Solving a Real-Life VRP with Inter-Route and Intra-Route Challenges

J. Gromicho, S. Haneyah*, A.L. Kok

Algorithmic R&D, ORTEC, P.O. Box 75, 2700AB Zoetermeer, Netherlands

Abstract

We address a rich vehicle routing problem (VRP) motivated by practice. The problem includes three challenges that have received at most little attention in literature: inter-route restrictions, unmatched pickups and deliveries, and priority orders. These characteristics arise in a VRP application where a large international herd improvement company needs to plan milk sampling and measuring tools distribution activities. The latter require expensive equipment that is of limited availability, resulting in inter-route restrictions due to the use of a central inventory. The equipment can also be moved directly from one farm to another, and thus can be modeled as unmatched pickup and deliveries. We solve the problem using a large neighborhood search heuristic, where we also employ local search procedures to intensify the search on promising solutions. We show that we are able to generate plans with more tasks planned and at the same time with 5% less distance traveled as compared to current practice.

Keywords: Rich vehicle routing problem; inter-route restrictions; Large Neighborhood Search; Unmatched pickups and deliveries.

1 Introduction

The Vehicle Routing Problem (VRP) is one of the richest and most studied problems within Operations Research (OR). Studies on VRP have been

*Corresponding author. Tel.: +31 88 678 3265; fax: +31 88 678 3239

E-mail adresses: {joaquim.gromicho, sameh.haneyah, leendert.kok}@ortec.com

developing and elaborating to move from the basic VRP toward a number of variants and extensions. A lot of the extensions are basically intra-route constraints, i.e., constraints affecting single routes. Examples are limited vehicle capacities, time windows for serving customers, and precedence relations (e.g., pickup and deliveries). On the other hand, fewer studies focus on inter-route constraints, which are constraints of a global effect. Hempsch and Irnich [1] propose a generic model for VRPs with inter-route constraints and derive efficient local search techniques for evaluating neighborhood solutions considering the inter-route constraints.

In this paper, we study a VRP motivated by practice, which entails three challenges that receive less attention in literature, especially when being dealt with simultaneously. The problem concerns a large international herd improvement company, which strives for a planning method for collecting milk samples, delivering sperm, and (re)distributing measuring tools. These activities require expensive equipment which is of limited availability. The first challenge they face is an unmatched pickup and delivery problem, where equipment have to be picked up or delivered at some farms in the network. The pickups and deliveries are unmatched in the sense that there is no strict relation on which picked up equipment should be used to satisfy a certain delivery.

The delivery of equipment issue leads us to the second challenge of this paper, which is inventory. Upon starting the route, the driver can carry equipment along from the central inventory available for all drivers. Therefore, the plan has to clearly indicate how many pieces of each equipment type should each driver carry at the beginning of the route. Here comes the issue of inter-route dependency, since drivers share the central inventory, i.e., taking more equipment by one driver will result in less equipment available to other drivers. Therefore, a main source of equipment is loading from the central inventory at the depot, while the other source, as mentioned earlier, can be the pickups along the route. In the latter case, we note that some picked up tasks may enable subsequent delivery tasks. Note that replenishment of the depot occurs when drivers bring back equipment at the end of their routes.

The third challenge is concerned with orders not being of equal priorities. We have high priority orders and low priority orders. This distinction has a significant impact on the solution approach and plays a crucial role in the criteria used to evaluate plans. In fact, the company's planning is executed in a rolling horizon where the planned tasks of a certain day may affect the

task priorities in other days to come, so the solution approach simultaneously affects and is affected by the priorities of tasks.

In this paper, we propose a large neighborhood search (LNS) heuristic to solve the rich VRP at hand. We refer to Ropke and Pisinger [2] who propose an adaptive LNS heuristic for the pickup and delivery problem with time windows.

This paper is structured as follows: Section 2 describes the problem and key modeling aspects. Section 3 presents the solution approach. Section 4 presents computational experiments. Finally, Section 5 concludes this paper.

2 Problem description and modeling aspects

The VRP we deal with comes from a periodic planning problem, which the company faces in a period of about four to six weeks. The activities to be planned are collection and distribution tasks that include deliveries and pickups of measuring tools, where a delivered tool to a certain location, e.g., farm, has to be picked up the next day. The VRP itself focuses on the *daily* planning in this rolling horizon, where for a certain day we have a batch of tasks available, which can be *Must* tasks, i.e., have to be done on the day being planned, or *May* tasks, i.e., can be done on another day in this rolling horizon.

