A Two-Phase Algorithm for the Cyclic Inventory Routing Problem

Masoud Chitsaz[1, *], Ali Divsalar[2], Pieter Vansteenwegen[3]

[1] *CIRRELT, GERAD and HEC Montréal, Montréal, H3T 1J4, Canada*
masoud.chitsaz@cirrelt.net

[2] *Department of Industrial Management, Ghent University, Technologiepark 903, 9052 Zwijnaarde, Belgium and Faculty of Mechanical Engineering, Babol University of Technology, Mazandaran, Iran*
ali.divsalar@ugent.be

[3] *KU Leuven, Leuven Mobility Research Center, Celestijnenlaan 300, box 2422, 3001 Leuven, Belgium*
pieter.vansteenwegen@kuleuven.be

Abstract

The Cyclic Inventory Routing Problem (CIRP) is concerned with finding a cyclic schedule for the distribution of a single product to a number of customers. The problem involves multiple vehicles that can be dispatched several times during their shift. Each customer has a local inventory capacity, a constant consumption rate and stockouts are not allowed. The goal is to compose multiple trips which serve all customers and minimize the combination of transportation, inventory and vehicle costs, in a cyclic distribution pattern. Each trip can have a different frequency in the vehicle schedule. This is an important aspect that makes this so called CIRP, with its non-linear non-convex objective function and a set of non-linear constraints, more complex than the basic Inventory Routing Problem (IRP).

Our solution approach decomposes the problem into two subproblems: routing and scheduling, which are dealt with in an iterative way. For each subproblem, we propose a new heuristic. Our first heuristic composes trips, based on the cost estimation of moving customers from one trip to another (routing). The second heuristic tries to combine these trips in an acceptable cyclic schedule (scheduling). In order to search the feasible area efficiently, our heuristic branches one-by-one on the edges of obtained local optima. The proposed algorithm is capable of finding high quality solutions in a reasonable time. When the algorithm is tested on 80 available benchmark instances, the best known solution is improved for 60 of these instances and on average a 3.5% improvement is obtained compared to previously best known results.

Keywords: *Cyclic inventory routing problem, decomposition heuristic, two-phase iterative algorithm, saving-based routing heuristic, infinite-time cyclic scheduling*

** Corresponding author: Tel.: 1 (514) 343-7575 - int 8706; Fax: 1 (514) 343-7121*

1. Introduction

Over the last decades, there has been an increased interest in the integration of logistics and resource management, both within and between companies. This has led to the emergence of concepts such as Enterprise Resource Planning (ERP), Supply Chain Management (SCM), Vendor Managed Inventory (VMI), etc. The main idea in these concepts is the collaboration and integration among different stages in the supply chain. Therefore, decisions for the different stages are integrated in order to obtain a better overall performance. Focusing on VMI, the distributor faces the problem of integrating its vehicle routing decisions with the inventory management at the customers. This integrated problem is known as the Inventory Routing Problem (IRP).

In this paper, we discuss a specific variant of the IRP, called the cyclic IRP (CIRP) which deals with the long-term decision making rather than the rolling horizon modeling concept that normally arise in the tactical/operational decision making process. CIRP belongs to the class of long-term and infinite planning horizon inventory routing problems. As a result, the cycle time of the vehicles will be an important decision variable. The cyclic replenishment pattern was first introduced by *Larson (1988)*. This can be considered as a strategic level IRP that aims to minimize the required fleet size over a long period. He studied a cyclic IRP for the New York City Department of Environmental Protection that planned a barge fleet acquisition. *Raa (2006)* solved a cyclic IRP for a paper goods distributor in the Benelux region (Europe) to set up long-term cyclic replenishment schemes from its single warehouse to a set of customers with stable consumption rates. Another example of the application of the long-term IRPs is in the ship fleet sizing for a liner shipping company with fixed long-term cargo contracts *(Christiansen et al. 2007)*. *Ekici et al. (2015)* performed a computational study to find cyclic delivery schedules in a perpetual time horizon for real-life instances provided by an industrial gases manufacturer.

Together with clearly defining the problem and presenting the first complete mathematical formulation of the problem (with all its practical assumptions), we will in this paper also provide some insights in the complexity of the problem. These include the fact that the vehicles are allowed to perform multiple trips on a single day, the importance of the compatibility of the trips in a vehicle schedule, and the bounds on cycle time of the vehicle schedule. Based on these insights, we will then develop a heuristic solution approach for the CIRP. This solution approach is based on decomposing the problem in a routing and a scheduling subproblem. These two subproblems are iteratively solved in separate phases. Within this approach, one can firstly assign the customers to a set of trips, which is the routing problem, and then estimate the minimum required number of vehicles to handle the trips, which is similar to a bin packing problem. In other words, a set of feasible trips is constructed and then the trips will be assigned to vehicles and the optimal frequency will be determined. Our algorithm outperforms existing algorithms in both the quality of the solutions and the calculation time. Another interesting outcome of our algorithm is that high quality initial solutions are calculated in less than half a second. All this makes it possible to apply the algorithm in practice.

The rest of this paper is organized as follows. In Section 2, the CIRP and its practical assumptions are defined in detail. In Section 3, a comprehensive discussion about the state of the art is given. In Section 4, we present the mathematical model of the CIRP. In Section 5, our algorithm is presented. Section 6 discusses the computational experiments and in Section 7 our contributions and findings are summarized and we suggest some directions for further research and real world applications. For the reader's convenience we present a summary of all notations in the Appendix A.1.

2. Problem definition

CIRP deals with a set N of n customers $i \in N$ with deterministic and constant demand rates, d_i units per day for a single commodity and capable of maintaining a local inventory with the capacity of k_i units. A single depot, 0, has to distribute the commodity among the customers in a long-term (infinite) planning horizon. Handling time (θ_i) and cost (φ_i) at the customers and the depot (hours (currency) per delivery and hours (currency) per dispatch, respectively), as well as the inventory cost at the customers, η_i (currency per unit per day) are also considered. Each customer should be served with equidistant intervals between services. In addition, travel times, t_{ij}, between the customers are considered stable over time. This assumption implies that the vehicle route and delivered quantities remain valid (and can be repeated) over the planning horizon. A homogeneous fleet of vehicles, $v \in V$, each with a capacity of κ units, is available. A maximum driving time per day, H, is considered for each vehicle. A fixed cost (ψ) is considered for using each vehicle and the size of the fleet is a decision variable. Every route over a sub-set of customers should start from and end at the depot. The Traveling Salesperson Problem (TSP) route will be considered as a trip. Each customer is served in a cyclic manner (with equidistant intervals between services). As such, a trip cycle time, Γ^p, is introduced, meaning that trip p is performed once every Γ^p time units. Stockouts and split delivery are not allowed and each customer will always be replenished by the same vehicle, in the same trip. This means every feasible solution for the problem is a fixed partition *(Anily & Bramel 2004; Zhao et al. 2007)* of the customers into different trips (but not necessarily a geographical partition).

In the CIRP, for each trip cycle time, a lower bound and an upper bound can be determined. The minimum trip cycle time, Γ^p_{min}, must be greater than or equal to the shortest travel time required to visit all selected nodes. Furthermore, in each trip, the sum of the deliveries over all visited nodes is limited by the vehicle capacity. More precisely, the trip cycle time is bounded from above by Γ^p_{max}, based on the capacity of the vehicle and the nodes visited in the trip. For each trip, the ideal cycle time would be the "economic-order-quantity" (EOQ) cycle time *(Blumenfeld et al. 1985; Aghezzaf et al. 2006)*. However, the EOQ cycle time can only be applied if it lies between Γ^p_{min} and Γ^p_{max}. If the EOQ cycle time is longer than Γ^p_{max}, Γ^p_{max} is the optimal cycle time; if it is shorter than Γ^p_{min}, Γ^p_{min} is the optimal cycle time. Note that changing the selection of nodes for a trip obviously changes the values for these lower and the upper bounds together with the EOQ cycle time.

Every vehicle can make multiple trips per day as long as the driving time limit is not violated, assuming an average vehicle speed of σ (distance/hour) and the vehicle operating costs of δ (currency/distance). The concept of "multi-trips", has been proposed for the CIRP first by *Aghezzaf*

et al. (2006). More generally, *Fleischmann (1990)* suggests that allowing multi-trips to happen can significantly reduce vehicle numbers and therefore the replenishment costs. In the multi-trip distribution scheme of the CIRP, each trip can have its own frequency. The generalized concept of a multi-frequency multi-trip is called a "vehicle schedule". In each schedule of vehicle v , a subset of trips $p \in P$ is repeatedly performed by the same vehicle. In another word, there is also a fixed partition of the trips into different vehicle schedules. Note that each trip can be executed a different number of times per vehicle cycle T^v, meaning that trip p is performed T^v/Γ^p times per vehicle cycle. All vehicle schedules together are called a "distribution pattern".

For this class of problems the appropriate objective function to be minimized contains the long-run distribution and inventory costs. Hence the objective function should be in form of the rate over a time period (currency/day) as described in *Aghezzaf et al. (2006)*. The cost rate includes the fixed operating cost per vehicle, vehicle transportation costs, cargo handling costs (loading and unloading) at the depot and customers, and inventory cost at the customers. In order to solve this problem, three important decisions need to be taken:

- The assignment of all customers to trips (and the order of visiting these customers per trip);
- The optimal cycle time for each trip that allows to avoid stockouts at the customers and minimizes the costs; and
- The assignment of trips to the vehicles and the distribution pattern for all vehicles.

3. Literature

In this section, we first extensively situate the (C)IRP in the state of the art and then we discuss its features making it a complex problem to tackle. Finally, a number of solution techniques for the CIRP and its most closely related variants will be discussed.

The severe economic environment pressure in 1979 with both recession and double digit inflation was the main reason that motivated Air Products and Chemicals, Inc. top management to seek for possible improvement in integrating inventory management and vehicle scheduling and dispatching within their industrial gas distribution system at their customer locations. This led to the early works on the IRP as described by *Bell et al. (1983)* based on the recognition of the fact that in the real world, inventory costs should play a role in routing problems. Although they considered the joint decision of inventory management and routing decisions over time periods, they did not consider the inventory holding costs in the objective function. *Federgruen & Zipkin (1984)* were the firsts who attempted to integrate the inventory and routing in a single nonlinear mixed integer model. They considered a continuous time inventory cost function and a single delivery to each customer during the planning period. Very soon, the trade-off between routing and inventory costs suggested researchers *(Blumenfeld et al. 1985; Burns et al. 1985)* to use the Economic Order Quantity (EOQ) model in routing problems. They considered distribution, inventory and production set-up costs in a constant demand rate environment. Also, they proposed the use of approximation methods for TSP route estimation in their work. *Gallego & Simchi-Levi (1990)* showed that when the

EOQ for the customers are close to the vehicle capacity, direct delivery is an efficient long-term policy.

The basic IRP assumes that adequate vehicles are available, such that the problem of assigning trips to vehicles is simply ignored. A trade-off between transportation and inventory costs is considered, without taking fixed vehicle costs into account *(Coelho et al. 2014)*. The problem of integrating inventory and distribution decisions has been approached in different ways depending, among others, on inventory policies, service level restrictions, and the time horizon considered. Several extensive literature reviews on IRPs appeared in the last two decades to summarize the state of the art. Some of these reviews are *Kleywegt et al. (2002)*, *Adelman (2004)*, *Moin & Salhi (2007)*, *Andersson et al. (2010)*, *Schmid et al. (2013)*, *Coelho et al. (2014) and Papageorgiou et al. (2014)*. IRP and its variants are now well-developed. The majority of the IRP literature may be classified into models of continuous time *(Anily & Federgruen 1990; Gallego & Simchi-Levi 1990; Aghezzaf et al. 2006)*, usually with a constant demand rate over an infinite horizon, and models in discrete time *(Bertazzi et al. 2002; Campbell & Savelsbergh 2004; Li et al. 2010)*, with a finite horizon and mostly varying demand for the customers. For the first class of models with continuous time, the visit frequency is normally modeled as a continuous decision variable that introduces a nonlinear term to the objective function *(Francis et al. 2006)*. Naturally, the objective function in this class of models with infinite time horizon often targets minimizing the long-run average, or mean average, costs involved *(Moin & Salhi 2007)*. Several models for the case of an infinite horizon have also been developed. *Burns et al. (1985)* are among the first to consider an inventory replenishment problem with vehicle routing costs for an infinite horizon and one warehouse multiple retailer system.

