MISTA 2013

An efficient metaheuristic for the inventory orienteering problem and the single-vehicle cyclic inventory routing problem

Pieter Vansteenwegen • Manuel Mateo

1 Introduction

Inventory Routing Problems (IRP) are currently getting a lot of attention (e.g. [1-3]). Besides the routing parameters, inventory and handling costs at the customers are also taken into account. Until now, a vehicle routing problem for which no inventory variant has been considered is the Orienteering Problem (OP). The goal of the regular OP [4] is to maximize the total score, collected by visiting a selection of customers, without violating a time constraint. Every customer can be visited at most once and the travel times between the customers are given. For the Inventory Orienteering Problem (IOP), inventory and handling costs are added.

The IOP considers n potential customers ($i=1,\ldots,n$), each with an inventory cost (I_i in euro/ton*hour), a handling cost (H_i in euro/delivery), a constant demand rate (D_i in tons/hour) and a fixed reward (R_i in euro/hour). The time required to travel between two customers i and j is given by t_{ij} and is considered constant. A maximum time limits the availability to visit the customers. Other aspects related to distribution taken into account are the fixed vehicle operating cost (ψ in euro/hour), the average speed of the vehicle (ν in km/hour), the travel cost (δ in euro per km) and the vehicle capacity (Q in tons). It is assumed that every customer has an infinite inventory capacity. Traditionally the objective is to minimise the total cost, although the reward may change the optimization into maximising profit. This problem is almost the same as the Single-Vehicle Cyclic Inventory Routing Problem (SV-CIRP) [5,6,7]. The only difference is that the available time to visit customers is not limited explicitly in the SV-CIRP.

We think the IOP would be closer to model reality than the SV-CIRP, since the IOP can limit this cycle time to a given upper bound, e.g. a number of hours or a working day. Nevertheless, this upper bound will only make the IOP easier to solve than the SV-CIRP. Moreover, benchmark test instances are only available for the SV-CIRP. Therefore, we will focus on the SV-CIRP, knowing that the proposed solution technique can also solve the IOP.

Pieter Vansteenwegen
KU Leuven - Centre for Industrial Management, Traffic & Infrastructure
E-mail: Pieter.vansteenwegen@cib.kuleuven.be

Manuel Mateo
Universitat Politècnica Catalunya, Departament Organització d'Empreses
E-mail: Manel.mateo@upc.edu

An important property, increasing the complexity of the problem, is that the single vehicle is allowed to make multiple trips from the depot during one cycle.

This means we are now looking for a cyclic distribution plan for a single vehicle, starting and ending at a single depot, for a set *S* of selected customers. A key decision variable will be the cycle time *T*, .i.e. the time between two deliveries to each customer. The total travel time will be T^r. Based on this cycle time, the total cost (*Cost* in euro/hour) can be defined as:

$$Cost(T) = \psi + \frac{\sum_{i\in S} H_i}{T} + T * \frac{\sum_{i\in S} D_i * I_i}{2} + \frac{\delta * v * T^r}{T} - \sum_{i\in S} R_i \qquad (1)$$

Please note that this total cost includes four terms, respectively, the vehicle cost, the handling cost, the inventory cost, the travelling cost and the rewards of visiting customers.

The cycle time is lower and upper bounded. It must be not shorter than the shortest travel time required to visit all selected customers divided in multiple trips (*Tmin*) and is limited to *Tmax*, based on the capacity *Q* of the vehicle. For a selection of customers divided in multiple trips, the ideal cycle time would be the "economic-order-quantity" cycle time *Teoq* [3]:

$$Teoq = \sqrt{\frac{\delta * v * Tmin + \sum_{i\in S} H_i}{\sum_{i\in S} \frac{D_i * I_i}{2}}} \qquad (2)$$

However, *Teoq* can only be selected if it lies between *Tmin* and *Tmax*. If *Teoq* is shorter than *Tmin*, the optimal cycle time *Topt* will be *Tmin* and if *Teoq* is longer than *Tmax*, it will be *Tmax.* Otherwise, *Topt*=*Teoq*.

The IRP was first introduced by Bell et al. [8]. A number of comprehensive survey papers about the IRP and its variants are available [1, 9-11]. The SV-CIRP and some exact and (meta)heuristic solution approaches are discussed in [3, 6, 7, 12, 13]. In Zhong and Aghezzaf [12], an in-depth analysis of the SV-CIRP reveals that the objective function is non-smooth and non-convex with many local minima. Zhong [7] discusses in detail the complexity of the SV-CIRP and presents an efficient mathematical formulation. Benchmark instances are available in [3,6,7].