The company generates tasks to be planned at a higher level and normally a time-window of several days is given to each task. However, for a batch of tasks to be planned on a certain day we only know whether a task is a Must or a May. The business rule behind this is that given, e.g., a one week time window, if a delivery (of a certain equipment) is planned on a certain day then collecting the equipment delivered, i.e., the pickup, is a new task that is generated as a Must for the next day. In addition, a delivery task can change from May to Must on some day, when the end of its time window approaches and it still needs to be delivered and then picked it up before a certain deadline.

There are about 10 shifts to be planned per day (around 20 tasks per shift), where each shift starts at the driver's home location and ends at the depot. The fleet is fixed and one route is planned for each driver. The planning of a working day is ready one day in advance. Therefore, when drivers end their routes at the depot, they can already load the required equipment for their route plan of the next day in order to start their routes

from home.

The tasks involve milk samples and also different types of measuring tools. The latter use vehicle capacity in terms of space and load. Efficient use of these expensive and limited measuring tools is important. In this context, we emphasize that there are different tool types. Therefore, a picked up tool along the route can be used for delivery to another farm along the route only if the latter farm requires the same tool type. The tools can be brought directly from a pick up location to a delivery location, so no intermediate processing, e.g., cleaning, of the equipment is required. There are no further restrictions on the source of the equipment as long as it is of the required type, and thus we have a VRP with unmatched pickups and deliveries.

Inventory is a critical element of this problem and presents an inter-route inventory restriction. Inventory needed at the start of each route is determined, and the total amount of inventory has to be present at the depot for the plan to be feasible. Since pickups along the route are the second source of equipment for deliveries, the sequence of pickups and deliveries on the routes is crucial to the initial inventory requirements. Exploring different sequences of pickups and deliveries has to take place in order to minimize inventory requirements.

The redistribution of measuring tools cannot be modeled by a typical pickup-delivery model, since we do not know beforehand which pickup will be matched with which delivery. Therefore, we set up a tasks distribution model where a task can contribute by either a positive loaded quantity (for pickups) or a negative quantity (for deliveries) for the relevant tool type. Given a certain route plan, we ensure that the vehicle never has a negative load on board by loading a minimum quantity per tool type at the start of the route. A complete day plan of multiple routes has to satisfy the inventory restriction, i.e., the total number of tools loaded over all routes should satisfy the inventory constraints for each tool type at the depot.

The tools are of different types and have different vehicle capacity requirements. Capacity is an intra-route restriction that again depends on sequencing and load changes along the route. For inventory calculations we already trace the quantity on board for each tool type and the changes on every pickup or delivery action. Therefore, the loads from all tool types on board define the required capacity. For a route to be feasible, the total load should not exceed the vehicle's capacity at any moment along the route, or when loading the vehicle at the depot.

In addition to the aforementioned restrictions, there are regional restric-

tions, i.e., certain regions can only be entered on specific days of the week. The objective of this problem has a hierarchical structure, where the aim is: first to maximize the number of Must tasks planned, second to maximize the number of May tasks planned, and third to minimize the distance traveled. This implies that we might not be able to plan all Must tasks on the required day due to, e.g., insufficient capacity.

3 Solution approach

In this paper, we propose an LNS heuristic to solve this problem, since this class of heuristics proved to be effective in solving rich VRP's in literature. Another important motivation stemming from the characteristic of the problem at hand is the Must and May tasks distinction. In order to maximize the number of Must tasks planned according to our first objective, we want to focus on planning these Must tasks first and then extend to the May tasks. However, this approach tends to construct a first group of routes filled up with Must tasks and then a second group of routes filled up with May tasks. This often leads to routes crossing each other and resulting in visits to the same geographical areas by multiple drivers. Simple local search procedures cannot solve these issues, but destruction mechanisms have a better chance of escaping local optima. Moreover, in this problem we cannot plan all tasks at hand, and thus it is interesting to destroy solution parts and reconstruct with a full range of unplanned tasks in an attempt to plan more tasks. The approaches we use in our solution method are as follows:

- **(Re)Construction**: for construction we use a parallel cheapest insertion (PCI) heuristic, where for a certain task we try all possible insertion points in all possible routes and insert at the location with least cost of insertion in terms of distance traveled.

- **Local search (LS) procedures**: we use a combination of classical local search operators from literature. The heuristics we use are: *2-Opt*, *Cross-exchange*, *Relocate*, *Exchange* In the latter two heuristics, we do not only try to shift or swap single nodes, but also sequences of nodes, and we apply the best feasible insertion locations in the routes.