One of the many variants of the IRP that has been considered is the cyclic IRP *(Aghezzaf et al. 2006)*. As mentioned in the introduction, this problem is actually an IRP with an infinite planning horizon and a constant demand rate, requiring a decision on the cycle time of visiting each customer. Therefore, it has both the complexity of IRPs and cyclic scheduling problems *(Levner et al. 2010; Raa 2015)*. According to *Schmid et al. (2013)* and *Lahyani et al. (2015)*, the CIRP can be classified as a rich routing problem and is, as previously discussed, another attempt to close the gap between vehicle routing models and reality. *Viswanathan & Mathur (1997)* proposed a CIRP for a stationary nested joint replenishment policy with multiple products and deterministic demand rates. Specifically they considered equally spaced replenishments for the customers. But they did not assume a fixed vehicle cost in the objective function of their model. *Zhao et al. (2007)* present a fixed partition policy for the CIRP, in which the replenishment interval of each of the customers' partition region as well as the depot is accorded to the power-of-two principle. *Ekici et al. (2015)* solve a CIRP with a simpler objective function that only considers the variable transportation cost which yields to the minimization of the average transportation / routing cost per day. *Zenker et al. (2015)* considered a CIRP when all customers are located on a line that may occur in liner shipping (when feeder ships service inland ports along a stream) and in facility logistics (when tow trains deliver part bins to the stations of an assembly line). *Raa (2015)* discusses a subproblem of the CIRP in which the set of routes that need to be repeated in an infinite time horizon are considered given. The route frequencies can be adjusted for the optimization purposes. The objective is to minimize the overall

cost rate, including the fixed vehicle costs, route-specific costs and inventory costs at the customers being periodically replenished. The last two cost items depend on the selected/modified frequency of the routes within a lower and upper bound for the frequencies. Similar to what *Archetti et al. (2007)* and *Solyali & Süral (2011)* proposed for a single vehicle case of the IRP, the Single-Vehicle Cyclic Inventory Routing Problem (SV-CIRP) occurs naturally as a subproblem of the CIRP. This is when one vehicle is considered for replenishing and when not all customers need to be visited *(Zhong & Aghezzaf 2011; Aghezzaf et al. 2012; Vansteenwegen & Mateo 2014)*.

In order to assure that IRP solutions are also acceptable for practitioners, "consistency" features are introduced. Consistency is discussed extensively in *Coelho et al. (2012)*. One example is that irregular visits to customers should be avoided. As we will discuss further in this section, the CIRP model proposes a cyclic and fixed visit schedule for each customer. Another consistency feature is that customers prefer no large variations in the amounts delivered. The CIRP model guarantees equal delivery quantities for each customer. A third element is that different visits to a certain customer should be performed by the same driver (or in terms of the mathematical modeling, the same vehicle). Again, the CIRP assumes a fixed partition of the customers and assigns each subset to one driver (vehicle). In addition to the consistency features, as mentioned before, the CIRP model takes fixed vehicle costs into account and also the fact that a single vehicle can make more than one trip in a certain period.

CIRP has some more complicating features; as the demand rate of each customer is assumed to be constant, the time between two visits to a customer (visiting frequency) also determines the amount that should be delivered. This significantly complicates the vehicle capacity constraint. Dividing the customers in trips and integrating this with assigning trips to the minimum number or required vehicles is highly complex. The problem of assigning customers to different trips and trips to vehicles and deciding on the cycle time for the trips and vehicles is a real challenge. Moreover, the problem of assigning trips to vehicles and of making a trade-off between transportation, inventory and fixed vehicle costs should be solved *(Raa 2015)*. Together with an infinite planning horizon, this will result in a CIRP, with a cyclic distribution pattern for each vehicle. This distribution pattern consists of different trips with different frequencies *(Aghezzaf et al. 2006)*. The latter is an important aspect of the CIRP which distinguishes it from other variants of the IRP. Also, when inventory capacities at the customers are smaller than the vehicle capacity, the IRP becomes significantly more difficult *(Bertazzi et al. 2008)*. In addition to all these, as we will discuss, the CIRP objective function is nonlinear, non-continuous piecewise and therefore, non-differentiable. That is why solving the CIRP is highly complex *(Haughton 2013)*.

In *Raa & Aghezzaf (2009)*, a heuristic algorithm based on column generation is implemented to solve the CIRP. Considering a cyclic planning approach where a long-term distribution pattern can be derived, they have developed an algorithm allowing vehicles to perform multiple trips. Initially, customers are partitioned over vehicles using a column generation algorithm. Then, for each vehicle, the set of assigned customers is partitioned over different trips for which frequencies are then determined. For each partition of customers over trips and each combination of trip frequencies, a

delivery schedule is then made to check feasibility. This is the state-of-the-art approach for the CIRP. The SV-CIRP is also approached by the researchers in order to develop exact and metaheuristic algorithms for the solution of this problem. As the first step in this process, the formulation of the SV-CIRP removes the nonlinearity in the constraints of the original subproblem formulation. However, the objective function is inherently nonlinear and remains nonlinear in the proposed formulation. Nevertheless, *Aghezzaf et al. (2012)* succeeded to develop an exact algorithm for the SV-CIRP. Average computational times for solving small test instances (only 20 customers) by the exact method vary from around a quarter to one hour or more. The complexity of the problem makes it very hard to solve large instances to proven optimality in most cases. *Zhong & Aghezzaf (2012)* and *Vansteenwegen & Mateo (2014)* presented heuristic algorithms to reach near optimum solutions for the SV-CIRP.

4. Mathematical model

CIRP assumptions were initially proposed by *Larson (1988)* and *Viswanathan & Mathur (1997)*. *Aghezzaf et al. (2006)* added the multi-trip concept to the model and further *Raa and Aghezzaf (2009)* considered the case of cargo handling (loading and unloading) times both at the depot and customers, although the associated costs were already considered in *Aghezzaf et al. (2006)*. Also, *Raa and Aghezzaf (2009)* added the customer inventory capacities and maximum driving time for the vehicles in their study. These practical assumptions make the mixed integer model very complicated. That is probably why *Raa and Aghezzaf (2009)* only presented the (lower and upper) bounds on the vehicle cycle time rather than the entire model. As an attempt to close the modeling gap, here we present the nonlinear non-convex mathematical formulation of the problem (CIRP). To ease the reading of the text we use node instead of customer from this point forward in the paper.

Let N_p^+ be the set of nodes ($i \in N_p$ and the depot) in trip $p \in P_v$, where P_v is the set of trips in the schedule of vehicle $v \in V$. Let the decision variable x_{ij}^p be a binary variable equal to 1, if node $i \in N_p^+$ was visited immediately before node $j \in N_p^+$ in trip $p \in P_v$ by the vehicle $v \in V$, and 0, otherwise. Constraints (1) limit the in-degree to one for all vehicles and incoming arcs of each node. Constraints (2) balance the in-degree and out-degree for all nodes. Also, constraints (1) and (2) together prevent the split delivery and force each node to belong to exactly one trip and therefore only one vehicle schedule. This implements the driver consistency, discussed in the literature section. Constraints (3) state that for each trip we need a vehicle to be dispatched from the depot.

$$\sum_{v \in V} \sum_{p \in P_v} \sum_{i \in N_p^+} x_{ij}^p = 1 \qquad \forall\, j \in N \qquad (1)$$

$$\sum_{i \in N_p^+} x_{ij}^p - \sum_{k \in N_p^+} x_{jk}^p = 0 \qquad \forall\, j \in N_p^+, p \in P_v, v \in V \qquad (2)$$

$$x_{0j}^p - y^v \leq 0 \qquad \forall\ j \in N, p \in P_v, v \in V \qquad (3)$$

Suppose that the decision variable z_{ij}^p is the sum of demand rates (units per day) of the remaining nodes in a trip $p \in P_v$ served by vehicle $v \in V$ when it travels to node $j \in N_p^+$ immediately

after it has visited node $i \in N_p^+$. This variable equals zero when the trip is not served by vehicle $v \in V$ or the vehicle is returning back to depot after visiting the last node in the trip. Constraints (4) are the flow conservation constraints assuring that the difference in cumulated demand rates of remaining nodes before and after visiting a certain node will be equal to the demand rate of that node. These constraints also prevent sub-tours. Constraints (5) relate z_{ij}^p and x_{ij}^p by forcing x_{ij}^p equal to 1 if z_{ij}^p has to accommodate some demand. Every execution of the trip p will be done with the same z_{ij}^p forcing the consistency in the delivered quantities to the customers.

$$\sum_{v \in V} \sum_{p \in P_v} \sum_{i \in N_p^+} z_{ij}^p - \sum_{v \in V} \sum_{p \in P_v} \sum_{k \in N_p^+} z_{jk}^p = d_j \qquad \forall\, j \in N \tag{4}$$

$$z_{ij}^p - (\sum_{k \in N} d_k) x_{ij}^p \leq 0 \qquad \forall\, i, j \in N^+, p \in P_v, v \in V \tag{5}$$

Each trip ($p \in P_v$) schedule has limits on its cycle time, $\Gamma^p \in \mathbb{N}$, which is expressed in days and is a decision variable. Constraints (6) impose, for each trip $p \in P_v$, a maximum, H, on the sum of the driving time for the TSP route and the cargo handling (loading at the depot and unloading at the nodes) time for visiting all its nodes. This also implies a lower bound of 1 day for each trip cycle time ($1 \leq \Gamma^p$). Furthermore, in each trip, the sum of the deliveries over all visited nodes is limited by the vehicle capacity. Also, the trip cycle time is bounded from above by the capacity of the vehicle and the nodes visited in the trip. The nonlinear constraints (7) and (8) introduce these upper bounds for the trip cycle time. Capacity of the vehicle in carrying the cumulative demand of the nodes in the trip is considered in constraints (7) forcing the vehicle to visit all the nodes in its schedule before the stockout. Each node's inventory capacity also imposes a similar limit to the maximum duration of the trip cycle time (constraints 8). Constraints (8) also force equal trip cycle times of Γ^p for each trip p or the consistency in customer visit spacing.

$$\sum_{i \in N_p^+} \sum_{j \in N_p^+} t_{ij} x_{ij}^p + \sum_{i \in N_p^+} \theta_i \leq H \qquad \forall\, p \in P_v, v \in V \tag{6}$$

$$\Gamma^p z_{0i}^p \leq \kappa \qquad \forall\ i \in N_p, p \in P_v, v \in V \tag{7}$$

$$\Gamma^p d_i \leq k_i \qquad \forall\ i \in N_p, p \in P_v, v \in V \tag{8}$$

There are also some limits associated with the vehicle ($v \in V$) schedule cycle time, $T^v \in \mathbb{N}$, which is expressed in days as well and is a decision variable. The nonlinear constraints (9) state that every vehicle schedule cycle time should be an integer multiple of each of its trips' cycle time. In other words, for a vehicle schedule to be feasible and complete, the frequency (f^p) of every trip in its schedule, which is a decision variable, has to be an integer number. These constraints imply that the T^v should be equal to the least common multiple of all trip cycle times and no vehicle schedule cycle time can be less than the cycle time of its trips since $f^p \in \mathbb{N}$. If some trips eventually have to be executed in a certain day of the same vehicle schedule due to their cycle times, then their total duration should be bounded by the maximum driving time, H. In other words, for every subset of trips ($p \in P, P \subseteq P_v, |P| \geq 2$) in a vehicle schedule, the nonlinear constraints (10) should be valid in

order to have a feasible vehicle schedule. These inequalities imply that if the greatest common divisor (gcd) of the frequency of some trips is equal to one, then those trips will have a common day in the vehicle schedule meaning that the vehicle needs to execute them within its available driving time limit. We discuss this issue in detail later in Section (5.4.2) with an illustrative example. There is an exponential number of these inequalities since they are formulated for all the subsets of the trips.

$$T^v = f^p \Gamma^p \qquad \forall\, p \in P_v, v \in V \tag{9}$$

$$\sum_{p \in P} \left(\sum_{i \in N_p^+} \sum_{j \in N_p^+} t_{ij} x_{ij}^p + \sum_{i \in N_p^+} \theta_i \right) - H.gcd_p(f^p) \leq 0 \qquad \forall\, p \in P, P \subseteq P_v, |P| \geq 2, v \in V \tag{10}$$

The appropriate objective function (11) contains the long-run distribution and inventory management cost rates. These cost rates include four items each expressed in currency/day. The first one is the fixed vehicle cost of ψ unit per day and is applied for each used vehicle (y^v). Considering the average vehicle speed of σ (distance/hour) and the vehicle operating costs of δ (currency/distance), every execution of a trip p, with route duration $\sum_{i \in N_p^+} \sum_{j \in N_p^+} t_{ij} x_{ij}^p$ (hours per one trip execution) will have a transportation cost of $\delta\sigma \sum_{i \in N_p^+} \sum_{j \in N_p^+} t_{ij} x_{ij}^p$ for a single execution of the trip. This term needs to be multiplied with f^p to determine the transportation cost for all its iterations in one vehicle cycle time. This gives the second cost item which is the variable transportation cost of $\frac{\delta\sigma}{T^v} \sum_{p \in P_v} f^p \sum_{i \in N_p^+} \sum_{j \in N_p^+} t_{ij} x_{ij}^p$ per day of each vehicle schedule. The third cost item is the fixed vehicle loading (at the depot) and unloading (at the nodes) of φ_i which will be applied for each dispatch from the depot and node visit. The last item is the inventory cost at the nodes. The quantity delivered to each node i covers the demand for $\frac{T^v}{f^p}$ days up to the next delivery which will be equal to $\frac{T^v}{f^p} d_i$ units. The average inventory level at node i during this period is $\frac{T^v d_i}{2f^p}$ which gives the cost of $\frac{T^v \eta_i \mathrm{d_i}}{2f^p}$ for each node and $\sum_{i \in N_p} \frac{T^v \eta_i \mathrm{d_i}}{2f^p}$ for each trip. Then, the total cost of the distribution pattern including all the vehicles can be written as the following nonlinear non-convex function.

$$\min\ C = \sum_{v \in V} \left[\psi y^v + \frac{\delta\sigma}{T^v} \sum_{p \in P_v} f^p \sum_{i \in N_p^+} \sum_{j \in N_p^+} t_{ij} x_{ij}^p + \frac{1}{T^v} \sum_{p \in P_v} f^p \sum_{i \in N_p^+} \varphi_i + \frac{T^v}{2} \sum_{p \in P_v} \frac{1}{f^p} \sum_{i \in N_p} \eta_i d_i \right] \tag{11}$$

Based on this, the mathematical formulation of the CIRP is to minimize the cost rate function (11) subject to the constraints (1) to (10) while the decision variables are as shown in (12).

$$\begin{Bmatrix} y^v \in \{0,1\}, T^v, \in \mathbb{N} & \forall\, v \in V \\ \Gamma^p, f^p \in \mathbb{N} & \forall\, p \in P \\ x_{ij}^p \in \{0,1\}, z_{ij}^p \geq 0 & \forall\, p \in P, \forall\, i,j \in N^+ \end{Bmatrix} \tag{12}$$

5. Two-phase Algorithm

In this section, we discuss in detail our two-phase heuristic to tackle the CIRP. As mentioned above, we decompose the problem into two subproblems and iteratively try to improve the solution quality in order to reach a local optimum. Then a simple local branching scheme is used to explore other parts of the feasible area, looking for better solutions. We start this section with presenting some insights in the complexity of the problem and the consequences of these insights for our solution approach. Then we discuss the general structure of our algorithm and all the different components.

