Map of ports at Ningbo Port, China

Apart from the 30 trucks directly managed by the company, there are some other trucks available for use if they have finished their tasks. However, this set of trucks' available time and quantity is unpredictable for each day. Thus, it is not included as part of the fleet. In addition, there is, on average, one truck that needs repairing or maintenance everyday. This leaves 29 trucks available at any time. Each truck can take either two small containers or one large container. However, a small but heavy container requires a dedicated truck.

The trucks are equipped with GPS tracking devices. However, the GPS information is not fully integrated with the current routing software developed by the company. The planning of their truck routes is carried out by a central server that communicates with drivers through mobile phone text messages. A driver sends a message to request a task. After finishing this task, the driver sends a "task finished" message to the server, and a new task is automatically assigned

to this driver. This seems to be a reasonably good solution. However, drivers may send the finish message a bit earlier so that he can get the next task without delaying. This makes the information recording inaccurate. If the data collection is based on the text message records, it will be very hard to know the exact time when a driver starts his task because the time of receiving message is not necessarily the time of starting the task. The driver may have to *deadhead* (travelling without loading anything) to the requested port first to start the task if the destination port of the previous task is not at the same port as the starting port of the new task. Sometimes there is also traffic congestion that prolongs the travel time. Thus, all data such as travel distances, travel time, loading and unloading time are collected by combining data from GPS and estimations from drivers.

### *B. Current Software and The Company's Challenge*

The current routing software adopts two routing heuristics to rank shipment tasks. The first heuristic tries to reduce empty loads by matching the connecting ports between tasks. Thus, the empty loading distance hopefully can be reduced. The second heuristic is designed for emergent situations where the deadlines of commodities are close. Thus tasks are assigned to trucks according to the closeness of deadlines. Some less-urgent tasks are postponed. Both routing heuristics are used in real-time routing. A new task is allocated to a truck which becomes available after its current task is completed. Tasks are assigned one by one. No initial routing plan is given for a shift at first. Sometimes, manual scheduling is required to improve the performance.

The historical results shows that this method of working has low efficiency and is comparable to the performance of manual schedules. The average number of trucks with heavy loads is below 70 percent.

To meet the growing service demands and the introduction of other relay services, the Ningbo Port company is trying to increase the efficiency and maximise throughput. This will reduce the resources used in the future, saving a lot of cost while at the same time contributing less pollution.

### *C. The Task Based Formulation*

In this section, we present the concept of "task" to help model the problem. A task is defined as a transportation volume that transports one or two containers from its source port to its destination port. Each task can be serviced by one truck only. Several properties are associated with each task. These properties are: *container size*, *source port*, *destination port*, *container quantity* and a *time window* $(a,\ b)$ indicating its available time and deadline. The length of the time window of a task varies from several hours to several shifts. However, most tasks are available for more than one shift before its deadline.

The detailed process of completing one task consists of three actions:

- Picking up commodity from the source port.
- Transporting the commodity to the destination port.
- Unloading the commodity at the destination port.

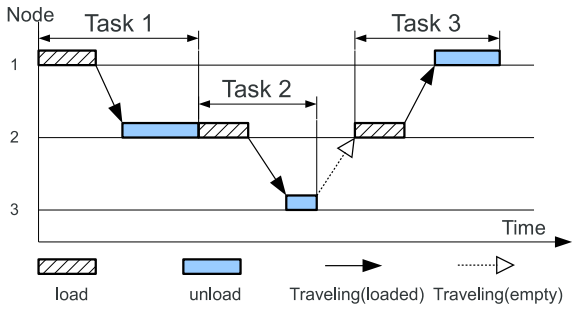


Fig. 2. Example of completing several tasks by a truck

Each action in the process will take some time to complete. Picking up container has a *loading time*. While unloading the container creates *unloading time*. The transportation time is the travel time from the source port to the destination port. The distance travelled during transportation time is called *heavy-load distance* or *loaded distance*. If the truck is just deadheading (i.e. not carrying anything), it is called *empty load distance*. This is shown in the Figure 2 as dotted arrow between task 2 and task 3.