  Within the solution algorithm, we execute either an extensive local search or a limited local search. We execute the extensive search to improve the overall solution before or after the LNS, where we always run

the sequence 2-Opt, cross-exchange, relocate, exchange, 2-Opt, cross-exchange. On the other hand, we execute the limited local search to intensify the search on promising intermediate solutions within the LNS after each destruction and construction step, where we run the sequence 2-Opt, cross-exchange, relocate.

- **LNS based on destruction mechanisms**: we use three different destruction mechanisms. For each mechanism, we destroy a certain percentage of the solution in terms of the number of tasks planned. We typically destroy 5, 10, or 20% of the solution. The destruction mechanisms are as follows:
  - *Random destruction*: this mechanism randomly selects tasks in the solution up to the configured percentage to destroy and removes them from the solution.
  - *Related destruction*: this mechanism selects one seed task randomly, and then includes nearest tasks to this seed until reaching the configured percentage to destroy.
  - *Worst destruction*: this mechanism defines the worst tasks in all routes, where the worst task is that whose removal results in the most gain to the objective function. Worst tasks are added until the configured percentage is reached.

For all destruction mechanisms, some related tasks are always added, e.g., if we select a task at some address in the network then all tasks at the same address are also added. Moreover, when we execute an LNS run we always do a destruction mechanism followed by reconstruction, and then a limited local search if the solution has improved. These steps are executed for every percentage configuration, and then we move to the next destruction mechanism in the sequence we listed them. Note that in the reconstruction step, we consider not only the tasks removed by the destruction, but also the unplanned tasks that were not part of the solution.

The aforementioned solution methods form the building blocks for our solution algorithm. However, we stress that we deal with large data sets and complex problem characteristics, which include various types of constraints to model all operational circumstances that we face in this business case. To this end, we have to modify and add to the solution approaches from literature to implement them in our routing software. The following highlight

the main additional functionality and enhancements that we incorporate in the solution methods:

- *Static Move Descriptor (SMD)*: In order to reduce the computational complexity and efficiently evaluate solution neighborhoods we implement SMD data structures, which encode local search moves in a systematic and solution independent manner, this helps us reduce computational time and increase the scope of search due to avoiding search in areas that are not useful and thus exploring other areas, see [3] for further explanation.

- *Estimators*: due to the complex data structures and large instances we cannot always make explicit calculations for the gain and loss or local search moves. Therefore, we make estimations which provide upper and lower bounds respectively on the expected gains and losses of local search moves. Actual values can be found with complete updates for all nodes in the network with respect to costs, time, distance, loads, etc. The worst tasks determined for the *worst* destruction mechanism are also based on estimations.

- *Feasibility checks and pre-checks*: Feasibility checks are one of the bottlenecks with regards to computational time, since there are all kinds of realistic constrains that we have to check whenever we make a change in the solution. To this end, we built a pre-checks framework that quickly evaluates in an approximate manner whether certain moves can be feasible. If a pre-check on a certain restriction fails it means that the extensive feasibility check for that restriction would also fail. Therefore, not all restrictions can be pre-checked. In order to be even more efficient, we order the pre-checks and feasibility checks to check the restrictions that fails more often first, since then we can save a lot of time on checking other restrictions when checks fail earlier in the process.

- *Groups construction and handling*: in literature local search is often based on single tasks or transport pairs. However, we have a functionality of creating groups of tasks that must be moved or inserted together as a batch, this is logical to do when we have for example multiple tasks close to each other, e.g., within 5 minutes driving time. In this case, it is sensible that one driver handles these tasks in the

same trip. To this end, we have to construct groups before the solution algorithms are executed and during execution we keep track of groups in insertion and in local search where we need to move a whole group at a time and not a single task. This creates additional challenges for the algorithms on, e.g., how to determine feasible local search moves based on task groups and how to deal with sequencing issues within the groups. However, this functionality contributes to both improving the quality of solutions and decreasing computational times.

- *Nearest neighbor*: for the parallel cheapest insertion heuristic we incorporate a nearest neighbor mechanism where we, beforehand, determine the set of nearest neighbors for each task, and then during insertion we do not consider options beyond this set since they are more likely to produce solutions with less quality and also consume more computational time on verifying more options.

- *Filtering mechanisms*: we implement special filters that are executed before the optimization begins. These are of different types, e.g., filter out tasks whose time window is beyond the planning horizon since these would not be planned but would consume optimization effort. Filters make sure we have a clean data set for optimization.