5.1. Fast cost estimation

The complexity of the problem makes it impractical to try to solve the mixed integer nonlinear programming formulation of the CIRP directly for all but the smallest instances *(Aghezzaf et al. 2006)*. The structure of the problem, though, argues for some type of decomposition algorithm. Similar problems such as IRP *(Campbell & Savelsbergh 2004)* and production routing problem *(Absi et al. 2013)* are also approached with two-phase decomposition algorithms. The nonlinear mixed integer problem and the non-convex nature of the feasible solution space *(Zhong & Aghezzaf 2011; Vansteenwegen & Mateo 2014)* demand an algorithm that should be capable of investigating as many local optima as possible. Since each local optimum exploration takes a huge computational and algorithmic effort regard to the complex problem structure, a fast solution value estimation procedure is our key to overcome this complexity.

The core idea of this estimation is based on the linear relaxation of the required vehicles (y^v) to replenish all the nodes. First, we introduce the trip utilization (u^p) out of a vehicle schedule which equals to the share of the trip p duration (considering its frequency, f^p) out of a vehicle schedule to drive around and visit all nodes of that trip, or:

$$u^p = \frac{f^p\left(\Gamma^p_{tsp}\right)/H}{T^v} \qquad \forall\, p \in P \qquad (13)$$

Where $\Gamma^p_{tsp} = \sum_{i\in N^+_p}\sum_{j\in N^+_p} t_{ij}x^p_{ij} + \sum_{i\in N^+_p}\theta_i$. When we relax the integrality constraint of the number of the necessary vehicles ($y^v \in [0,1]$), then it is trivial to see that the amount of necessary vehicles to execute all the trips is equal to $\sum_{p\in P} u^p$. Also, from constraints (9) we know that $\Gamma^p = T^v/f^p$. These help us to rewrite the objective function as follows (when the integrality constraint on the number of vehicles is relaxed):

$$C = \sum_{v\in V}\left[\psi\sum_{p\in P_v} u^p + \delta\sigma\sum_{p\in P_v}\frac{1}{\Gamma^p}\sum_{i\in N^+_p}\sum_{j\in N^+_p} t_{ij}x^p_{ij} + \sum_{p\in P_v}\frac{1}{\Gamma^p}\sum_{i\in N^+_p}\varphi_i + \frac{1}{2}\sum_{p\in P_v}\Gamma^p\sum_{i\in N_p}\eta_i d_i\right] \qquad (14)$$

or:

$$C = \sum_{v \in V} \sum_{p \in P_v} \left[\psi u^p + \frac{\delta \sigma}{\Gamma^p} \sum_{i \in N_p^+} \sum_{j \in N_p^+} t_{ij} x_{ij}^p + \frac{1}{\Gamma^p} \sum_{i \in N_p^+} \varphi_i + \frac{\Gamma^p}{2} \sum_{i \in N_p} \eta_i d_i \right] \quad (15)$$

What is inside the brackets in the relaxed objective function (15) is actually the trip execution cost, E^p. Therefore, the cost of a solution can be estimated as the sum of the costs of all its trips ($\sum_{p \in P} E^p$), according to their trip cycle time (Γ^p) and the fixed costs we need to pay for the minimum necessary vehicles (u^p) to execute all the trip schedules. Within this assumption and reformulation of the objective function, we will be able to avoid some important sources of the problem complexity; necessary vehicles (y^v), assignment of the trips to the vehicles (P_v) and their schedule cycle times (T^v) as well as the trip frequencies (f^p). This leads us to an approximation of the objective function instead of its exact value but within considerably less computational time.

Let Γ_{max}^p be the maximum cycle time of the trip p. According to the constraints (7) and (8) we can obtain this upper bound for every trip as presented in equations (16).

$$\Gamma_{max}^p = min \left\{ \left\lfloor \min_{i \in N_p} \frac{k_i}{d_i} \right\rfloor, \left\lfloor \frac{\kappa}{\sum_{i \in N_p} d_i} \right\rfloor \right\} \qquad \forall\, p \in P \qquad (16)$$

We define a lower bound on the trip utilization ($\hat{u}^p$) out of a vehicle schedule based on the maximum cycle time limit (or minimum trip execution frequency) by using $\hat{\Gamma}^p = \Gamma_{max}^p$. Then we can write:

$$\hat{u}^p = \frac{\Gamma_{tsp}^p / H}{\hat{\Gamma}^p} \qquad \forall\, p \in P \qquad (17)$$

As an example, consider the nodes 1 to 7 in 3 different trips as illustrated in Table 1 with given demand rates (d_i), storage capacities (k_i), TSP route duration (Γ_{tsp}^p) and the vehicle capacity of $\kappa = 30$ units. We can calculate the maximum cycle time for each trip (Γ_{max}^p) and then trip utilization ($\hat{u}^p$) according to the equations (16) and (17) as presented in this table. It is interesting to notice that while the first node can be visited every 6 days according to its demand rate and storage capacity ($\frac{k_i}{d_i} = 6$), the limited vehicle capacity ($\frac{\kappa}{\sum_{i \in N_p} d_i} = 3$) forces more frequent visits to this node since it is combined with two more nodes into one trip. Also, although the second trip can be executed every 5 days based on the vehicle capacity ($\frac{\kappa}{\sum_{i \in N_p} d_i} = 5$), the demand rate and storage capacity of node 4 limits the maximum cycle time of this trip to 4 days ($\frac{k_i}{d_i} = 4$). This simple example reveals the fact that transferring a node from one trip to another trip not only changes the TSP route durations but also changes the maximum cycle time of both trips engaged in the transfer.

Table 1. An example of different nodes combined in different trips

Trip (p)	Node (i)	Demand rate (d_i)	Storage capacity (k_i)	Γ_{tsp}^{p} (hours)	$\frac{k_i}{d_i}$	$\min_{i \in N_p} \frac{k_i}{d_i}$	$\sum_{i \in N_p} d_i$	$\frac{\kappa}{\sum_{i \in N_p} d_i}$	Γ_{max}^{p} (days)	$\hat{u}^p = \frac{\Gamma_{tsp}^{p}/H}{\Gamma_{max}^{p}}$
1	1	2	12	5	6	6	10	3	3	0.208
	2	4	32		8					
	3	4	40		10					
2	4	2	8	5	4	4	6	5	4	0.156
	5	4	24		6					
3	6	1	7	6	7	7	3	10	7	0.107
	7	2	20		10					

Having the $\hat{\Gamma}^p$ and $\hat{u}^p$, the trip execution cost estimation, $\hat{E}^p$, can be calculated using equation (18).

$$\hat{E}^p = \psi \hat{u}^p + \frac{\delta \sigma}{\hat{\Gamma}^p} \sum_{i \in N_p^+} \sum_{j \in N_p^+} t_{ij} x_{ij}^p + \frac{1}{\hat{\Gamma}^p} \sum_{i \in N_p^+} \varphi_i + \frac{\hat{\Gamma}^p}{2} \sum_{i \in N_p} \eta_i d_i \qquad \forall\, p \in P \qquad (18)$$

This fast trip cost estimation function (18) can efficiently be used to evaluate the potential cost saving of moving a node from one trip to another trip or in other words, the neighborhood evaluation of a current solution. The fast evaluation of moving a node from one trip to another will be discussed in detail in the routing subproblem (section 5.4.1) of our algorithm.

5.2. General structure of the algorithm

Here, we discuss the general structure of our heuristic (presented in ***Algorithm 1***). In the next subsections all building blocks are discussed in detail. The algorithm starts with constructing a feasible initial solution (Section 5.3, Initialization). Then the current trips are improved in the routing subproblem (Section 5.4.1, Routing) based on fast trip cost estimation and neighborhood exploration until no feasible cost saving node transfer can be found. Afterwards, the trips are assigned to feasible schedules for the vehicles in the scheduling subproblem (Section 5.4.2, Scheduling) to obtain the integer number of required vehicles (feasible vehicle schedules). This will enable us to calculate the actual cost of the current solution. The routing and scheduling subproblems are iteratively solved in two integrated loops: an inner loop with routing and scheduling and an outer loop with a Vehicle Decrease heuristic (Section 5.4.4). This integrated process is referred to as Local Search (Section 5.4, Local Search) in the remaining of this paper. Therefore, the Local Search algorithm includes the two main subroutines of Routing and Scheduling plus a heuristic (Vehicle Decrease heuristic) for intensification. Each time a better solution S^* (incumbent solution) is found during the Local Search algorithm, S^* is updated and all the edges that are not adjacent to the depot in the solution S^* will be added to the set B. These arcs will be used as the branching elements for our simple local branching scheme. To branch on an arc of the set B, a large value will be assigned to its travel time and also its opposite direction's travel time. This will prevent the branched element from being selected during the remainder of the search. Then, the Local Search will be used to continue the search based on the current set of trips. Using this branching tool, in each iteration, we branch on one of the edges of the branching set B, in order to escape from the local optimum and to better

explore the search area. This will be done for a maximum number of different edges of the set B, $MaxB$. $MaxB$ is a parameter of the algorithm and will be discussed in Section 6.2. This branching scheme is used as a diversification phase of the algorithm.

$Algorithm\ 1: General\ Framework$

0. ***Initialization***
1. $iterB \leftarrow 0$ & $B \leftarrow \emptyset$
2. Run the ***Local Search*** algorithm, $S^* \leftarrow S$
3. Add all edges $(i,j) \in S^*: i,j \neq 0$, to B
4. **Do** while $B \neq \emptyset$ and $iterB \leq MaxB$
5. | Choose element $b_{IterB} \in B$, branch on this element
6. | Run the ***Local Search*** algorithm
7. | $iterB \leftarrow iterB + 1$
8. **End**
9. Output:S^*

5.3. Initialization

In order to create an initial solution, each node is assigned to a separate trip. Then, for each trip, the TSP travel time and the maximum cycle time are calculated and the trip cost is estimated based on Eq. 13.

5.4. Local Search

The Local Search (***Algorithm 2***) starts by solving the Routing and Scheduling subproblems. Each time a better solution is found, the best found solution, S^*, is updated, all the edges (not adjacent to the depot) of the solution S^* will be added to the set B. Then, the cost estimations for the different trips are updated to be used in the next iteration. The "update trip values" module (Section 5.4.3) is used to update the cost estimation of each trip and the cost estimation of moving each node to another trip. This update will lead to new opportunities for the next routing and scheduling iteration.

$Algorithm\ 2: Local\ Search$

0. iterP $\leftarrow 0, iterU \leftarrow 0$
1. **Do** while $iterP \leq MaxP$
2. | **Do** while $iterU \leq \min\{iterP, MaxU\}$
3. | | Run the ***Routing*** algorithm
4. | | Run the ***Scheduling*** algorithm
5. | | If S^* is updated, reset $iterP$ & $iterU$, add all new edges of S^* to B
6. | | ***Update trip values***
7. | | $iterU \leftarrow iterU + 1$
8. | **End**
9. | Run the ***Vehicle Decrease heuristic***
10. | $iterP \leftarrow iterP + 1, iterU \leftarrow 0$
11. **End**

The experimental results will show that using this update strategy helps our algorithm to find better solutions. The inner loop with Routing, Scheduling and trip value updating, stops when a maximum number of iterations, $min\{iterP, MaxU\}$ is reached. In the outer loop, the Vehicle Decrease heuristic is applied and $iterP$ is increased by one in each iteration. The Vehicle Decrease heuristic (Section 5.4.4) tries to find a distribution pattern with one vehicle less. Both bounds on the number of iterations, $MaxU$, and $MaxP$, will be discussed and set in Section 6.2.