This problem is a short-haul routing problem since the travel time ranges from half an hour to three hours. The loading time and unloading time takes from about half an hour to one hour. It is obvious that loading and unloading time has a heavy weight on the total time of completing a task. These two periods should be counted separately and should not be ignored. They cannot simply be combined as part of "travel time" on the arcs or "servicing time" on nodes because the time taken at each port is dependent on whether both loading and unloading are involved or only one of them is required.

The urgency of a task is measured by how close the current time is to the task's deadline. Correctly evaluating the urgency of each task is crucial in this problem. If a task is due, it will cause severe problem for the company. In this formulation, the urgency level of each task is evaluated in each shift. If a task must be finished before a shift ends, then the task is a *mandatory task* for this shift. If, for a given shift, a task can be finished in later shifts, then the task is an *optional task* for this shift.

Assume that there are three tasks to be finished as shown in Figure 3. Task 2 is available from shift 1 to shift 2. Thus, it is an optional task for shift 1 and a mandatory task for shift 2. Task 3 has a shorter time window and it must be finished in shift 1. Thus it is a mandatory task for shift 1. Task 1 is a special case. The available time of task 1 in shift 3 is quite short. It is not possible to finish this task in shift 3. Thus, task 1 is a mandatory task for shift 2 and an optional task for shift 1.

For convenience of implementation, we classify tasks into mandatory task set and optional task set whenever a shift is considered. For the example in Figure 3, shift 1 has one mandatory task (task 3) and two optional tasks (task 1 and

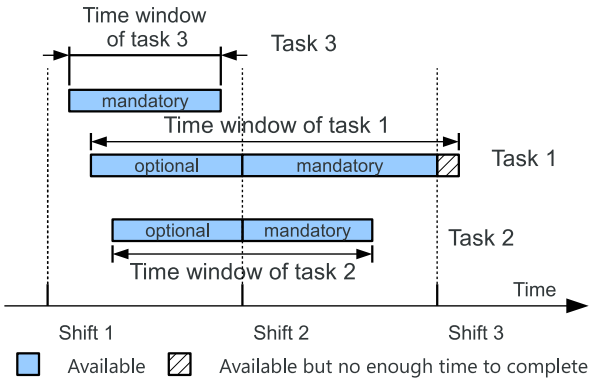


Fig. 3. Example of three tasks and four shifts.

task 2). The mandatory task set of shift 2 depends on whether task 1 or task 2 was finished in shift 1. If task 1 is not finished, then the mandatory task set of shift 2 will have task 1. This is same for task 2. It can be inferred that the task sets of one shift is partially dependent on the routing of previous shifts.

The task based routes have several advantages over a node based representation in this problem. In a node based representation, routes are represented as sequences of ports. Thus, additional information should be added to arcs to indicate which task is being serviced or whether the truck is simply deadheading to another node. This will make the problem more complex. While in this task based representation, it is possible to directly tell whether a truck is deadheading by checking whether the destination node and source node of neighbouring tasks are the same. Another advantage is that in a task based formulation, there is no need to consider the size or weight of containers because the information is preprocessed in such a way that any truck can complete any task if time window constraint is not violated. Most importantly, a task based representation makes it possible to let the algorithm consider commodity shipments as "nodes". Since many neighbourhood functions designed for VRP operates on nodes, this model will allow algorithms that were proposed for VRPs to be used on commodity flow problems.

### D. Objectives

The total travel distance of completing a given list of tasks varies according to the routing plan. If the destination port and source port of two consecutive tasks match, then the travel distance of completing these two tasks is minimal. This minimal travel distance equals to the heavy-load distance of completing the two tasks. This characteristic is also true when the truck is trying to complete a list of tasks. That is, the lower bound of the travel distance of completing a list of tasks is equal to the sum of heavy-load distance of completing all the tasks in the list. However, a route plan with zero deadhead distance is very rare in real-life problems.

From the discussion above, it can be concluded that reducing the deadhead distance of trucks in a route plan is the key to reduce costs. In this paper, we aim to minimize the empty-load distance as the objective in the model.

