8-2019

Simulated annealing for the multi-vehicle cyclic inventory routing problem

Aldy GUNAWAN
Singapore Management University, aldygunawan@smu.edu.sg

Vincent F. YU
National Taiwan University of Science and Technology

Audrey Tedja WIDJAJA
Singapore Management University, audreyw@smu.edu.sg

Pieter VANSTEENWEGEN
Katholieke Universiteit Leuven

Citation

GUNAWAN, Aldy; YU, Vincent F.; WIDJAJA, Audrey Tedja; and VANSTEENWEGEN, Pieter. Simulated annealing for the multi-vehicle cyclic inventory routing problem. (2019). *2019 IEEE 15th International Conference on Automation Science and Engineering: Vancouver, Canada, August 22-26: Proceedings*. 691-696.
Available at: https://ink.library.smu.edu.sg/sis_research/6021

Simulated Annealing for the Multi-Vehicle Cyclic Inventory Routing Problem

Aldy Gunawan[1], Vincent F. Yu[2], Audrey Tedja Widjaja[1] and Pieter Vansteenwegen[3]

Abstract— **This paper studies the Multi-Vehicle Cyclic Inventory Routing Problem (MV-CIRP) as the extension of the Single-Vehicle CIRP (SV-CIRP). The objective is to minimize both distribution and inventory costs at the customers and to maximize the collected rewards simultaneously. The problem is treated as a single objective optimization problem. A subset of customers is selected for each vehicle including the quantity to be delivered to each customer. For each vehicle, a cyclic distribution plan is developed. We construct a mathematical programming model and propose a simulated annealing (SA) metaheuristic for solving both SV-CIRP and MV-CIRP. For SV-CIRP, experimental results on benchmark instances show that SA is comparable to the state-of-the-art algorithms and it is able to improve 12 best known solutions. For the MV-CIRP, the results show that SA performs better than an Iterated Local Search algorithm.**

I. Introduction

In the traditional manufacturing supply chain context, supplier and customers are independently making their decisions on how and when to replenish the inventory. Both suppliers and customers try to minimize their own costs, which are the transportation cost for suppliers and the inventory handling cost for customers. In order to synchronize different activities, Vendor Managed Inventory (VMI) was introduced [1]. In VMI, customers provide their stock levels to the supplier. The supplier takes a full responsibility to manage the inventory of his customers.

A class of routing problems, namely the Inventory Routing Problem (IRP) is discussed as a VMI problem [2]. In IRP, both managing the inventories at the customers and distributing products from a central depot to the customers are considered. The supplier manages the routing of vehicles distributing products and the timely replenishment of inventories at the customers as well. The Cyclic IRP (CIRP), a variant of the IRP, assumes customer demand rates are stable with an infinite planning horizon. The objective is to find a cyclic distribution plan for a set of customers with the objective of minimizing the long term transportation and inventory costs [3].

The Single-Vehicle CIRP (SV-CIRP), a special case of the CIRP, assumes that it is not compulsory to visit every customer. Therefore, the decision of selecting which customers to be served should be carefully made in order to maximize the profit collected minus the transportation and inventory costs. Another property of the SV-CIRP is that the single vehicle is allowed to make multiple trips from the depot within one cycle. The cycle time represents the time between two deliveries to each customer [2], [4], [5].

We extend the SV-CIRP problem by introducing the Multi-Vehicle CIRP (MV-CIRP). In a big retail company, it is common to have a limited number of vehicles available in order to satisfy customers' demands. The complexity of the problem increases since we assume that each customer can only be served by one vehicle, considering that the number of vehicles is limited and split delivery is not allowed. The problem is to decide which customers should be selected for each vehicle. Each vehicle may have a different cycle time due to different customers requiring a visit. However, a crucial decision to be made in this problem is determining which cycle time would minimize the combination of inventory, handling and routing costs across all vehicles. Reference [2] highlighted the complexity of having different cycle times for each vehicle as a challenging path for future research.