5.4.1. Routing

In this phase, a steepest descent framework is implemented to improve the assignment of nodes to trips (***Algorithm 2-1***). We define a neighborhood of a solution based on all its feasible node (i) transfers from current trips (p_0) to other trips (p). Based on the trip cost estimations function (18), it is determined, for each node and for each trip, which (estimated) total saving can be expected when moving a node to a trip. Obviously, the cost saving associated with its current trip ($\hat{E}^{p_0} - \hat{E}^{p_0-\{i\}}$) as well as its destination trip ($\hat{E}^{p} - \hat{E}^{p\cup\{i\}}$) will be taken into account. The move of a node that leads to the best (estimated) saving ($\hat{E}^{p_0} + \hat{E}^{p} - \hat{E}^{p_0-\{i\}} - \hat{E}^{p\cup\{i\}}$) is selected and actually performed. The way we try to save costs could be considered as a generalization of the *Clarke & Wright (1964)* heuristic to save not only the travel time but also the trip cost estimation.

$Algorithm\ 2-1: Routing$	
0.	For all feasible node transfers, calculate the saving: $\hat{E}^{p_0} + \hat{E}^{p} - \hat{E}^{p_0-\{i\}} - \hat{E}^{p\cup\{i\}}$
1.	**Do** until no cost saving transfer is found
2.	| Execute the node transfer with the best saving
3.	| Remove i^* from p_0 and add i^* to p^*
4.	| Update all savings for p_0 and p^*
5.	**End**
6.	Apply 2-Opt and Relocate

During the routing phase many changes in cost need to be calculated each iteration and that is exactly the reason fast cost estimations (Section 5.1) are used instead of exact calculations. After the node is actually moved, the cost estimations for all the affected trips and nodes are updated and the procedure is repeated until no more cost saving node transfers are found. Notice that one interesting property of the algorithm is that for all other node-to-trip transfers the saving values remain as in the previous iteration and we do not need to update them. The saving update is only necessary for all the trips that had a change in the current iteration (the trip with a leaving node and the trip with an incoming node). Also, for all other nodes, the transfer saving to these two trips should be updated. This process will not get trapped in an infinite loop. Since every node transfer is done based on a strictly greater than zero saving. The routing algorithm works with the trip cost estimations and stops when no improvement in the estimated costs can be found. Therefore, the algorithm will reach a local optimum after a number of finite iterations. At the end of the routing phase, when no more cost saving is possible, the well-known 2-Opt and Relocate *(Laporte 1992;*

Gendreau& Potvin 2010) local search moves are applied to each trip, in order to improve the travel time of the trip. These local search moves are applied (first improvement) until no more improvements can be found.

5.4.2. Scheduling

The scheduling subproblem, about assigning the trips to the vehicles, is related to the well-known problem of bin packing. However, in this case the different frequencies of the deliveries significantly complicate the process. Nevertheless, the scheduling algorithms (***Algorithms 2-2-1*** *and* ***2-2-2***) are based on the best-fit decreasing (BFD) bin packing heuristic *(Coffman et al. 1996)*. A cyclic schedule is required in which the vehicle visits the nodes (divided in trips) in order to prevent stockouts. This should be done such that the time between consecutive iterations of the same trip remains constant, i.e. a cyclic pattern per trip. In addition, two or more trips of a vehicle should either not occur on the same day or the combination of trips should not exceed the maximum driving time of the vehicle per day. Holding these criteria, two or more trips are called "compatible".

Notice that the minimum number of required vehicles for a set of trips to be scheduled is not necessarily equal to the roundup amount of the estimations calculated in the routing phase, it can easily be more. This may happen because of different and non-compatible trip cycle times. For instance, consider trips 1 and 2 in the Table 1 which both require more than half a day of driving time. The first trip has a maximum cycle time of three days and the second has a maximum cycle time of four days. These two trips cannot be assigned to the same vehicle and are non-compatible trips due to the constraints (10). Indeed, assume that the first trip is scheduled on the first day (and therefore the fourth, seventh, tenth, etc.). As a result, the second trip cannot be scheduled on the second day then it would conflict with the first trip on the tenth day. This is illustrated in Figure 1-a, where the second trip has to be executed in the 10^{th} day of the vehicle schedule together with the first trip which also requires five hours. The execution of these two trips needs a total driving time of 10 hours for the same vehicle. Also, it is not feasible to schedule the second trip on the third day (then it would conflict with the first trip on the seventh day) or any other day of this vehicle. Sticking to these trip cycle times ($\Gamma^1 = 3$ and $\Gamma^2 = 4$), we need two vehicles to execute these two trips as drawn in Figures 1-b and 1-c. If the cycle time of the first trip would be (changed to) two days, $\Gamma^1 = 2$, the trips would be compatible and can be assigned to the odd and even days of a vehicle schedule (Figure 1-d). It is also possible to change the cycle time of the second trip to three days ($\Gamma^2 = 3$) and schedule both trips as illustrated in Figure 1-e. In other words, the cycle time of three, instead of four, for the second trip makes it possible to combine these two trips into one vehicle schedule. Reducing the cycle time of any of the trips (Figures 1-d and 1-e) results in more frequent trip execution during the vehicle schedule and hence more transportation and cargo handling costs but it also saves one vehicle to schedule both trips.

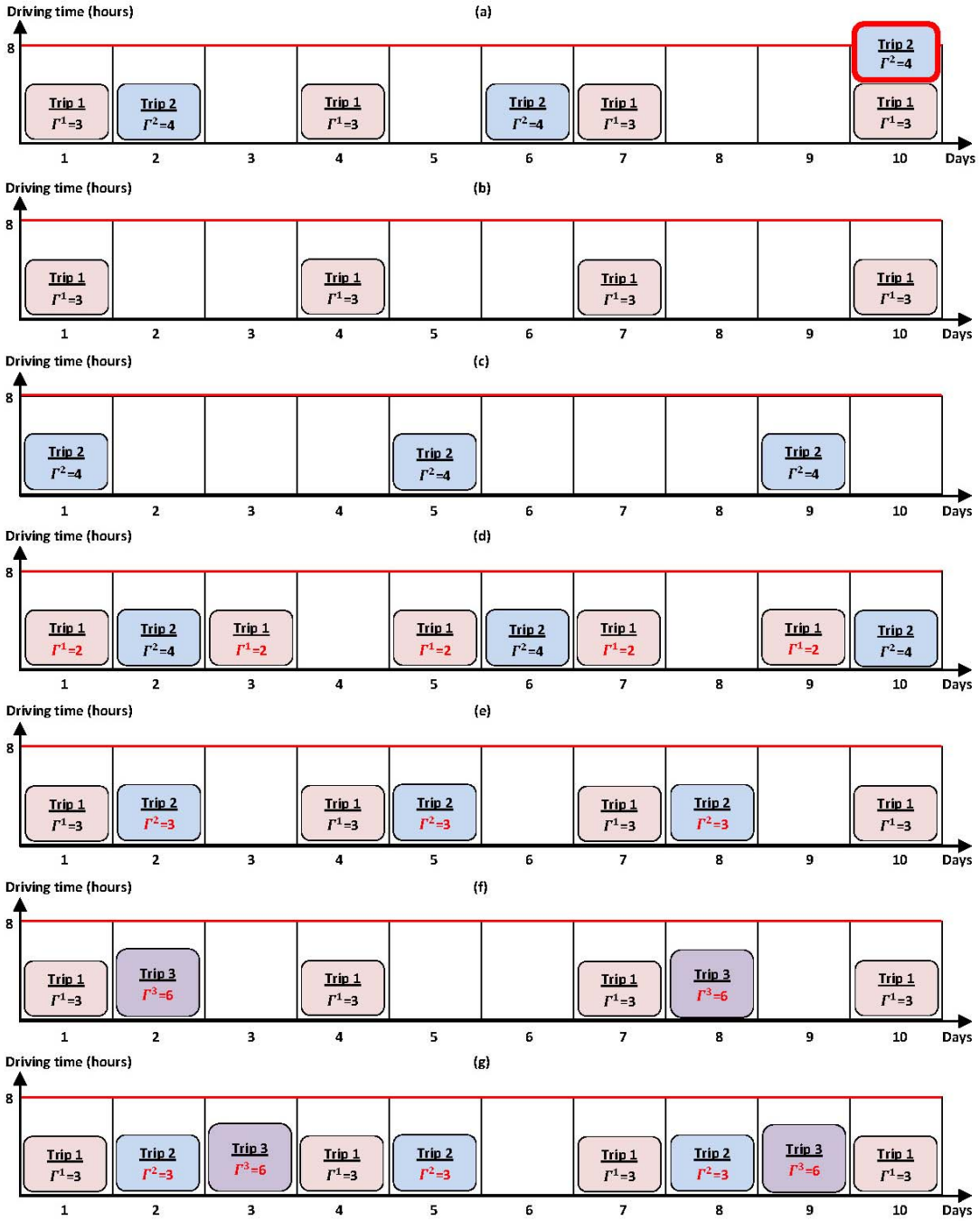

Figure 1. Different trips and vehicle schedules

a) an infeasible schedule due to the incompatible trips; b) and c) feasible schedules for two vehicles each with only one trip; d) feasible schedule for trips 1 and 2 in one vehicle after changing the cycle time of trip 1 from 3 to 2; e) feasible schedule for trips 1 and 2 in one vehicle after changing the cycle time of trip 2 from 4 to 3; f) feasible schedule for trips 1 and 3 in one vehicle after changing the cycle time of trip 3 from 7 to 6; g) feasible schedule for trips 1, 2 and 3 in one vehicle after changing the 2nd and 3rd trip cycle times to 3 and 6, respectively;

Solving this subproblem of scheduling to optimality is computationally expensive. Actually, *Raa (2015)* wrote an entire paper about only this part of the CIRP. Therefore, we propose a fast greedy

heuristic. In each iteration, the algorithm starts with the current set of trips and an empty schedule for every vehicle. Then, it starts with the first vehicle and tries to add the trips one by one until no more trips can be added to this vehicle. This procedure fills the days of the vehicle schedule one by one. The procedure continues with a new vehicle until all trips are scheduled. Since each extra vehicle in the solution brings a significant extra cost, each included vehicle should be filled as much as possible. Therefore, we use the maximal cycle time of each trip (Γ^p_{max}), instead of the EOQ or optimal cycle time, when filling the vehicle schedules. In this way, more trips can be added to each vehicle.

To fill a vehicle schedule, the algorithm starts by selecting the trip with the smallest maximum cycle time ($min_p\ \Gamma^p_{max}$) as the base trip for developing the vehicle schedule. Notice that the one with the shortest cycle time needs to be visited more frequently, thus it will fill the vehicle schedule more than any other trip and therefore it is interesting (greedy) to start with this trip. In all cases, if there would be a tie in maximum cycle time of some trips, it selects the trip with the maximum travel time (again, since that trip will fill the schedule the most). Based on this first trip (p^*) and its cycle time ($\Gamma^{p^*} = \Gamma^{p^*}_{max}$), the vehicle will have other days available that can only be filled with trips with a cycle time ***compatible*** with $\Gamma^{p^*}_{max}$ days. For example, in Figure 1-b, the first trip has the maximum cycle time of three days and needs five hours of travel time. By selecting this trip first in the vehicle schedule, it will leave two whole days available in the vehicle schedule and three hours in the same day of its schedule available for other trips. The next compatible trip to be added to this vehicle schedule should have a cycle time of three or a multiple of it. Each day in the vehicle schedule should have the same base cycle time (T^{base}) to keep the schedule feasible. In this example, the base cycle time is three. Therefore, for the remaining vehicle driving time in every day of the vehicle schedule, again, the algorithm selects the compatible trip with the smallest cycle time that fills the remaining schedule the most. After examining a wide range of different functions evaluating the compatibility, a simple function is selected to find a proper trip to be added to the vehicle schedule in $O(n)$: It is clear that the best fitting trip is the one with a cycle time equal to T^{base} and a travel time that most fits the remaining vehicle driving time in the specific day of the vehicle schedule. Therefore, among all feasible trips for a specific day, nearest maximum cycle times as well as longer travel times (or smaller $1/\Gamma^p_{tsp}$) are preferred. In case there would be no trip available with a similar maximum cycle time, one with a longer (compatible) trip cycle time should be selected. Otherwise we need to modify (decrease) the trip cycle time to make it compatible. This can be stated as choosing a trip with the smallest $\Gamma^p_{max}\ mod\ T^{base}$. Note that no shorter cycle times are available since the smallest one was selected as the base trip cycle time.