### E. Mathematical Formulation

Consider a graph $G = (N, A)$ that consists of a set of nodes representing geographically distributed ports, and a set of arcs that interconnects all nodes. The node with index 0 is the *depot*. Each node $i \in N$ is associated with two values: the *loading time* $l[i]$ and *unloading time* $u[i]$. Each arc from node $i$ to node $j$ is associated with a *travel time* $t[i,j]$ and a *travel distance* $d[i,j]$, and $\forall i \in N, d[i,i] = 0, t[i,i] = 0$. The travel time and travel distance is symmetric. That is, $\forall i,j \in N, d[i,j] = d[j,i], t[i,j] = t[j,i]$. There is a set of *tasks* $C$ to be serviced. Each task $c \in C$ has its source node $\alpha(c) \in N$ and destination node $\beta(c) \in N$ and a time window $(a[c], b[c])$. $\sigma(c)$ is the starting time of task $c$ and $\varsigma(c)$ is the finishing time of task $c$.

A solution $x$ contains a continuous set of *shifts* $S$. "Continuous" here means that the end time of one shift is the start time of the next shift. For each shift $s \in S$, the start time and end time of the shift is represented as $(e_s, f_s)$. In each shift, there are a maximum of $v$ *routes*. A route $r$ is a feasible sequence of tasks that a truck can complete in the shift. The binary variable $\Gamma_c^r$ indicates whether task $c$ in $C$ is used in $r$.

$$\Gamma_c^r = \begin{cases} 1 & \text{if } c \text{ is serviced in } r. \\ 0 & \text{if } c \text{ is not serviced in } r. \end{cases}$$

Assume that set $R$ contains the permutations of all feasible *routes* for the problem. Then a feasible solution contains a set of routes from $R$ that finishes all the tasks in $C$. $\delta_r^s$ is a binary decision variable that indicates whether $r$ is used in shift $s \in S$.

$$\delta_r^s = \begin{cases} 1 & \text{if } r \text{ is used in } s. \\ 0 & \text{if } r \text{ is not used in } s. \end{cases}$$

We relabel the tasks in $r$ in shift $s$, so that the tasks in route $r$ becomes $(c_1^{rs}, c_2^{rs}, \ldots, c_n^{rs})$. For a given task $c_k^{rs}$ in route $r$, Let $\alpha(c_k^{rs})$ be the source node of task $c_k^{rs}$ and $\beta(c_k^{rs})$ be the destination node of task $c_k^{rs}$. Thus, it can be inferred that $d[\alpha(c_k^{rs}), \beta(c_k^{rs})]$ is the travel distance of task $c_k^{rs}$ and $t[\alpha(c_k^{rs}), \beta(c_k^{rs})]$ is the travel time for this task. Thus the starting time of $c_k^{rs}$ can be represented as:

$$\sigma(c_k^{rs}) = \begin{cases} \varsigma(c_{k-1}^{rs}) + t[\beta(c_{k-1}^{rs}), \alpha(c_k^{rs})] & k > 1 \ , \varsigma(c_{k-1}^{rs}) > a[c_k^{rs}]. \\ a[c_k^{rs}] + t[\beta(c_{k-1}^{rs}), \alpha(c_k^{rs})] & k > 1 \ , \varsigma(c_{k-1}^{rs}) < a[c_k^{rs}]. \\ e_s + t[0, \alpha(c_k^{rs})] & k = 1 \ , e_s > a[c_k^{rs}]. \\ a[c_k^{rs}] + t[0, \alpha(c_k^{rs})] & k = 1 \ , e_s < a[c_k^{rs}]. \end{cases}$$

and the finishing time $\varsigma(c_k^{rs})$ of $c_k^{rs}$ becomes:

$$\varsigma(c_k^{rs}) = \sigma(c_k^{rs}) + l[\alpha(c_k^{rs})] + t[\alpha(c_k^{rs}), \beta(c_k^{rs})] + u[\beta(c_k^{rs})]$$

This recursive presentation ensures that each task's starting time and finishing time are correctly calculated from the previous task in the same route. The first task's starting time in a route is either the task's available time or the earliest arrival time for the truck to the task's source node. The objective of the problem is to finish all mandatory tasks of the given set of shifts while keeping the empty load distance as low as

possible. The problem can be modelled as follows:

$$\text{Minimize} f(r) = \sum_{s\in S}\sum_{r\in R}\delta_r^s(\sum_{i}^{n-1} d[\beta(i),\alpha(i+1)] + d[0,\alpha(1)] + d[\beta(n),0]) \quad (1)$$