- *Sorting mechanisms*: these mechanisms are used to provide tasks in an order of importance with regard to our optimization criteria. For example, we probably like to sort tasks on priority, so having the Must tasks first and then the May tasks, but we also have other sorting mechanisms for sub-criteria that can be based on driving time, distance from depot, distance from nearest planned task, etc.

- *Limiting insertion attempts*: it is important to limit the extensive number of insertion possibilities, so in addition to the techniques mentioned above, we also have a configurable setting on the maximum number of failed insertions. If this setting is set to 100 for example, then once we have tried 100 different options and could not insert a certain task then we would skip it and try the next one on the sorted list. This is essential to not waste much computational time on a task that is too difficult to plan. In addition, we determine the possible trips for each task, some times a task needs special drivers to execute it due to certain capability requirements. In such cases, and for possibly many

other reasons, we determine the possible trips per task and try insertion points only in those trips.

- *Limiting move possibilities*: the functionality used to limit insertion are also used for move possibility in the local search algorithms. In particular, we also have a tabu list of all moves that we tried and turned out to be infeasible, or feasible but not beneficial. All moves from the tabu list are not tried in local search algorithms, e.g., 2-Opt. The tabu moves are removed from the list when any route involved in the move has changed during the course of the solution.

- *Only changed functionality*: we use this technique in local search algorithms that are used after LNS to focus on trips which have changed in the solution. If there are many trips that remained the same as in the previous solution and have already undergone local search then in the new solution we do not need to execute local search again on them, we would better focus on trips that changed to try to improve them further. This technique proved to be efficient in terms of significantly reducing computational time while not compromising solution quality.

Using the solution methods and additional enhancements for algorithmic efficiency and quality, we set up the algorithm in more concrete terms as follows (see Figure 1):

1. Recursive construction of Must tasks: a pickup might enable inserting deliveries, which could not be inserted before, so we go over the batch of tasks to insert until we cannot plan any more tasks.

2. Execute extensive local search (100 times, break if there is no improvement 10 times in a row).

3. Recursive construction of May tasks with the remaining Must tasks (the latter may be enabled by planning some May tasks).

4. Execute extensive local search.

5. Execute LNS with limited local search (50 times, break if there is no improvement 10 times in a row).

6. Execute extensive local search.

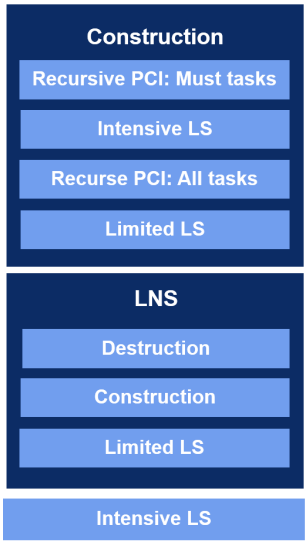

Figure 1: Scheme of the solution algorithm.

After each destruction, the remaining solution should be feasible before reconstruction is executed, e.g., the solution becomes infeasible if we destroy some pickup tasks that were enabling other deliveries in a route. In order to not create additional computational burden with minimal benefit, we did not make changes to repair the remaining solution, e.g., by loading more equipment at the depot after executing a destruction mechanism. We verified in the computational experiments that repairing is not essential for obtaining solution improvements. Finally, we emphasize that the global best solution is always saved and would be the one reported at the end of the algorithm.

4 Computational experiments

For our experimental setup, we run real-life instances of five different working days[1]. We compare current practice to our approach with the LNS. In current practice, the company's optimization software uses parallel cheapest insertion to create a start solution, where the Must tasks are attempted first and when no more Must tasks can be planned in the routes (corresponding to driver shifts) then the May tasks are attempted. This is followed by simple local

[1]The data sets of this business case are provided on the VRP-Repository, see [4]

search procedures, namely relocate, exchange, and 2-Opt. In addition, to cope with the problem of geographical spread due to the Must and May task planning, they use a pre-construction stage in which a seed solution is generated. The pre-construction aims at creating a seed solution with one task per route, where tasks (regardless of their Must or May status) farthest from the depot or any other task location in the network are candidates to be in the seed solution. The candidate tasks are inserted one at a time in one of the two closest empty routes if available/possible. Otherwise, the task remains unplanned at this stage.