For example, suppose that we have already added trip 1 to a vehicle schedule as in Figure 1-b and need to schedule trips 2 and 3 (from Table 1) as well. Trip 2 has a travel time of 5 hours and a maximum cycle time of 4 days (Figure 1-c) while trip 3 has a travel time of 6 hours and a maximum cycle time of 7 days. Adding the first trip to the vehicle schedule, it imposes a base cycle time of 3 days ($T^{base} = 3$) on the vehicle schedule. The algorithm needs to select one of them first to add to the schedule. If trip 2 is selected we have $\Gamma^2_{max}\ mod\ T^{base} = 1$ and $\frac{\Gamma^2_{max}}{T^{base}} = 4/3$. Therefore, its cycle

time should be modified to 3 to be both feasible and compatible with this schedule (Figure 1-e). Similarly, if trip 3 is selected we have $\Gamma^3_{max}\ mod\ T^{base} = 1$ and $\frac{\Gamma^3_{max}}{T^{base}} = 7/3$. Then its cycle time can be modified to 6 to be compatible with the current schedule of the Figure 1-b resulting in the schedule of the Figure 1-f. Note that adding trip 3 with the cycle time of 6 to the vehicle schedule will split that specific day of the schedule in two new days (days 2, 8, 14, etc and days 5, 11, 17, etc) each with a new base cycle time of 6 while it fills just one of these new spaces (Figure 1-f). Then the rest of these days need to be scheduled with trips compatible with this new base cycle time of 6 (for these days of the schedule). Generally, trip 2 should be preferred over trip 3, since, in the context of bin packing, it can be considered as a bigger item to be picked up. As shown in Figure 1-g it is also possible to add all trips into one vehicle schedule with modified cycle times. This results in having less vehicles for scheduling non compatible trips.

For the purpose of developing an acceptable criterion, several possible functions were developed and tested. Finally, for each day, the best compatible trip $p^*$is chosen with the minimum value of $\left[\Gamma^p_{max}\ mod\ \Gamma^{base} + \mu\ int\left(\Gamma^p_{max}/T^{base}\right) + 1/\Gamma^p_{tsp}\right]$. To make it possible to distinguish between the amount we need to modify a trip cycle time and the times a trip cycle time differs from the base, the first two terms are added to the formula but with different weights (μ is a weight adjustment parameter). Since in case of a tie between trip cycle times, the trips with bigger travel times are preferred, the last term is added to the formula. Then the trip maximum cycle time will be modified using function (19) to obtain a compatible trip cycle time. For example, if trips with maximum cycle times of four, five, seven and ten need to be scheduled on a day with base cycle time of three, function (19) modifies their cycle times to three, three, six and nine respectively.

$$\Gamma^{p^*} \leftarrow \Gamma^{p^*}_{max} - \Gamma^{p^*}_{max}\ mod\ T^{base} \qquad (19)$$

Each time a trip $p^*$is added to a vehicle schedule, the vehicle schedule cycle time (T^v) will be changed to the least common multiple (lcm) of all trip cycle times ($p \in P_v$) to reach to a full length vehicle schedule. ***Algorithm 2-2-1*** shows the general structure of the first scheduling heuristic. For the instances where the number of required vehicles is smaller than or equal to three, an alternative algorithm to solve the scheduling problem is designed. In this alternative, the focus lies on finding solutions with only one or two vehicles. This second scheduling heuristic (***Algorithm 2-2-2***) will benefit from the fact that trips with even values of cycle times are always compatible. After scheduling the trips with a maximum cycle time of one day (the minimal cycle time), this algorithm partitions the vehicle schedule into two parts: odd and even days. It reduces all the odd values of trip maximum cycle times by one. This will result in even maximum cycle times for all trips. Then, it starts filling up the even days of the schedule. When no more trips could be added to the even days, it continues with filling the odd days. This is the only difference with the original algorithm. Generally, the first version works better for larger instances, with more trips and more possibilities for fitting the trips into vehicle schedules. The alternative will work better for smaller instances with only a few vehicles.

Algorithm 2 − 2 − 1: First scheduling heuristic

I-0. P': set of all current trips
I-1. Pick the first vehicle $v \leftarrow 1$
I-2. **Do** until all trips are scheduled ($P' = \emptyset$)
I-3. | Find trip $p^*$with minimum Γ_{max}^{p} cycle time
I-4. | remove p^* from $P'$and add $p^*$to vehicle v
I-5. | Vehicle schedule cycle time $\Gamma^v \leftarrow \Gamma_{max}^{p^*}$
I-6. | **Do** until no trip $p \in P'$can be added to vehicle v
I-7. | | **Do** until no trip $p \in P'$ can be added to the current day
I-8. | | | Select the next trip:

$$p^* \leftarrow argmin_p \left[\Gamma_{max}^{p} \; mod \; T^{base} + \mu \; int\left(\frac{\Gamma_{max}^{p}}{T^{base}}\right) + \frac{1}{\Gamma_{tsp}^{p}}\right]$$

I-9. | | | Remove p^* from P' and add p^* to the current day
I-10. | | | Modify the cycle time of the selected trip (if necessary):
Calculate $T^{p^*}$from using function (19)
I-11. | | | Update vehicle cycle time (if necessary): $T^v \leftarrow lcm\left(T^v, \Gamma^{p^*}\right)$
I-12. | | **End**
I-13. | | Choose next non-scheduled day in vehicle v
I-14. | **End**
I-15. | Choose a new vehicle: $v \leftarrow v + 1$
I-16. **End**

Algorithm 2 − 2 − 2: Second scheduling heuristic

II-0. P': set of all trips with $\Gamma_{max}^{p} = 1$
II-1. Schedule all trips $p \in P'$ by the first scheduling heuristic
II-2. $P' \leftarrow P - P'$
II-3. For all trips $p \in P'$ with an odd value of Γ_{max}^{p}, $\Gamma_{max}^{p} \leftarrow \Gamma_{max}^{p} - 1$
II-4. Lines I-1 to I-5
II-5. **Do** until no trip $p \in P'$ can be added to the even days of vehicle v
II-6. | Lines I-7 to I-12
II-7. | Choose next non-scheduled even day in vehicle v
II-8. **End**
II-9. **Do** until no trip $p \in P'$ can be added to the odd days of vehicle v
II-10. | Lines I-7 to I-12
II-11. | Choose next non-scheduled odd day in vehicle v
II-12. **End**
II-13. Lines I-15 to I-16

5.4.3. Update trip values

As a result of the scheduling algorithm, after all trips were scheduled, a feasible solution is determined and the exact cost of each trip can be calculated. At this point, the question remains if

nodes can be moved between trips in order to reduce the total cost further. To answer this question rapidly, the costs of these transfers should be calculated. Therefore, the "trip utilization" and the "maximum trip cycle time" need to be determined for all trips and nodes for the next iteration based on the latest available information. This information includes the trip utilization (u^p) and cycle time (T^p) for each trip at the end of the scheduling phase. Notice that the transfer of any node will change the trip cycle times of the involved trips. As a result, the involved vehicle schedules could become infeasible. Therefore, in the next iteration, the algorithm will use newly estimated values, based on the previous iteration, to estimate the costs.

$$u^p = \frac{\hat{u}^p}{\sum_{p \in P_v} \hat{u}^p} \quad \forall p \in P_v, v \in V \qquad (20)$$

The example of three trips in Table 1 and Figure 1 clarifies this. Suppose that these three trips are the only ones on the vehicle schedule as illustrated in Figure 1-g. Each of these trips has its own $\hat{u}^p$ as presented in the last column of the Table 1. Using equation (20), for each of these three trips u^p equals 44.2%, 33.1% and 22.7%, respectively. In this way, the fixed vehicle cost is splitted over the trips in its schedule proportional to their share of the schedule even when they do not occupy the entire vehicle schedule. Then the algorithm modifies the "trip utilization" estimations for all trips to be used for the next iteration as presented in equation (21).

$$\hat{u}^p = u^p \quad \forall p \in P_v, v \in V \qquad (21)$$

Furthermore, for each trip, the estimated maximum trip cycle time in the next iteration is set equal to its current cycle time. These new limits (upper bound for the cycle time and lower bound for the utilization) will affect the cost estimation (equation 18) of the trips during the next iteration. The third and last step is to assign, in each trip, these same upper and lower trip bounds to each individual node in the trip. This is to force them to temporarily hold the current bounds while moving between trips, instead of using many time consuming calculations during each iteration.

5.4.4. Vehicle Decrease heuristic

This heuristic tries to decrease the number of necessary vehicles by one. As the fixed vehicle costs are rather high, reducing the necessary vehicles in most cases can significantly decrease the total cost. This algorithm selects the vehicle with the minimum utility. Assume that $\Gamma_{max}^{Single\ i}$ is the maximum cycle time of serving node i in a separate trip.Due to the vehicle capacity, this cycle time is always greater than or equal to the maximum cycle time of any trip (Γ_{max}^{p}) including node i. All the nodes i of the vehicle with minimal utility are sorted according to the increasing rate of $(\Gamma_{max}^{Single\ i} - \Gamma_{max}^{p})/\Gamma_{max}^{p}$. Then, the first node in the sorted list, i^*, will be selected for transfer to one of the existing trips or an empty trip of another vehicle (empty space in another vehicle schedule). This is only considered if this vehicle's schedule does not need to be changed due to this move. When it fails to find an option for transferring node i^*, the whole procedure stops and it is assumed that this vehicle cannot be removed from the solution. Otherwise it continues to try to transfer the other nodes from the sorted list, one by one. Moreover, after transferring every ρ% of the nodes, the

algorithm will try to schedule the whole set of modified trips with one less vehicle. If this is feasible, the heuristic was successful and finishes. If not, the heuristic continues with the current solution and tries to transfer the second ρ% of the nodes to other vehicles. (ρ is a parameter of the algorithm and is discussed in Section 6.2)

6. Computational results

This section first describes the benchmark instances that were used to evaluate the performance of the heuristic and the setting of the parameters. Then, the performance of the algorithm on all the benchmark instances will be presented.

6.1. Set of benchmark instances

Almost all well-known IRP benchmark instances in the literature belong to the finite time (rolling) horizon versions of the problem (e.g. *Archetti et al. (2007)* and *Archetti et al. (2012)*) and cannot be used as benchmark instances for the CIRP. The only available benchmark instances of the CIRP are published by *Raa (2006)*. In *Raa (2006)*, 320 benchmark instances are presented for CIRP. *Raa & Aghezzaf (2009)* present the results they obtained on 80 out of these 320 benchmark instances. To assess the performance of our algorithm, we compare it with the results of the best existing solution approach in the literature, based on these 320 benchmark instances. The instances are available at *www.mech.kuleuven.be/en/cib/op* and fully described in both mentioned references.

6.2. Parameter setting

Our algorithm has 5 parameters to be set. $MaxU$ determines the number of times the inner loop in the Local Search, with Routing and Scheduling is run consecutively. $MaxP$ is the number of times that the outer loop, with the Vehicle Decrease heuristic, will be used as an intensification tool to improve the resulting solution of Routing and Scheduling in the Local Search. Within each execution of this Vehicle Decrease heuristic, after transferring every ρ% of the nodes, the scheduling algorithm will be applied in order to reduce the number of vehicles. In the Scheduling algorithms (I and II) a parameter μ is used to distinguish between the extent to which we need to modify a trip cycle time compared to the base, $\Gamma^p_{max}\ mod\ T^{base}$, and the times a trip cycle time differs from the base, $int\left(\Gamma^p_{max}/T^{base}\right)$. $MaxB$ is the maximum number of elements in set B to branch on or in another word, is the number of times we look for the local optimum by executing the Local Search while we made one of the arcs (edges in both direction) present in the incumbent solution tabu.

To set these parameters, a dataset of 32 instances is generated for this purpose: from each 32 data sets, one instance is randomly chosen. First of all, without any branching iteration ($MaxB = 0$), we tried to set the parameters used in the Local Search including $MaxU$, $MaxP$, μ and ρ. For each of these parameters some different values were tested as follows:

- $MaxU$= 1, 5, 10 and 20,
- $MaxP$ = 1, 5 and 10 (notice that in all cases examined the $MaxU$ was less than or equal to $MaxP$ with respect to the line 2 of ***Algorithm 2*** that keeps the maximum $iterU$ iterations equal to the minimum of $iterP$ and $MaxU$),

- μ = 0.8, 0.9, 0.95, 1, 1.05, 1.1 and 1.2, and
- ρ = 10, 20, and 25.

Based on a trade-off between the quality of the results (in terms of the average solution value) and the required computational effort, following values were selected after the preliminary experiments:

- $MaxU = 10$, $MaxP = 5$, $\mu = 0.95$, $\rho = 20$

This procedure ensured us to have an effective Local Search. Then, for different values of $MaxU$, $MaxP$ and$MaxB$ we performed some more experiments to find the best value for the $MaxB$.

- $MaxU$= 5, 10 and 20,
- $MaxP$ = 5 and 10 , and
- $MaxB$= 1, 5, 10, 20, 50, 100 and 200

Note that some small instances ($n \in [30,70]$) did not need values of 100 and 200 since the total number of the edges in the new incumbent solutions were less than 100. This naturally happens because the quality of the first found solution limits the number of further best found (incumbent) solutions and therefore, the number of edges in set B. Similarly, based on the quality of the results and the required computational effort, we accepted the $MaxU = 10$ and $MaxP = 5$ from our previous experiments and set $MaxB = 50$.