Subject to

$$\sum_{r\in R}\delta_r^s \leq v, \quad \forall s \in S. \quad (2)$$

$$a[c] \leqslant \sigma(c) \text{ and } \varsigma(c) \leqslant b[c], \quad \forall c \in C \quad (3)$$

$$\varsigma(c_k^{rs}) + t[\beta(k),0] \leq f_s, \quad \forall t_n \in r. \quad (4)$$

$$\sum_{s\in S}\sum_{r\in R}\delta_r^s\Gamma_c^r = 1, \quad \forall c \in C. \quad (5)$$

$$\delta_r^s = 0,1 \quad (6)$$

Constraint (2) limits the maximum routes so that it does not exceed the number of available trucks. Constraint (3) is to ensure that the time window of each task is satisfied. Constraint (4) limits the maximum travel time so that trucks return back to the depot before this shift ends. Constraint (5) makes sure that each task is serviced once only.

## IV. Methodology

We considered several methodologies to solve this new problem. The time-space network can be used to model this problem in the way similar to what SNDP does. Since this is a short-haul routing problem, using a time-space network will result in a very large network. This becomes much more complex when there are more than 200 containers everyday. It will be impossible to find a good solution within a reasonable time. Many other exact algorithms have similar issue.

Since the tasks in task based model is quite similar to the nodes in VRP, it is quite natural for one to consider the metaheuristics for VRP. A two-stage metaheuristic algorithm is proposed. In the first stage, an insertion heuristic is used to quickly generate a good quality starting solution. The insertion heuristic has be demonstrated to be a good starting solution method heuristic for the VRP [8]. Then in the second stage, a VNS algorithm with a tabu list is used to further improve the quality of the starting solution. The VNS and tabu search are also efficient metaheuristics for the VRPTW and SDVRP. Thus we hybridize VNS with tabu search in hope to gain improvement over existing software used in the company.

### A. Data Preprocessing

The commodity information from declaration forms cannot be directly used in task based route planning and it is necessary to pre-process these data. This section describes in detail how commodities are converted into tasks defined in the previous section and how commodities are classified as mandatory or optional.

Before the algorithm starts, all pairs of small containers with the same declaration ID[1] are combined into one task which corresponds to the maximum load that a truck can sustain. Large containers are not combined due to the capacity constraints of trucks. After tasks are generated, the urgency level of tasks in each shift is then calculated.

The total time of completing a task $c \in C$ is $TC_c = l_{\alpha(c)} + t[\alpha(c),\beta(c)] + u_{\beta(c)}$. The $DST_c$, which represents the deadline for task $c$ to be started, is calculated as follows:

$$DST_c = b[c] - TC_c \quad (7)$$

Similarly, the $ECT_c$, which represents the earliest completion time possible for task $c$, is calculated as:

$$ECT_c = a[c] + TC_c \quad (8)$$

For a shift $s$, a mandatory task $c$ should satisfy:

$$e_s \leqslant DST_c \text{ and } b[c] \leqslant f_s - TC_c \quad (9)$$

While optional tasks are available before the end of shift but can be completed later, this is mathematically represented as:

$$ECT_c < f_s \text{ and } DST_c > f_s \quad (10)$$

We do not use the available time of commodity because if the available time is before the end of shift $f_s$ but there is not enough time to finish it, then it will be meaningless to even consider this task in the shift.

### B. Insertion Heuristic

The proposed first stage algorithm is a parallel insertion heuristic similar to the insertion heuristic in [8]. To generate routes for a shift $s$, the insertion heuristic first tries to create initial routes by assigning one task to each truck's route based on initialization criteria. Then, it inserts mandatory tasks of the current shift to the initial routes. After all mandatory tasks of the current shifts are inserted, the algorithm will try to assign tasks from the optional task set of shift $s$. Instead of picking a task from the whole optional task set for the current shift, the algorithm only picks the mandatory tasks from the next shift. It should be noted that these mandatory tasks in the next shift still belong to the optional task set for the current shift $s$. This insertion process is carried out until all trucks have an initial task in shift $s$ or there is no more tasks can be assigned in shift $s$. This picking order can reduce the work-load of the next shift. The pseudo code is presented in Algorithm 1. Note that $x.s$ represents the solution for shift $s$.