2 Iterated Local Search

The Iterated Local Search (ILS) framework was successfully implemented before to deal with the SV-CIRP [3] and a variant of the OP [14]. The general structure of ILS is described by Lourenço et al. [15]. Algorithm 1 gives an overview of our ILS implementation, which is completely different from the implementation of Zhong and Aghezzaf [3]. The most important functions of this algorithm are *Initialisation, Local Search*, *Perturbation*, *Acceptance* and *Stopping Criterion*. Nevertheless, it is interesting to first discuss two building blocks used by these functions: *Divide* and *Insert.*

Divide tries to determine, for a given selection of customers *S*, the best division in trips and the best sequence of visits in each trip. Remember that the single vehicle can make multiple trips from the depot during one cycle. This division in trips is based on solving a Capacitated Vehicle Routing Problem (CVRP), for an increasing number of trips. The CVRP is solved by using the parallel version of the Clarke and Wright Savings Algorithm [16]. The CVRP-solution is improved, if possible, to decrease the total *Cost.*

Insert adds an extra customer to a given solution, without changing the division of the customers in trips. One by one, all non-visited customers are considered for insertion and the customer leading to the highest decrease in *Cost* is inserted. Then again, all non-visited customers are considered until no more customers can be inserted. When testing a customer for insertion, all possible positions in all possible trips are considered. The best position in each trip is selected based on the lowest increase in travel time, and the best trip to insert the customer is selected based on the highest decrease of *Cost*.

Algorithm 1: ILS for the SV-CIRP

```
NoImprove ← 0
Solution ← Initialisation
BestFound ← Solution
while NoImprove < MaxNoImprove do
    Solution ← Local Search (Solution)
    if Solution better than BestFound then
        BestFound ← Solution
        NoImprove ← 0
    else
        NoImprove=NoImprove+1
    end if
    Solution ←Acceptance Criterion(BestFound, Solution)
    Solution ←Perturbation(Solution)
end while
Return (BestFound)
```

Initialisation starts by selecting randomly half of the customers. Then, *Divide* is applied in order to determine the best division in trips. If this division has a negative *Cost*, meaning it is worthwhile for the vehicle to be used for this tour, an initial solution is found. If this is not the case, again half of the customers are removed and *Divide* is applied again, until a solution with a negative *Cost* is found.

In *Local Search*, a heuristic considers one by one, all non-visited customers for insertion and the best one, decreasing *Cost* most, is inserted. Then again, all non-visited customers are considered until no more customers can be inserted. This heuristic alternates between two ways to look for the best non-visited customer: *Divide* (after adding the "non-visited customer" to the current set) and *Insert*. The most important advantage of this alternation is a high saving of computation time compared to using only *Divide*.

Perturbation evaluates a different combination of customers each iteration. It will remove *Nr* of subsequent customers from the current solution, starting at "position" *Pos*. The parameter *Nr* is initialised to one and is increased by one every iteration. *Nr* is limited to a given percentage of the customers in the current solution. If it becomes larger, it is reset to one. Similarly, *Pos* is initialised to one and is increased by *Nr* every iteration. If *Pos* would be larger than the number of customers in the solution, it is reduced by the number of customers. As a result of this setting of *Pos* and *Nr*, a different combination of customers is evaluated every iteration. This is one of the crucial elements in order to obtain high quality solutions. After the perturbation, the local search heuristic is applied to find a new local optimum.

The *Acceptance Criterion* typically has two extreme implementations [15]: with "random walk", the search always continues from the current solution, regardless its *Cost*; with "better" the search always continues from the best solution (*BestFound*). Our algorithm uses something in between. As a result, the algorithm is able to spend enough effort to further improve the best found solution. But when the number of iterations without improvement increases, the algorithm can spend more and more time on diversification in order to try to improve the current solution.

The algorithm stops when the number of subsequent iterations without improvement (*NoImprove*) reaches the value of *MaxNoImprove.* After testing several values, the following results were obtained with *MaxNoImprove* =200.

3 Experimental Results

The experimental results are based on five sets of ten benchmark instances for the SV-CIRP [3, 6, 7]. We made all sets available here:
http://www.mech.kuleuven.be/en/cib/op/opmainpage#section-21.

The results are summarised in Table 1. Our algorithm is compared with the best available metaheuristic for the SV-CIRP, called "Zho", of Zhong and Aghezzaf [3]. Our ILS algorithm was coded in Java and all experiments were executed on a Dell Latitude E5410 notebook, with Intel Core i5 2.40GHz processor and 4.0GB of RAM. The specifications for "Zho" can be found in [3] and [7].

The first two columns of Table 1 present the name of the set and the number of customers in these instances. The third one indicates the number of instances where our algorithm found better solutions than "Zho". The fourth column gives the average gap (in %) between the two approaches, calculated in this way: Gap = 100*($\mathrm{Cost}_{\mathrm{Zho}}$ – $\mathrm{Cost}_{\mathrm{ILS}}$)/$\mathrm{Cost}_{\mathrm{Zho}}$. Thus, a negative gap means that our approach outperforms "Zho". Finally, the fifth and sixth columns show the average CPU time of both solution methods.

Set	n	Better instances	Average gap (%)	CPU of ILS (s)	CPU of Zho (s)
1	15	3	-0.32	2	3
2	20	10	-21.62	4	212
3	25	2	0.67	28	106
4	30-67	10	-32.71	86	896
5	30-67	10	-26.22	221	1666
All	**15 - 67**		**-16.04**	**68**	**577**

Table 1: Summary of experimental results

Our ILS algorithm clearly outperforms the best technique currently available on all sets of instances, except for the instances of Set 3. For five of these instances, Zho performs better than our ILS. For three instances, both procedures reach the same results and for two instances our ILS performs better. When analysing these ten instances, it appeared to us that for these instances, it is very easy (and worthwhile) to visit all customers. The only decision that needs to be made is how these customers should be divided in trips in order to optimise the cycle time and the total cost; no selection is necessary. Zho focuses much more on that aspect and succeeds in obtaining a better division in trips than our ILS. Nevertheless, for two instances of Set 3, our ILS obtains new best known solutions.