6.3. Results

Our heuristic algorithm was programmed in MS Visual Basic 6. Computational testing was done on a VAIO laptop with 2.30 GHz Intel® Core™ i5-2410M processor, 4.0 GB of RAM and a 64-bit MS Windows operating system. All 320 test instances are solved. Table 2 shows the computational results for all the 320 test instances. Each row represents the average amounts for 10 instances in the set. Columns 2 to 6 show each data set's characteristics or the parameters used (in *Raa (2006)*) to generate 10 instances in that specific row. The rest of the columns present the results obtained by our algorithm: the objective function value, the average number of vehicles, the average number of trips, the average vehicle utilization, the average inventory level at the nodes and the average number of nodes per trip. Based on Table 2, it can be concluded that the average number of nodes per trip (obtained by the algorithm) is rather small, between 1.2 and 6.0, that the average number of trips varies between 8.2 and 86.4 and that the average utilization per vehicle is rather high, between 72 and 89.1 percent. It should be noted here again that it is not the number of nodes per trip that determines the difficulty of the instances, but the fact that these (sometimes high number of) trips should be combined in multi-frequency multi-trip vehicle schedules. Furthermore, changing one node from one trip to another can cause different frequencies of trips and thus completely different schedules. All details about these results are available at: *http://www.mech.kuleuven.be/en/cib/op*.

In Appendix A.2 we present an illustration of a feasible solution for the instance LNHLL-0 from the first data set with 86 nodes. Every TSP route starting from and ending to the depot forms a trip. The vehicle schedules then are determined by different colors. This solution uses 5 vehicles to schedule all the trips (expressed in red, purple, green, cyan and blue).

Table 2. Computational results for 320 instances

Set	Veh Cap	Node Cap	HC	Node Nr	Area	Avg. Obj. value (currency/hr)	Avg. No. of vehicles	Avg. No. of trips	Avg. utilization (%)	Avg. inventory level	Avg. No. of nodes per trip
1	100	No	0.8	[80, 120]	Large	4486.00	4.3	30.8	87.3	1412.8	3.5
2	100	No	0.8	[80, 120]	Small	3120.07	2.9	24.6	88.0	1125.4	4.1
3	100	No	0.8	[30, 70]	Large	2230.16	2.4	13.5	74.6	605.9	3.3
4	100	No	0.8	[30, 70]	Small	1721.12	1.8	12	77.5	533.8	4.2
5	100	No	0.08	[80, 120]	Large	3125.53	3.9	61.5	77.4	2827.3	1.7
6	100	No	0.08	[80, 120]	Small	2057.98	2.6	52.5	76.1	2391.8	1.9
7	100	No	0.08	[30, 70]	Large	1518.90	2	32	68.8	1433.1	1.4
8	100	No	0.08	[30, 70]	Small	1072.18	1.4	29.3	71.4	1295.8	1.7
9	100	Yes	0.8	[80, 120]	Large	5709.70	6.3	23.1	87.5	975.8	4.6
10	100	Yes	0.8	[80, 120]	Small	3635.17	4.1	16.8	81.6	773.3	5.9
11	100	Yes	0.8	[30, 70]	Large	3039.90	3.6	11.1	80.1	406.2	4.1
12	100	Yes	0.8	[30, 70]	Small	2022.00	2.4	8.2	76.4	371.8	6.0
13	100	Yes	0.08	[80, 120]	Large	4972.57	6.2	26.8	88.2	1119.3	4.0
14	100	Yes	0.08	[80, 120]	Small	3030.89	4.1	22	76.8	1012.1	4.5
15	100	Yes	0.08	[30, 70]	Large	2734.35	3.6	12.1	78.8	454.2	3.7
16	100	Yes	0.08	[30, 70]	Small	1721.09	2.4	10.4	72.0	470.6	4.8
17	50	No	0.8	[80, 120]	Large	5079.75	7.3	47.7	87.6	1121.1	2.2
18	50	No	0.8	[80, 120]	Small	3414.26	4.8	37	88.0	874.1	2.7
19	50	No	0.8	[30, 70]	Large	2434.67	3.8	21.1	77.5	490.3	2.2
20	50	No	0.8	[30, 70]	Small	1792.63	2.7	19.2	78.9	444.0	2.6
21	50	No	0.08	[80, 120]	Large	3969.98	6.8	86.4	83.6	1958.0	1.2
22	50	No	0.08	[80, 120]	Small	2473.01	4.2	76	84.7	1721.8	1.3
23	50	No	0.08	[30, 70]	Large	1913.52	3.5	38.1	75.1	845.5	1.2
24	50	No	0.08	[30, 70]	Small	1315.23	2.4	41.7	73.9	914.6	1.2
25	50	Yes	0.8	[80, 120]	Large	5634.33	8.6	37.7	87.8	863.4	2.9
26	50	Yes	0.8	[80, 120]	Small	3629.35	5.4	29.3	89.1	695.0	3.4
27	50	Yes	0.8	[30, 70]	Large	2743.03	4.4	16.8	82.5	373.3	2.7
28	50	Yes	0.8	[30, 70]	Small	1934.28	3.1	14.8	79.6	345.9	3.3
29	50	Yes	0.08	[80, 120]	Large	4917.26	8.4	48	87.9	1091.3	2.2
30	50	Yes	0.08	[80, 120]	Small	3022.72	5.3	43.6	84.7	1001.7	2.3
31	50	Yes	0.08	[30, 70]	Large	2459.69	4.4	20.1	81.4	435.3	2.2
32	50	Yes	0.08	[30, 70]	Small	1659.24	3.1	19.6	76.4	443.2	2.5
		Average				2955.96	4.1	30.7	80.7	963.4	3.0

The results are also compared to the previous studies. Table 3 shows the comparison between our computational results with *Raa (2006)* for all the test instances and Table 4 shows only the comparison with *Raa and Aghezzaf (2009)* for 80 instances. Their computational testing was done on a 2.0 GHz Intel Centrino processor with 1 GB of RAM. Note that in *Raa (2006)*, three algorithms were presented and tested on all the 320 test instances (plus one more algorithm for the single-vehicle version of the problem). One of their algorithms performs better than the two others, which is the one we compared our results with. The mentioned algorithm is a column generation based heuristic solution approach which is also presented in *Raa & Aghezzaf (2009)*, but only for 80 instances with the most important results. These are the instances included in 8 data sets presented in Table 4 and also with odd numbers from 1 to 15 marked by an asterisk in Table 3. To compare the results, we used the following gap percentage formula:

$$Gap\% = 100 * \frac{(Base\ result - To\ be\ compared)}{Base\ result} \qquad (22)$$

Table 3. Results comparison for 320 test instances

	Raa (2006)				Two-phase heuristic				
Set	Obj. value	CPU time (s)	No. of best found	Worst Gap %	Obj. value	CPU time (s)	Nr of best found	Avg Gap %	Worst Gap %
1*	4782.56	657.4	0	-10.38	4486.00	136.55	**10**	6.32	2.73
2	3350.37	674.6	0	-12.06	3120.07	141.45	**10**	6.66	1.34
3*	2297.03	65.2	1	-19.01	2230.16	28.39	9	2.63	-0.12
4	1752.85	97.7	4	-10.07	1721.12	41.20	6	1.23	-0.94
5*	3122.11	8020.5	7	-0.20	3125.53	332.38	3	-0.10	-0.51
6	2017.27	12351	7	-0.77	2057.98	322.97	3	-2.27	-21.76
7*	1514.77	620.5	8	-1.30	1518.90	28.43	2	-0.30	-1.27
8	1068.38	1508.0	7	-0.83	1072.18	72.01	3	-0.32	-2.15
9*	6135.86	336.6	0	-11.67	5709.70	87.65	**10**	6.87	1.41
10	3719.20	340.2	0	-10.33	3635.17	101.46	**10**	2.09	0.51
11*	3112.45	23.7	2	-21.90	3039.90	15.97	8	2.73	-0.99
12	2025.38	40.4	5	-2.64	2022.00	26.28	5	-0.08	-3.16
13*	5393.03	400.1	0	-13.61	4972.57	110.95	**10**	7.67	0.69
14	3102.43	519.9	0	-4.17	3030.89	131.32	**10**	2.24	0.85
15*	2804.01	35.8	2	-12.27	2734.35	18.67	8	2.19	-0.29
16	1737.29	61.6	2	-2.67	1721.09	31.29	8	0.80	-4.55
17	5287.07	883.8	0	-7.69	5079.75	132.54	**10**	4.09	2.31
18	3570.91	816.8	0	-7.81	3414.26	124.16	**10**	4.25	1.59
19	2445.34	76.6	2	-3.58	2434.67	21.68	8	-0.09	-10.24
20	1825.22	127.1	1	-3.55	1792.63	34.91	9	1.59	-0.72
21	3975.74	3814.0	8	-4.44	3969.98	114.73	2	0.02	-1.22
22	2446.54	6041.2	**10**	0.35	2473.01	182.87	0	-1.11	-2.06
23	1889.43	209.8	**10**	0.55	1913.52	12.90	0	-1.23	-2.39
24	1295.79	613.3	**10**	0.30	1315.23	13.97	0	-1.72	-4.29
25	5906.69	502.1	0	-7.29	5634.33	104.67	**10**	4.48	2.00
26	3883.45	462.3	0	-9.71	3629.35	108.34	**10**	6.54	2.46
27	2853.25	36.3	0	-12.10	2743.03	18.06	**10**	3.80	1.05
28	2013.65	62.5	1	-12.26	1934.28	28.96	9	3.31	-0.52
29	5163.67	571.9	0	-8.64	4917.26	137.49	**10**	4.75	0.43
30	3178.95	821.5	0	-9.75	3022.72	148.05	**10**	4.64	1.41
31	2528.04	37.4	2	-12.33	2459.69	18.76	8	2.73	-1.34
32	1678.99	97.1	2	-3.36	1659.24	34.48	8	1.01	-0.53
Average / *Worst/*	3058.68	1278.98		*-21.90*	2955.96	89.49		2.36	*-21.76*
Sum			**91**				**229**		

* The results of these rows are also presented in *Raa & Aghezzaf (2009)*

Since we compare our results with the results of *Raa (2006)* as a base, a positive gap means the base is outperformed. Next to the average gap over 10 instances of each set, we also report the "worst" gap over each of these instances. When this is a positive number, it means we outperform the base algorithm on each instance of that set. *Raa (2006)* was able to find 30 better results (each instance of sets number 22, 23 and 24), while our algorithm found better solutions for 13 complete sets out of 32. Moreover, our algorithm is on average around 14 times faster. For 229 of the 320 instances, our algorithm improved the best known solution. Our algorithm improved the average gap by 2.36%.

Considering the instance results published in *Raa & Aghezzaf (2009)*, we were able to obtain 60 new best known solutions (out of 80 instances) and the average gap was improved by 3.5% (Table 4). These results show the overall performance of our algorithm in tackling this complex problem. To illustrate in more detail the behavior of our algorithm, we present more details in Tables 4 and 5.

Table 4. Results comparison for 80 test instances (Raa & Aghezzaf, 2009)

	Raa & Aghezzaf (2009)				Two-phase heuristic				
Set	Obj. value	CPU time (s)	No. of best found	Worst Gap %	Obj. value	CPU time (s)	Nr of best found	Avg Gap %	Worst Gap %
1*	4782.56	657.4	0	-10.38	4486.00	136.55	**10**	6.32	2.73
3*	2297.03	65.2	1	-19.01	2230.16	28.39	9	2.63	-0.12
5*	3122.11	8020.5	7	-0.20	3125.53	332.38	3	-0.10	-0.51
7*	1514.77	620.5	8	-1.30	1518.90	28.43	2	-0.30	-1.27
9*	6135.86	336.6	0	-11.67	5709.70	87.65	**10**	6.87	1.41
11*	3112.45	23.7	2	-21.90	3039.90	15.97	8	2.73	-0.99
13*	5393.03	400.1	0	-13.61	4972.57	110.95	**10**	7.67	0.69
15*	2804.01	35.8	2	-12.27	2734.35	18.67	8	2.19	-0.29
Average / *Worst/*	3645.23	1269.98		*-21.90*	3477.14	94.87		3.50	*-1.27*
Sum			**20**				**60**		

* The results of these rows are also presented in *Raa & Aghezzaf (2009)*

In Table 5, a summary of the gaps is presented. When comparing with *Raa & Aghezzaf (2009)*, the table shows that we have only one gap in the interval of -5% to -1% compared to their results. Their results, on the other hand, have 24 gaps of worse than -5% compared to our results for those 80 instances. When the best performing algorithm of *Raa (2006)* is compared to our results, they have 68 and 19 (out of 320 test instances) gaps worse than or equal to -5% and -10%, respectively. We just have one gap in the -20% to -10% interval (a gap of -10.24% in the set number 19) and one gap in the -10% to -5% interval. The results show that both our study and *Raa (2006)* have only one gap of worse than -20% compared to each other.