*a) Initial routes:* The initial routes are generated using two *initialization criteria*. In the first initialization criterion, all routes are initialized with the most urgent tasks that have deadlines closer to the shift start time. In the second initialization criterion, tasks that have earlier available time are inserted first.

The detail of the process of initializing a shift is described as following. Assume that the algorithm is initializing the shift $s$, the mandatory tasks of shift $s$ is first considered. Sometimes there are not enough mandatory task in shift $s$. This will leave some trucks with no initial task at all. Thus, a strategy that is similar to the insertion strategy is used. It tries to assign mandatory tasks from the next shifts until all trucks have an

[1]The containers with the same declaration ID have the same source and destination nodes and the same time window

initial task or there is no more tasks in the optional task set of $s$.

*b) Insertion:* In the insertion function, the algorithm evaluates possible insertion points in all routes in this shift for all tasks from set $C'$, which is passed from the parallel insertion heuristic in Algorithm 1. The evaluation process is done by function $evaluateInsertion()$. The evaluated result is stored in the form of [task, route, slot, empty load distance caused]. All evaluated results are stored in list $L$ and sorted in the increasing order of "empty load distance caused" . The pseudo-code is shown in Algorithm 2.

**Algorithm 1** Parallel Insertion Heuristic

**Require:** empty solution $x$
  **for** $s$ in $S$ **do**
    Initialise $v$ routes of $x$ for shift $s$.
    insertTask($s$.mandatory, $x.s$);
    $s' = s$;
    **while** there is more shift(s) after $s'$ **do**
      $s'$ = next shift after $s'$;
      insertTask($s'$.mandatory, $x.s$);
      **if** $s'$ is the last shift in $S$ **then**
        insertTask($s'$.optional, $x.s$);
      **end if**
    **end while**
  **end for**
  **return** $x$

**Algorithm 2** Pseudo-code for insertTask()

**Require:** task list $C'$, solution $x.s$
  $L = evaluateInsertion(C', x.s)$;
  **while** $L$ is not empty **do**
    insert the best task stored in $L$ into $x.s$;
    remove all insertion points with same task/route in $L$;
  **end while**
  **return** $x.s$

### C. VNS with Tabu List

We propose a variable neighbourhood search (VNS) meta-heuristic approach, which operates once the starting solution has been generated by insertion heuristic. The VNS uses multiple neighbourhoods to explore the solution space. The VNS is discussed in detail in [23] with several variants. By exploring different neighbourhood structures, algorithms would have increased possibilities to find better solutions than single neighbourhood approaches.

The pseudo-code for the basic variable neighbourhood search is shown in Algorithm 3. $x$ is an starting solution to be improved. $m$ is the index of neighbourhood function. It indicates that the $m^{th}$ neighbourhood function will be used in the *shaking* function. The shaking function picks a random neighbouring solution generated by the $m^{th}$ neighbourhood. It is implemented to avoid becoming trapped in a local optima. $m_{max}$ is the index of last neighbourhood function.

**Algorithm 3** Pseudo-Code for Variable Neighbourhood Search

**Require:** starting solution $x$, $maxTime$, $m_{max}$
  **while** CpuTime() $< maxTime$ **do**
    $m = 1$, $x' = x$
    **while** $m \leqslant m_{max}$ **do**
      $x'$ = shake($x'$, $m$);
      $x'$ = VND($x'$, $m_{max}$);
      $m$ = fitness($x$, $x'$, $m$);
    **end while**
  **end while**
  **return** $x$

**Algorithm 4** Fitness Evaluation Function: fitness()

**Require:** original solution $x$, modified solution $x'$, $m$
  **if** $f(x) < f(x')$ **then**
    $m = 1$, $x = x'$;
  **else**
    $m = m + 1$;
  **end if**
  **return** $m$

The VNS in this paper uses variable neighbourhood descend (VND) shown in Algorithm 5 to optimize a solution. The $bestImprove()$ function in VND picks the $\mu^{th}$ neighbourhood function and returns the best neighbouring solution generated by that neighbourhood.