The mathematical model of the MV-CIRP is first introduced. Due to the complexity of the model, Simulated Annealing (SA) is designed to solve both SV-CIRP and MV-CIRP. SA is a descent algorithm modified by random ascent moves in order to escape local minima. Computational results show that SA obtains 12 new best known solutions for the SV-CIRP while it also provides high-quality results for the MV-CIRP that can be used as a baseline for future research.

II. Literature Review

In IRP, the decisions of inventory and routing are solved simultaneously, with the objective of minimizing the total cost including holding and transportation costs while avoiding stockouts. Reference [6] introduced a model with an infinite planning horizon for the IRP. This problem is called CIRP [3], in which a route as well as a cyclic distribution scheme for the selected vehicles have to be designed. The vehicles are allowed to perform multiple trips on a single day. Reference [7] proposed a new solution approach for solving CIRP. It builds the solution in two different phases. The first phase establishes routes that cover all customers and the second one focuses on assigning routes to vehicles such that the number of vehicles is minimized.

The Single-Vehicle CIRP (SV-CIRP) is a special case of the CIRP. It is a one-to-many CIRP. An important aspect

[1]Aldy Gunawan and Audrey Tedja Widjaja are with the School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore aldygunawan@smu.edu.sg, audreyw@smu.edu.sg

[2]Vincent F. Yu is with the Department of Industrial Management, National Taiwan University of Science and Technology, 43, Sec. 4, Keelung Road, Taipei 106, Taiwan vincent@mail.ntust.edu.tw

[3]Pieter Vansteenwegen is with the KU Leuven Mobility Research Center, Celestijnenlaan 300 - box 2422, 3001 Leuven, Belgium pieter.vansteenwegen@kuleuven.be

of SV-CIRP is that it is not mandatory to visit all customers. A reward will be imposed for every customer that is visited. Therefore, the objective is thus to simultaneously minimize transportation and inventory costs and maximize the collected rewards from visited customers. Reference [8] proposed a DC-programming approach which is combined with the proposed steepest descent hybrid algorithm in order to solve the benchmark SV-CIRP instances. Reference [9] formulated the SV-CIRP as a mixed-integer programming model with a nonlinear objective function. A steepest descent-like exact algorithm is proposed for solving the problem. In order to deal with larger instances, reference [2] presented an iterated local search (ILS) metaheuristic. Reference [5] reformulated the SV-CIRP as a convex optimization problem and proposed a modified branch and bound procedure to solve the problem.

III. The MV-CIRP

Consider an undirected network graph $G = (S^+, A)$. $S^+ = \{0, 1, 2, \ldots, |S|\}$ is the set of nodes and $A = \{(i, j) : i \neq j \in S^+\}$ refers to the set of arcs connecting two different nodes i and j. Let $S = S^+ \setminus \{0\}$ be a set of potential customers. Node 0 represents the depot. Let $V = \{1, 2, \ldots, |V|\}$ be the set of vehicles that will be used.

Each customer j has an inventory cost η_j (price/time.quantity), a handling cost ϕ_j (price), a demand rate d_j (quantity/time) and a fixed reward λ_j (price/time). The reward corresponds to the collected profit when the customer is selected for replenishment. The non-negative travel time from customer i to customer j is represented by t_{ij}. Each vehicle k is assumed to have a fixed operating cost ψ (price/time), a fixed average vehicle speed v (distance/time), the travel cost δ (price/distance) and the vehicle capacity κ (quantity) [5]. It is assumed that each customer has an infinite inventory capacity. The largest possible quantity that can be delivered to customer j is denoted as ${Q_j}^{max}$, which is calculated as follows:

$$ {Q_j}^{max} = d_j \times \frac{\kappa}{\min_{j \in S}\{d_j\}} \tag{1} $$

The decision variables of the mathematical model are listed below:

- ${x_{ij}}^k$ is a binary variable with the value of 1 if customer j is replenished immediately after customer i by vehicle k
- ${Q_{ij}}^k$ represents the quantity of product remaining in the vehicle k when it travels to customer j immediately after it has replenished customer i. This quantity equals zero if the link (i, j) is not on vehicle k's trip
- ${q_j}^k$ is the quantity of the product delivered to customer j using vehicle k in each cycle
- T^k is the cycle time of the trip made by vehicle k