Table 5. Comparison of gap distribution in the results

Study	*Raa & Aghezzaf (2009)*	Two-phase heuristic	*Raa (2006)*	Two-phase heuristic
Number of instances	80		320	
Number of best found solutions	20	60	91	229
Gap Interval (%)				
< -20	1	0	1	1
[-20, -10)	9	0	18	1
[-10, -5)	14	0	49	1
[-5, -1)	20	1	119	28
[-1,0)	16	19	42	60
All	60	20	229	91

Table 6 shows the results obtained when the algorithm is stopped as soon as a first feasible solution is found (columns 2 to 6 of Table 6). This happens when a first iteration of the Routing and Scheduling heuristic are performed. These results show that the first obtained solution of the algorithm (even without the Vehicle Decrease heuristic), found 66 new best known solutions out of 320 test instances in a very short period of time. These results have an average gap of -5% with the solutions of *Raa (2006).* The average running time for all test instances when the first solution is found is about 0.1 seconds. In other words, this approach of decomposing and solving the problem in two phases of Routing and Scheduling obtains acceptable results very quickly. It is a very fast way to obtain initial solutions and this can be useful for other (exact) algorithms as well.

Columns 7 to 11 of Table 6 present the results when the Local Search is executed only once and no branching is performed ($MaxB = 0$). The results show that in an average of 2.77 seconds per instance, the algorithm found 171 best known solutions with a negligible average gap of -0.01%. However, these results show a maximum gap of -45.51% in one of the instances. Then, the results after branching on 10 edges ($MaxB = 10$) are also presented in Table 6 in columns 12 to 16. The results show that in an average of 26 seconds per instance, the algorithm found 208 best known solutions with an average gap of 1.78% (thus performing clearly better than the previous results). However, the maximum gap becomes -21.8% in two of the instances. These results illustrate the importance of some decisions during the design of the algorithm, for example the branching as a diversification strategy and the Vehicle Decrease heuristic in the Local Search.

Table 6. Computational results for the first obtained solution and *MaxB*=0 & 10

Set	First obtained solution					*MaxB*=0					*MaxB*=10				
	Obj. Value	CPU time (s)	Nr of best found	Avg Gap %	Worst Gap %	Obj. Value	CPU time (s)	Nr of best found	Avg Gap %	Worst Gap %	Obj. Value	CPU time (s)	Nr of best found	Avg Gap %	Worst Gap %
1	4843.71	0.21	4	-1.35	-7.50	4589.27	4.86	**10**	3.88	0.58	4530.64	31.97	**10**	5.28	2.60
2	3534.02	0.22	2	-5.49	-11.82	3205.32	3.67	**10**	4.06	0.65	3136.05	32.21	**10**	6.21	1.04
3	2465.61	0.03	0	-7.97	-27.95	2252.14	0.90	6	1.66	-2.52	2232.19	7.27	8	2.53	-0.31
4	2008.44	0.04	0	-16.90	-44.93	1766.24	1.08	1	-1.43	-7.79	1738.26	10.10	4	0.30	-3.31
5	3178.51	0.09	0	-1.82	-2.40	3147.73	8.70	1	-0.82	-1.14	3134.07	81.83	1	-0.39	-0.68
6	2147.07	0.10	0	-6.94	-25.55	2110.49	10.02	0	-5.16	-22.56	2067.76	87.28	1	-2.78	-21.76
7	1677.81	0.01	0	-12.44	-48.07	1606.89	1.56	0	-7.66	-45.28	1519.13	17.57	2	-0.31	-1.27
8	1300.62	0.02	0	-26.86	-58.36	1155.59	2.63	0	-9.50	-45.51	1075.20	25.06	1	-0.65	-2.15
9	5984.95	0.21	7	2.57	-2.94	5805.41	2.25	**10**	5.32	0.61	5730.10	21.09	**10**	6.53	1.33
10	3901.52	0.23	1	-5.34	-17.35	3718.42	3.00	6	-0.17	-9.64	3648.74	25.37	**10**	1.72	0.11
11	3278.75	0.03	1	-4.79	-14.21	3118.16	0.39	4	0.45	-9.65	3056.87	4.60	6	2.18	-1.05
12	2251.28	0.05	0	-11.48	-25.62	2084.42	0.69	2	-3.35	-23.10	2070.42	7.05	3	-2.71	-21.79
13	5306.38	0.19	7	1.58	-9.37	5072.40	3.51	9	5.85	-0.41	4995.24	29.77	**10**	7.23	0.28
14	3158.48	0.20	2	-1.84	-13.19	3102.81	4.12	8	-0.03	-11.56	3037.79	35.90	**10**	2.01	0.14
15	2913.20	0.03	1	-4.67	-22.96	2862.78	0.60	1	-2.94	-21.27	2744.73	5.54	6	1.80	-1.15
16	1885.63	0.05	2	-8.84	-30.13	1751.87	0.79	4	-0.80	-5.13	1724.43	8.90	8	0.60	-4.55
17	5277.87	0.16	6	0.14	-3.57	5175.17	3.13	**10**	2.09	0.66	5108.28	34.53	**10**	3.43	2.05
18	3560.76	0.18	4	0.18	-2.63	3456.04	3.12	**10**	3.08	1.25	3419.01	31.37	**10**	4.11	1.59
19	2501.37	0.02	2	-2.96	-15.39	2452.54	0.77	7	-0.81	-10.70	2437.50	7.40	8	-0.20	-10.26
20	1945.16	0.04	0	-7.02	-16.74	1824.30	0.99	7	-0.28	-10.08	1815.67	9.85	8	0.21	-9.65
21	4028.04	0.06	1	-1.38	-2.53	4016.42	5.73	1	-1.10	-2.15	3974.66	63.06	2	-0.10	-1.43
22	2550.85	0.06	0	-4.27	-12.37	2504.28	6.62	0	-2.38	-10.52	2475.95	71.45	0	-1.23	-2.06
23	1994.56	0.01	0	-6.55	-25.68	1942.22	1.11	0	-2.92	-16.85	1913.52	10.49	0	-1.23	-2.39
24	1374.32	0.01	0	-6.51	-21.92	1321.37	1.90	0	-2.11	-4.29	1315.23	14.57	0	-1.72	-4.29
25	5835.08	0.17	7	1.06	-3.13	5708.98	3.03	**10**	3.19	1.09	5676.81	27.35	**10**	3.76	1.82
26	3823.80	0.18	5	1.40	-3.32	3657.34	3.09	**10**	5.77	0.94	3640.20	27.86	**10**	6.23	2.46
27	2979.32	0.03	1	-5.16	-16.81	2853.80	0.61	8	-0.25	-13.84	2815.72	5.50	9	0.87	-13.84
28	2108.41	0.04	1	-6.77	-28.18	1974.51	0.74	8	1.53	-0.52	1938.90	7.81	9	3.11	-0.52
29	5128.32	0.14	7	0.63	-0.94	5046.60	3.21	9	2.13	-0.75	4954.09	34.32	**10**	4.08	0.43
30	3186.96	0.17	3	-0.48	-8.67	3045.99	4.16	**10**	3.89	0.59	3030.28	37.91	**10**	4.38	0.59
31	2611.80	0.03	1	-3.50	-14.90	2532.59	0.72	6	-0.47	-12.11	2500.32	7.05	6	0.84	-2.29
32	1779.47	0.05	1	-7.89	-33.61	1696.60	1.06	3	-1.11	-10.11	1663.17	10.40	6	0.75	-0.80
Avg /	3141.31	0.10		-5.05		3017.46	2.77		-0.01		2972.53	26.01		1.78	
Worst					*-58.36*					*-45.51*					*-21.79*
/Sum			**66**					**171**					**208**		

7. Conclusions and further research

In this paper, the cyclic inventory routing problem (CIRP) is discussed. It is a challenging complex combinatorial optimization problem. According to the new model presented in Section 4, the objective function is nonlinear and non-convex and on top of that we have sets of nonlinear constraints (constraints 7, 8, 9 and 10). Therefore one can understand that the feasible solution space and the convex hull are rather complicated. The idea of the linear relaxation of the number of vehicles helped us to evaluate the potential node transfers between trips and vehicles rapidly without the need for determining the actual number of necessary vehicles. This is the key to our fast and effective solution approach.

This paper provided additional insights in the structure of the CIRP by proposing a two-phase deterministic algorithm decomposing the problem into two subproblems; one with the nature of routing and the other of scheduling. For each subproblem, we developed a new heuristic algorithm. Our heuristic for the routing phase is based on the steepest descent concept. The second heuristic solves the scheduling subproblem based on building vehicle schedules day by day. Then our algorithm improves the trip cost estimation for the next iteration of the routing algorithm. Trip utility out of a vehicle schedule and trip cycle times play a central role in our approach as they create a connection between the routing and scheduling phases. To escape the local optimum, we used a simple branching scheme. The results compared to what is presented in the literature show that our decomposition approach is very effective not only in obtaining high quality solutions but also in being very fast.

Experimental results were obtained based on 320 test instances (available from the PhD thesis of *Raa (2006)*) including 30 to 120 nodes. The results show that our algorithm outperforms the former heuristic approaches on the benchmark test sets and provides high quality solutions. Applying our algorithm on all test instances, for more than 70% of them new best known solutions are found and the average solution values are improved by 2.36%. Also, among all 80 solutions presented in a journal publication by *Raa & Aghezzaf (2009)*, our algorithm obtained 60 new best found solutions and improved the average solution values by 3.5% .

Another interesting contribution of our paper is that the first iteration of our algorithm can already generate very good initial solutions for this complex problem in less than half a second for problems with 120 nodes. Only the first iteration of Routing and Scheduling produces 66 best known solutions in an average running time of 0.1 seconds. Finding promising initial solutions in such a very short computation time can make our algorithm interesting for researchers working on CIRP to obtain high quality initial solutions for their heuristic or exact algorithms. Branch and cut or branch and price algorithms are the best known tools to solve complex inventory routing problems to optimality. Our heuristic can significantly help these algorithms since starting with a very good initial solution can result in reducing the running times by exploring fewer branches in the search tree and therefore it will be possible to tackle larger problems. Moreover, a further research direction could be the use of a complete branch and bound algorithm structure using upper bounds of our heuristic. This method of using heuristics in an exact algorithm structure is discussed, for instance, by *Fischetti*

& Lodi (2003). We also believe that our idea of the linear relaxation of the number of vehicles with a fixed cost in the objective function and then decomposing the problem into two main subproblems of Routing and Scheduling can help other researchers to develop even better algorithms for this complex problem or related problems with multi-frequency and multi-trip routing.

Facing real life problems, our algorithm can help distribution operators to find promising results in a reasonable time frame. Note that it can work even faster if, in the routing phase, instead of best insertion, 2-OPT and Relocate heuristics, the approximation method offered in *Burns et al. (1985)* will be used to estimate the necessary time of the TSP route of each trip. The fact that our solution approach does not require a commercial solver like CPLEX or Gurobi could also be of interest to software development companies.

8. Acknowledgements

The authors would like to thank professors Raa and Aghezzaf from Gent University for providing the results of their studies. Part of this research was done when the first author was a visiting researcher at CIRRELT and GERAD, Montréal. The authors would like to thank professors Cordeau and Jans from HEC Montreal for all their support during this period of time and three anonymous reviewers for their constructive comments on the previous versions of this paper.

Appendix A.1

Table A. Summary of the notations

Sets	
N, N^+	Set of all nodes and $N \cup \{0\}$, respectively
V	Set of all vehicles
P	Set of all trips p
N_p, N_p^+	Set of all nodes in trip p and $N_p \cup \{0\}$, respectively
P_v	Set of all trips assigned to vehicle v schedule
P'	Set of all non-scheduled trips
p_0	The current trip of a certain node
S, S^*	Solution and best solution identified
B	Set of edges non incident to the depot in best solution identified
Decision variables	
y^v	Equals to 1, if vehicle $v \in V$ is used, and 0, otherwise
T^v	Schedule cycle time of vehicle v (days per vehicle cycle) , $T^v \in \mathbb{N}$
x_{ij}^v	Equals to 1, if vehicle v ∈ V traverses the link between i and j directly, and 0, otherwise
z_{ij}^p	Sum of demand rates (units per day) of the remaining nodes in a trip $p \in P_v$ served by vehicle $v \in V$ when it travels to node $j \in N_p^+$ immediately after it visited node $i \in N_p^+$
Γ^p	Cycle time for the trip p, $\Gamma^p \in \mathbb{N}$ (days per one trip execution)
f^p	Frequency of the trip $p \in P_v$, in the schedule of vehicle v, $f^p \in \mathbb{N}$ (times per vehicle cycle)
Parameters	
ψ	Fixed cost of using a vehicle (currency per day)
δ	Vehicle operating cost (currency per distance)
σ	Average vehicle speed (distance per hour)
φ_i	Handling cost at the node i and the depot for $i = 0$ (currency/node visit and currency/dispatch, respectively)
θ_i	Handling time at node i and the depot (hours/node visit and hours /dispatch, respectively)
η_i	Inventory cost at the node i (currency per unit per day)
μ	Weight adjustment parameter
ρ	A percent of the nodes
d_i	Demand of node i (unit/day)
k_i	local inventory capacity of node i (unit)
κ	Vehicle capacity (unit)
H	Maximum driving time per day (hours per day)
Γ_{min}^p	Minimum cycle time for trip p (days per one trip execution)
Γ_{tsp}^p	TSP route travel time for trip p (hours)
Γ_{max}^p	Maximum cycle time for trip p (days per one trip execution)
$\Gamma_{max}^{Single\ i}$	Maximum cycle time for serving node i in a single-node trip (days per one trip execution)
$\Gamma^p, \hat{\Gamma}^p$	Trip p cycle time and its estimation (days per one trip execution)
$u^p, \hat{u}^p$	Vehicle utilization of trip p and its estimation
$E^p, \hat{E}^p$	Trip p cost and its estimation (currency per day)
T^{base}	Base cycle time for a certain day of a vehicle schedule (days per one trip execution)
$iterB, MaxB$	Iteration counter and the maximum number of different edges or elements of the set B
$iterU, MaxU$	Iteration counter and the maximum number of Routing and Scheduling algorithms runs
$iterP, MaxP$	Iteration counter and the maximum number of Vehicle Decrease heuristic runs