In this research, the VND and shaking uses three types of neighbourhoods: Insertion, swap and remove. They are further split into the following neighbourhood functions.

- Remove one task from a route and insert it into another route in the same shift.
- Swap the position of two tasks from different routes in the same shift.
- Remove one task from a route and insert it into another route that belongs to an adjacent shift.
- Swap the position of two tasks from two routes that belongs to an adjacent shifts.
- Insert one task from the mandatory task list of the current shift into a route.
- Insert one task from the next day's mandatory task list into a route in the current shift.
- Insert one task in the optional task list of the current shift

**Algorithm 5** Variable Neighbourhood Descent: VND()

**Require:** starting solution $x$, $\mu_{max}$
  **while** improved($x$) **do**
    $\mu = 1$, $x' = x$;
    **while** $\mu \leqslant \mu_{max}$ **do**
      $x'$ = bestImprove($x'$, $\mu$);
      $\mu$ = fitness($x$, $x'$, $\mu$);
    **end while**
  **end while**
  **return** $x$

into a route.

- Remove an optional task from a route in the current shift.

The third and forth neighbourhood functions only swap or move tasks between adjacent shifts. Moving or swapping tasks from non-adjacent shifts may sometimes increase the workload of the next shift because there might be less tasks finished in the next shift. The last neighbourhood function allows the algorithm to remove tasks in the current route. This is more useful in the shaking function as it allows the algorithm to try different sets of tasks for a shift. Thus, regions of the solution space that are further away from the current solution can be explored. The mandatory tasks in the current shift and next shift will not be removed in order to maintain the feasibility of the solution during the search.

Neighbourhood functions of the above list are called one by one in VND. Once a neighbourhood function can no longer improve the current solution, the next one is called. If any better solution is found, the algorithm will return to the first neighbourhood function.

However, the shaking function alone was found to be insufficient in our initial experiments. After VND, the solution reached a local optimum. Thus, the new random solution by shaking is always worse than the local optimum. In the next loop, there is a good chance that the search will revert back to the same local optimum, wasting computational resource.

In our implementation, this cycling issue is addressed by incorporating a tabu list and repeating the shaking process several times before the VND starts again. This proves to be an effective way to increase the performance of the algorithm based on our experimental results.

The tabu list stores the declaration form ID. Once a task has been processed by a neighbourhood function, all tasks with the same source, destination and time window will be prohibited by neighbourhood functions. The purpose of incorporating this tabu list is to prevent the solution from returning to the same local optimum after the shaking function. Since the tabu list is also effective during the VND search process, recently moved tasks will not be moved back again. Thus, the algorithm is encouraged to explore solutions farther away from the current region. It was observed that the length of the tabu list should be no less than the number of shakings after VND. After trying different tabu lengths, it was found out that the tabu list with a length of 7 enables a more effective search. Our initial experiments also show that longer tabu list did not obtain significant improvement but neither did it make the performance worse.

## V. Data and Experiment Results

In this section, the results of the proposed algorithm are shown. The possible improvement by using this method is also discussed. We evaluate our algorithm's performance by the heavy load distance rate (LDR), which is calculated as:

$$LDR = \frac{loaded\ distance}{total\ travel\ distance} \tag{11}$$

The company currently does not have records of loaded distance rate for single days. Thus, the average of results of busier shifts is compared. As mentioned in section III-B, the average LDR results of the company's current software is less than 70%. The criterion of choosing data is based on two aspects: commodity quantity and diversity of nodes.

Busier shifts with about 200 - 300 commodities are preferred as the data input. It is observed that in most cases, shifts with very few commodities will naturally reduce the efficiency. This is understandable because trucks will usually be assigned with just one task. The tests on busier days suits the goal of the problem better.

Apart from the quantity of commodities, the diversity of commodities in the data set is also considered. This is to test the algorithm's performance under more complex situations. That is, whether the algorithm still performs well when tasks in the shifts have very different sources and destinations. If the most mandatory commodities in a shift have a same source and a same destination, then longer empty load distance will be inevitable. The improvement from algorithm becomes less obvious.