When considering all 50 instances, our ILS obtains on average a *Cost* that is 16.04% better, and it is significantly faster. 32 new best known solutions are found. The quality improvement and the computation time reduction are the most noticeable for the larger instances of Set 4 and 5. Nevertheless, for all ten instances of only 15 customers (Set 1), the solutions obtained by our algorithm are the optimal solutions, in only 2 seconds on average. Only for Set 1 optimal solutions are available [6].

According to us, the main reason we outperform Zho is that we have a stronger diversification during the search process. We remove a changing number of customers in each diversification phase while Zhong and Aghezzaf [3] only relocate customers by swapping pairs of customers from different tours. Furthermore, we exploited some typical characteristics of the SV-CIRP.

4 Conclusions

In this paper, a fast and effective metaheuristic to deal with the inventory orienteering problem (IOP) as well as the single-vehicle cyclic inventory routing problem (SV-CIRP) is presented. One of the characteristics to consider it a very challenging optimisation problem is the fact that the single vehicle can make multiple trips. As a result, not only a selection of customers needs to be made, but the selected customers should also be divided in trips in an optimal way. We developed an efficient Iterated Local Search (ILS) metaheuristic by exploiting this problem's characteristics.

The experimental results clearly show that our algorithm outperforms the existing solution approaches. On average, our ILS improves the results of Zhong and Aghezzaf [3] by 16.04% and 32 new best known solutions are found and reported. For the twenty largest instances with 30 to 67 customers, our approach improves the results with 29.46% and is eight times faster.

Moreover, the fact that our implementation of the ILS framework obtains much better results than the implementation by Zhong and Aghezzaf [3] clearly shows that different implementations of the same framework can lead to significantly different solution algorithms and that the performance obtained by any given framework depends strongly on the specific implementation of that framework.

References

1. Andersson H, Hoff A, Christiansen M, Hasle G, Lokketangen A, Industrial Aspects and Literature Survey: Combined Inventory Management and Routing, Computers & Operations Research, 37, 1515–36 (2010)
2. Schmid V, Doerner K, Laporte G, Rich routing problems arising in supply chain management, European Journal of Operational Research, 224, 435-48 (2013)
3. Zhong Y, Aghezzaf E-H, Effective Local Search Approaches for the Single-Vehicle Cyclic Inventory Routing Problem, International Journal of Services Operations and Informatics, accepted for publication (2013)
4. Vansteenwegen P, Souffriau W, Van Oudheusden D, The orienteering problem: a survey, European Journal of Operational Research, 209,1-10 (2011)
5. Christopher M, Logistics and Supply Chain Management: Strategies for Reducing Cost and Improving Service. Financial Times-Prentice Hall, UK (1998)
6. Aghezzaf E-H, Zhong Y, Raa B, Mateo M, Analysis of the single-vehicle cyclic inventory routing problem, International Journal of Systems Science, 43, 2040-9 (2012)
7. Zhong Y, Exact and Heuristic Methods for the Cyclic Inventory Routing Problem with Side-Constraints. PhD dissertation 2012, Ghent University, Ghent, Belgium
8. Bell WJ, Dalberto LM, Fisher ML, Greenfield AJ, Jaikumar R, Kedia P, et al., Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer, Interfaces, 13, 4–23 (1983)
9. Coelho LC, Cordeau J-F, Laporte G, The inventory-routing problem with transhipment, Computers & Operations Research, 39, 2537–48 (2012)
10. Kleywegt AJ, Nori VS, Savelsbergh MWP, The Stochastic Inventory Routing Problem with Direct Deliveries, Transportation Science, 36, 94–118 (2002)
11. Adelman D, A Price-Directed Approach to Stochastic Inventory/Routing, Operations Research, 52, 499–514 (2004)
12. Zhong Y, Aghezzaf E-H, Combining DC-programming and steepest-descent to solve the single-vehicle inventory routing problem, Computers and Industrial Engineering, 61, 313-21 (2011)
13. Aghezzaf E-H, Raa B, Van Landeghem H, Modeling inventory routing problems in supply chains of high consumption products, European Journal of Operational Research, 169, 1048–63 (2006)
14. Vansteenwegen P, Souffriau W, Vanden Berghe G, Van Oudheusden D, Iterated Local Search for the Team Orienteering Problem with Time Windows, Computers & Operations Research, 36, 3281-90 (2009)
15. Lourenço H, Martin O., Stützle T, Iterated Local Search. In: Glover F, Kochenberger G, editors. Handbook of Metaheuristics, 321-53. Kluwer Academic Publishers, Dordrecht (2003)
16. Clarke G, Wright JW, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research, 12, 568-81 (1964)