Appendix A.2

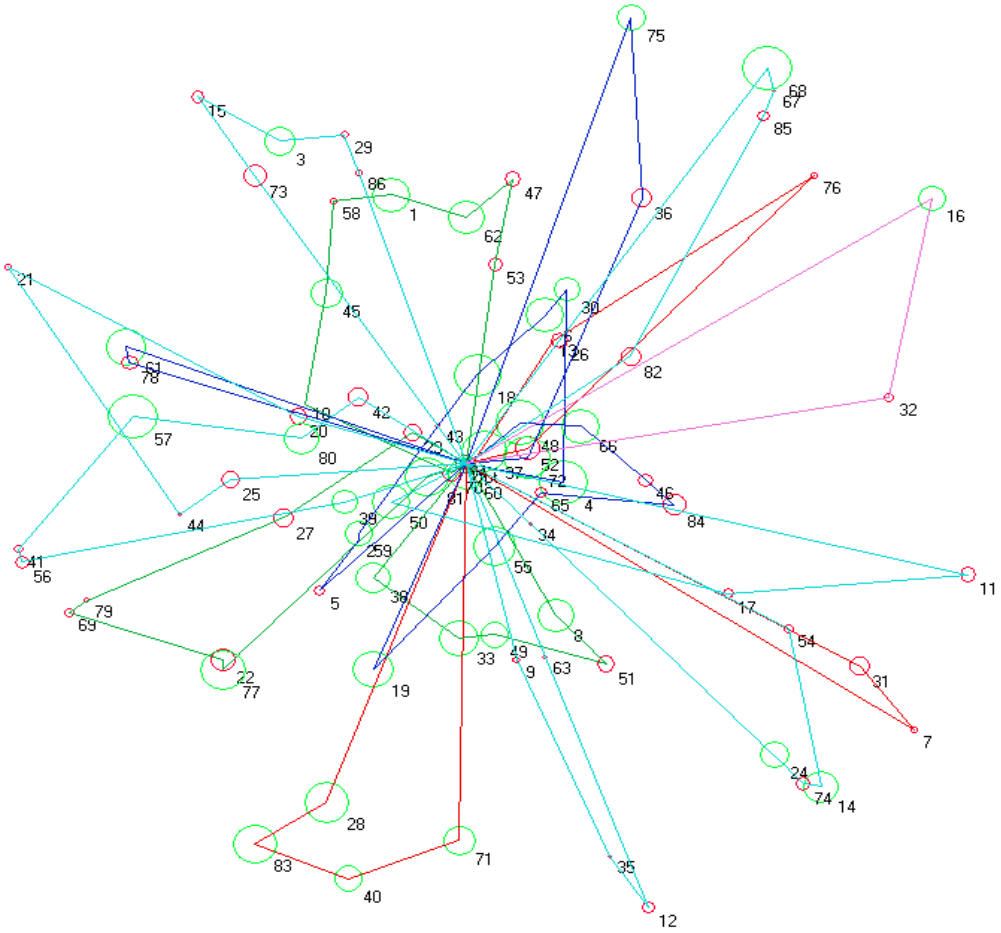

Figure A.2. An illustration of a feasible solution for the instance LNHLL-0 of the data set with 86 nodes

* The circle radius around the node determines its demand rate magnitude where the green circles indicate demands greater than the nodes' average rate and red circles show smaller ones.

9. References

Absi, N., Archetti, C., Dauzère-Pérès, S., & Feillet, D. (2014). A two-phase iterative heuristic approach for the Production Routing Problem. Transportation Science, Articles in Advance, pp. 1–12. doi: 10.1287/trsc.2014.0523.

Adelman, D. (2004). A price-directed approach to stochastic inventory/routing. Operations Research, 52(4), 499-514. doi: 10.1287/opre.1040.0114

Aghezzaf, E.-H., Raa, B., & Van Landeghem, H. (2006). Modeling inventory routing problems in supply chains of high consumption products. European Journal of Operational Research, 169(3), 1048–1063. doi: 10.1016/j.ejor.2005.02.008

Aghezzaf, E.-H., Zhong, Y., Raa, B., & Mateo, M. (2012). Analysis of the single-vehicle cyclic inventory routing problem. International Journal of Systems Science, 43(11), 2040–2049. doi: 10.1080/00207721.2011.564321

Andersson, H., Hoff, A., Christiansen, M., Hasle, G., & Løkketangen, A. (2010). Industrial aspects and literature survey: Combined inventory management and routing. Computers & Operations Research, 37(9), 1515–1536. doi: 10.1016/j.cor.2009.11.009

Anily, S., & Bramel, J. (2004). An asymptotic 98.5%-effective lower bound on fixed partition policies for the inventory-routing problem. Discrete Applied Mathematics, 145(1), 22-39.

Anily, S., & Federgruen, A. (1990). One warehouse multiple retailer systems with vehicle routing costs. Management Science, 36(1), 92-114.

Archetti, C., Bertazzi, L., Hertz, A., & Speranza, M. G. (2012). A hybrid heuristic for an inventory routing problem. INFORMS Journal on Computing, 24(1), 101-116.

Archetti, C., Bertazzi, L., Laporte, G., & Speranza, M. G. (2007). A branch-and-cut algorithm for a vendor-managed inventory-routing problem. Transportation Science, 41(3), 382-391.

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., & Prutzman, P. J. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. Interfaces, 13(6), 4-23.

Bertazzi, L., Paletta, G., & Speranza, M. G. (2002). Deterministic order-up-to level policies in an inventory routing problem. Transportation Science, 36(1), 119-132.

Bertazzi, L., Savelsbergh, M., & Speranza, M. G. (2008). Inventory routing. In: Golden B, Raghavan S, Wasil E, editors. The vehicle routing problem: Latest advances and new challenges (pp. 49–72). Springer US.49–72. ISBN 978-0-387-77778-8

Blumenfeld, D. E., Burns, L. D., Diltz, J. D., & Daganzo, C. F. (1985). Analyzing trade-offs between transportation, inventory and production costs on freight networks. Transportation Research Part B: Methodological, 19(5), 361–380. doi: 10.1016/0191-2615(85)90051-7

Burns, L. D., Hall, R. W., Blumenfeld, D. E., & Daganzo, C. F. (1985). Distribution strategies that minimize transportation and inventory costs. Operations Research, 33(3), 469–490. doi: 10.1287/opre.33.3.469

Campbell, A. M., & Savelsbergh, M. W. P. (2004). A decomposition approach for the inventory-routing problem. Transportation Science, 38(4), 488–502. doi:10.1287/trsc.1030.0054

Christiansen, M., Fagerholt, K., Nygreen, B., & Ronen, D. (2007). Maritime transportation. Handbooks in operations research and management science, 14, 189-284.

Clarke, G. U., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4), 568-581.

Coelho, L. C., Cordeau, J. F., & Laporte, G. (2012). Consistency in multi-vehicle inventory-routing. Transportation Research Part C: Emerging Technologies, 24, 270-287.

Coelho, L. C., Cordeau, J. F., & Laporte, G. (2014). Thirty years of inventory-routing. Transportation Science, 48(1), 1–19. doi:10.1287/trsc.2013.0472

Coffman Jr., E. G., Garey, M. R., & Johnson, D. S. (1996). Approximation algorithms for bin packing: a survey. Approximation algorithms for NP-hard problems (pp. 46–93). PWS Publishing Co. ISBN 0-534-94968-1

Dror, M., Ball, M., & Golden, B. (1985). A computational comparison of algorithms for the inventory routing problem. Annals of Operations Research, 4(1), 1-23.

Ekici, A., Özener, O. Ö., & Kuyzu, G. (2015). Cyclic Delivery Schedules for an Inventory Routing Problem. Transportation Science, 49(4), 817-829.

Federgruen, A. W., & Zipkin, P. (1984). A combined vehicle routing and inventory allocation problem. Operations Research, 32(5), 1019–1037. doi: 10.1287/opre.32.5.1019

Fischetti, M., & Lodi, A. (2003). Local branching. Mathematical programming, 98(1-3), 23-47.doi: 10.1007/s10107-003-0395-5

Fleischmann, B. (1990). The vehicle routing problem with multiple use of vehicles. Working paper, Fachbereich Wirtschaftswissenschaften, Universitat Hamburg.

Francis, P., Smilowitz, K., & Tzur, M. (2006). The period vehicle routing problem with service choice. Transportation Science, 40(4), 439-454.

Gallego, G., & Simchi-Levi, D. (1990). On the effectiveness of direct shipping strategy for the one-warehouse multi-retailer R-systems. Management Science,36(2), 240-243.

Gendreau, M., & Potvin, J. Y. (2010). Handbook of metaheuristics (Vol. 2). New York: Springer. ISBN 978-1-4419-1663-1

Haughton, M. (2013). Tackling complexities of cyclic inventory routing under conditions of limited modelling and computing capacity. International Journal of Logistics Research and Applications, 1–16. doi: 10.1080/13675567.2013.837158

Kleywegt, A. J., Nori, V. S., & Savelsbergh, M. W. (2002). The stochastic inventory routing problem with direct deliveries. Transportation Science, 36(1), 94-118. doi: 10.1287/trsc.36.1.94.574

Lahyani, R., Khemakhem, M., & Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. European Journal of Operational Research, 241(1), 1-14.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 59(2), 231-247.

Larson, R. C. (1988). Transporting sludge to the 106-mile site: an inventory/routing model for fleet sizing and logistics system design. Transportation Science, 22(3), 186-198.

Levner, E., Kats, V., de Pablo, D. A. L., & Cheng, T. E. (2010). Complexity of cyclic scheduling problems: A state-of-the-art survey. Computers & Industrial Engineering, 59(2), 352-361.

Li, J., Chen, H., & Chu, F. (2010). Performance evaluation of distribution strategies for the inventory routing problem. European Journal of Operational Research, 202(2), 412-419.

Moin, N. H., & Salhi, S. (2007). Inventory routing problems: a logistical overview. Journal of the Operational Research Society, 58(9), 1185-1194.

Papageorgiou, D. J., Nemhauser, G. L., Sokol, J., Cheon, M. S., & Keha, A. B. (2014). MIRPLib–A library of maritime inventory routing problem instances: Survey, core model, and benchmark results. European Journal of Operational Research, 235(2), 350-366.

Raa, B. (2006). Models and algorithms for the cyclic inventory routing problem. PhD thesis: Gent University. ISBN: 10 90-8578-120-5.

Raa, B. (2015). Fleet optimization for cyclic inventory routing problems. International Journal of Production Economics, 160, 172-181.

Raa, B., & Aghezzaf, E.-H. (2009). A practical solution approach for the cyclic inventory routing problem. European Journal of Operational Research, 192(2), 429–441. doi: 10.1016/j.ejor.2007.09.032

Schmid, V., Doerner, K. F., & Laporte, G. (2013). Rich routing problems arising in supply chain management. European Journal of Operational Research, 224(3), 435–448. doi: 10.1016/j.ejor.2012.08.014

Solyali, O., & Süral, H. (2011). A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem. Transportation Science, 45(3), 335-345.

Vansteenwegen, P., & Mateo, M. (2014). An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. European Journal of Operational Research, 237, 802-813. doi: 10.1016/j.ejor.2014.02.020

Viswanathan, S., & Mathur, K. (1997). Integrating routing and inventory decisions in one-warehouse multiretailer multiproduct distribution systems. Management Science, 43(3), 294-312.

Zenker, M., Emde, S., & Boysen, N. (2015). Cyclic inventory routing in a line-shaped network. European Journal of Operational Research, Article In Press.

Zhao, Q. H., Wang, S. Y., & Lai, K. K. (2007). A partition approach to the inventory/routing problem. European Journal of Operational Research, 177(2), 786-802.

Zhong, Y., & Aghezzaf, E.-H. (2011). Combining DC-programming and steepest-descent to solve the single-vehicle inventory routing problem. Computers & Industrial Engineering, 61(2), 313–321. doi: 10.1016/j.cie.2011.02.006

Zhong, Y., & Aghezzaf, E.-H. (2012). Effective local search approaches for the single-vehicle cyclic inventory routing Problem. International Journal of Services Operations and Informatics, 7(4), 260–279. doi: 10.1504/IJSOI.2012.052179