We choose a night shift or a shift with less commodities as a starting shift. This allows some mandatory tasks to be finished before the busy shift starts. Otherwise, there might be some unfinished mandatory tasks, which would have a chain effect and result in unfinished tasks in later shifts.

Based on these two criteria, the following shifts are chosen as the test case:

| instance | Shift Starting Date | number of shifts |
|---|---|---|
| NP1 | 27-3-2012 20:00:00 | 8 |
| NP2 | 01-4-2012 08:00:00 | 8 |
| NP3 | 24-4-2012 20:00:00 | 6 |

TABLE I
SHIFTS TESTED IN THREE INSTANCES AT NINGBO PORT IN CHINA

The commodity data is extracted from the declaration forms from the company. The algorithm is run on a PC with single core of a Intel I7-2672 processor. VNS is allowed to run for 50 seconds. By observation, the algorithm starts to converge after about 20 seconds.

The two initialization criteria of the insertion heuristic generates similar results as shown in Table II. The available time based criterion generates slightly better results. The gap is within 1 percent. However, it sometimes failed to allocate mandatory tasks in the route. The reason might be that the available time based criterion is not aware of some urgent tasks. When multiple shifts are considered, the chance of failing to assign mandatory tasks is greatly decreased. It is observed that most tasks are finished before the current shift. However, after the optimization of VNS, this infeasibility issue is solved.

The final results are summarized in the Table III. N1/N2/N3-a are optimized based on the starting solutions generated by the insertion heuristic with available time based initialization criterion. While N1/N2/N3-b are optimized based on the starting solutions generated by the insertion heuristic with deadline based initialization criterion. Result shows that VNS

| Instance | Avg. of criterion 1 | Avg. of criterion 2 |
|---|---|---|
| NP1 | 71.6% | 71.5% |
| NP2 | 75.5% | 75.2% |
| NP3 | 70.0% | 70.0% |

TABLE II
LDR OF THE SOLUTIONS OBTAINED BY INSERTION CRITERION 1 AND 2

| Shift | NP1-a | NP2-a | NP3-a | NP1-d | NP2-d | NP3-d |
|---|---|---|---|---|---|---|
| 1 | 87.9% | 76.5% | 93.3% | 86.3% | 76.4% | 92.2% |
| 2 | 68.2% | 85.6% | 73.1% | 81.8% | 87.1% | 72.8% |
| 3 | 87.7% | 78.7% | 69.2% | 82.8% | 80.4% | 69.7% |
| 4 | 70.5% | 67.2% | 83.0% | 72.2% | 71.6% | 87.0% |
| 5 | 81.0% | 68.8% | 93.7% | 79.6% | 69.6% | 92.6% |
| 6 | 69.4% | 88.3% | 61.0% | 70.2% | 91.8% | 59.4% |
| 7 | 75.4% | 84.3% | - | 74.2% | 95.5% | - |
| 8 | 72.1% | 85.3% | - | 73.4% | 86.3% | - |
| Avg. | 76.0% | 79.0% | 76.6% | 77.1% | 80.1% | 76.6% |

TABLE III
LDR OF FINAL SOLUTIONS BY VNS ALGORITHM WITH DIFFERENT SETTINGS OF INSERTION HEURISTIC

with tabu list further improves the efficiency of routes. Some shifts can be optimized to very high efficiency. In most results, the LDRs range from 75 to 80 percent. This gives us about 5-10% of improvement over the current solution in practice.

## VI. Concluding Remarks and Future Research

In this paper, we have developed a model for a real-world problem that resembles a vehicle routing problem and a service network design problem but has its own distinct features. A task based modelling approach, which allows the use of node based optimization algorithms on commodity flow problems, is introduced. A two-stage algorithm based on this model shows promising results when tested on real world data with an improvement of 5-10% compared with the results in practice. If this is integrated with the Ningbo Port company's routing software, the results can potentially provide useful information for decision makers when they decided to build more relay nodes or new ports in the long run. Though the context of this project is based on the case of a local company in Ningbo port, the model can be extended to solve other problems that shared the same features by many other companies.

There still exists some uncertainty factors that can affect this problem, making it even more challenging. In the future, the uncertainty of travel time should be investigated as it is a important factor that affects the actual efficiency of the routing plan.