We report on the Key Performance Indicators (KPI's) and other Performance Indicators (PI's). Table 1 shows the results of the solution method in current practice, while Table 2 shows the results of the proposed solution method. We see that our solution method improves significantly on the total number of Must tasks planned as the first objective, where 14,27% more Must tasks are planned in total for all instances. This is gained while not compromising the number of May tasks planned. Moreover, a significant reduction is achieved in the distance traveled for routes, where in total the distance traveled is 5,26% less compared to current practice.

It is important that the execution time of the algorithm is acceptable, i.e., for the company this is a daily planning activity for which they want to run the optimization and expect a solution within half an hour at most. As can be seen in Table 2, the CPU optimization time is well within acceptable limits, where the longest optimization run takes about 15 minutes. Actually, LNS is an approach that is computationally intensive on its own, and together with the other algorithms and algorithmic steps we do the time becomes more crucial. However, given the enhancements we incorporate in the algorithms and steering toward solution quality and efficiency (see Section 3), we were able to get these results within relatively short optimization times on a workstation with an Intel Core i7-3740QM CPU 2,70GHz processor and 8.00 GB RAM.

5 Conclusions

This paper discussed an innovative VRP that encompasses a combination of unique features. The first feature is the unmatched pickup and delivery tasks, yet whose sequencing has a crucial effect on inventory requirements per route and on the adherence to capacity restrictions. The second feature is the

Instance	Tasks Planned	May tasks Planned	Must tasks Planned	Routes	Distance (Km)	Duration (h)
Inst. 1	309	114	195	11	2365,47	75,72
Inst. 2	272	86	186	11	2151,59	70,56
Inst. 3	243	97	146	10	2198,49	66,60
Inst. 4	204	92	112	13	2271,89	64,91
Inst. 5	244	98	146	10	2085,85	65,22
Total	1272	487	785	55	11073,29	343,01

Table 1: Results of the solution method in current practice

Instance	Tasks Planned	May tasks Planned	Must tasks Planned	Routes	Distance (Km)	Duration (h)	CPU (min:sec)
Inst. 1	339	117	222	11	2265,54	75,58	15:46
Inst. 2	277	88	189	11	2185,45	70,19	12:27
Inst. 3	274	94	180	10	2024,35	66,24	08:50
Inst. 4	240	91	149	13	2010,33	64,73	06:25
Inst. 5	254	97	157	10	2004,72	65,01	12:44
Total	1384	487	897	55	10490,39	341,75	56:12

Table 2: Results of the proposed solution method

shared inventory, where a central depot has a limited inventory of equipment to be used by all routes. This feature is challenging since it creates an inter-route dependency in the planning. The third feature is that pickup and delivery tasks are not all of equal priority, i.e., some are a must do on certain days while others are a may do.

We proposed a solution method based on LNS as the main tool for solution diversification, with the use of a number of local search heuristics for solution improvement. For the construction of the initial solution, we used a parallel cheapest insertion heuristic. We implemented the aforementioned algorithms in a framework that contains multiple enhancement procedures that aim for improved efficiency and also solution quality.

The proposed method outperforms current practice, it focuses on the hierarchical objectives (i.e., number of Must tasks planned, number of May tasks planned, and the distance traveled), and achieves 8.8% more tasks planned over a complete work week. The additional number of tasks planned comes with more than 5% reduction in the distance traveled. Moreover, with the proposed solution method we show that the initial solution does not become an obstacle for finding better solutions. This is testified by eliminating the need for the pre-construction stage that is used in current

practice. The LNS is able to diversify on constructed initial solutions in contrary to local search approaches that can more easily be locked in a local optimum and find no improvements on some ill-constructed initial solutions.

As a conclusion, we recommend that more attention is paid to algorithms for solving rich VRP's like the one described in this paper. In practice, such VRP's appear with big instances. Therefore, efficiency of the solution approach is essential. The solution approach proposed would be of limited practical relevance without the enhancement procedures that we incorporated.

Acknowledgment

References

[1] C. Hempsch and S. Irnich. Vehicle routing problems with inter-tour resource constraints. In Bruce Golden, S. Raghavan, Edward Wasil, Ramesh Sharda, and Stefan Vo, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pages 421–444. Springer US, 2008. 10.1007/978-0-387-77778-8_19.

[2] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

[3] Emmanouil E. Zachariadis and Chris T. Kiranoudis. A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & Operations Research*, 37(12):2089 – 2105, 2010.

[4] J.E. Mendoza, C. Guret, M. Hoskins, H. Lobit, V. Pillac, T. Vidal, and D. Vigo. Vrp-rep: the vehicle routing community repository. third meeting of the euro working group on vehicle routing and logistics optimization (verolog). oslo, norway, 2014.