*Acknowledgement*

This research is supported by the National Natural Science Foundation of China (NSFC 71001055) and Zhejiang Natural Science Foundation (Y1100132).

## References

[1] P. Toth and D. Vigo, Eds., *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002.

[2] T. G. Crainic, "Service network design in freight transportation," *European Journal of Operational Research*, vol. 122, no. 2, pp. 272 – 288, 2000.

[3] G. Dantzig and J. Ramser, "The truck dispatching problem," *Transportation Science*, 1959.

[4] N. Christofides, A. Mingozzi, and P. Toth, *The vehicle routing problem*. Wiley, Chichester, 1979.

[5] G. Laporte, "Fifty Years of Vehicle Routing," *Transportation Science*, vol. 43, no. 4, pp. 408–416, NOV 2009.

[6] B. Eksioglu, A. V. Vural, and A. Reisman, "The vehicle routing problem: A taxonomic review," *Computers & Industrial Engineering*, vol. 57, pp. 1472 – 1483, 2009.

[7] J.-F. Cordeau, G. Laporte, M. W. SavelsBergh, and D. Vigo, *Vehicle Routing*. Elservier, 2007.

[8] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, March-April 1987.

[9] E. Taillard, P. Badeau, M. Gendreau, F. Geurtin, and J. Potvin, "A tabu search heuristic for the vehicle routing problem with time windows," *Transportation Science*, vol. 31, pp. 170–186, 1997.

[10] W.-C. Chiang and R. A. Russell, "A reactive tabu search metaheuristic for the vehicle routing problem with time windows," *Journal on Computing*, vol. 9, no. 4, 1997.

[11] O. Bräysy and M. Gendreau, *Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms*. Transportation Science, February 2005, vol. 39.

[12] N. A. El-Sherbeny, "Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods," *Journal of King Saud University - Science*, vol. 22, no. 3, pp. 123 – 131, 2010.

[13] M. Dror and P. Trudeau, "Savings by split delivery routing," *Transportation Science*, vol. 23, p. 141145, 1989.

[14] M. Dror, G. Laporte, and P. Trudeau, "Vehicle routing with split deliveries," *Discrete Applied Mathematics*, vol. 50, no. 3, pp. 239 – 254, 1994.

[15] J.-F. Cordeau, G. Laporte, J. Y. Potvin, and M. W. Savelsbergh, "Chapter 7 transportation on demand," in *Transportation*, C. Barnhart and G. Laporte, Eds. Elsevier, 2007, vol. 14, pp. 429 – 466.

[16] C. Archetti and M. Speranza, "The split delivery vehicle routing problem: A survey," *The Vehicle Routing Problem: Latest Advances and New Challenges*, vol. 43, pp. 103–122, 2008, cited By (since 1996) 9.

[17] T. G. Crainic and J.-M. Rousseau, "Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem," *Transportation Research Part B: Methodological*, vol. 20, no. 3, pp. 225 – 242, 1986.

[18] T. G. Crainic and J. Roy, "Or tools for tactical freight transportation planning," *European Journal of Operational Research*, vol. 33, no. 3, pp. 290 – 297, 1988.

[19] T. G. Crainic and G. Laporte, "Planning models for freight transportation," *European Journal of Operational Research*, vol. 97, no. 3, pp. 409 – 438, 1997.

[20] J. Andersen, T. G. Crainic, and M. Christiansen, "Service network design with management and coordination of multiple fleets," *European Journal of Operational Research*, vol. 193, no. 2, pp. 377 – 389, 2009.

[21] A. Hoff, A.-G. Lium, A. Lokketangen, and T. G. Crainic, "A metaheuristic for stochastic service network design," *Journal of Heuristics*, vol. 16, no. 5, pp. 653–679, OCT 2010.

[22] R. Bai, G. Kendall, R. Qu, and J. A. Atkin, "Tabu assisted guided local search approaches for freight service network design," *Information Sciences*, vol. 189, no. 0, pp. 266 – 281, 2012.

[23] P. Hansen, N. Mladenovic, and J. A. M. Prez, "Variable neighbourhood search: methods and applications," *Springer*, 2008.