The MV-CIRP mathematical model is presented below. This model is the extension of the one of SV-CIRP [5].

$$ \begin{aligned} Min \;\; &|V|\psi - \sum_{k \in V} \sum_{i \in S^+} \sum_{j \in S^+, j \neq i} \lambda_j x_{ij}^k \\ &+ \sum_{k \in V} \sum_{i \in S^+} \sum_{j \in S^+, j \neq i} ((\delta v t_{ij} + \phi_j)\frac{1}{T^k} + \frac{1}{2}\eta_j d_j T^k) x_{ij}^k \end{aligned} \tag{2} $$

$$ \sum_{k \in V} \sum_{i \in S^+} x_{ij}^k \leq 1 \;\; \forall j \in S \tag{3} $$

$$ \sum_{i \in S^+, i \neq j} x_{ij}^k - \sum_{l \in S^+, l \neq j} x_{jl}^k = 0 \;\; \forall j \in S^+, \forall k \in V \tag{4} $$

$$ \sum_{i \in S^+} \sum_{j \in S^+, j \neq i} t_{ij} x_{ij}^k - T^k \leq 0 \;\; \forall k \in V \tag{5} $$

$$ Q_{ij}^k \leq \kappa x_{ij}^k \;\; \forall i, j \in S^+, \forall k \in V \tag{6} $$

$$ \sum_{i \in S^+} Q_{ij}^k - \sum_{l \in S^+, l \neq j} Q_{jl}^k = q_j^k \;\; \forall j \in S, \forall k \in V \tag{7} $$

$$ 0 \leq d_j T^k - q_j^k \leq Q_j^{max}(1 - \sum_{i \in S^+} x_{ij}^k) \;\; \forall j \in S, \forall k \in V \tag{8} $$

$$ x_{ij}^k \in \{0, 1\} \;\; \forall i, j \in S^+, \forall k \in V \tag{9} $$

$$ Q_{ij}^k \geq 0 \;\; \forall i, j \in S^+, \forall k \in V \tag{10} $$

$$ q_j^k \geq 0 \;\; \forall j \in S^+, \forall k \in V \tag{11} $$

$$ T^k \geq 0 \;\; \forall k \in V \tag{12} $$

The objective function (2) minimizes the difference between the total cost and the total collected reward. The number of vehicles $|V|$ is fixed beforehand. The total cost consists of the transportation, delivery and holding costs. Constraints (3) ensures that each customer is visited at most once and only by one vehicle. Constraints (4) ensure that for each vehicle that visits or enters a particular node (customer), it also leaves that node. For each vehicle, the total traveling time does not exceed the cycle time, as expressed in (5). Constraints (6) ensure that the quantity delivered by each vehicle does not exceed the capacity of the vehicle. Constraints (7) are flow constraints that ensure the difference between the quantities in the vehicle before it visits a customer and after it has visited the customer equals to the quantity delivered to that customer. Constraints (8) guarantee that for a visited customer, the quantity delivered should equal the demand during the cycle time. Q_j^{max} is utilized as the upper bound limit of T^k. Equations (9) - (12) limit the domain of the decision variables.

IV. Proposed Algorithm

The initial solution is built by grouping customers into clusters. Each cluster will be served by a vehicle. Each customer will only be assigned to a particular cluster. Algorithm 1 gives an overview of the initial solution construction. Let S_i be a set of customers allocated to cluster/vehicle i. S' represents a set of customers that has not been assigned. For $iter = 0$, a customer is selected randomly and it is assigned as the center of a cluster (lines 4 - 8). Each unassigned customer is then selected one by one according to the sequence in S' and assigned to the nearest center of a particular cluster with respect to the Euclidean distance between them (lines 9 - 13). Once all customers have been assigned to clusters, the center of the cluster for subsequent $iter$ is redefined by taking the average of coordinate values of all customers in the cluster (lines 5 - 8) which is a standard approach used in K-means Clustering. The allocation of all customers to new cluster centers is repeated $Niter$ times (line 14), e.g. five times.

Algorithm 1: Initial Solution Construction

```
1  Input: $(S, V)$
2  $iter = 0$;
3  while $iter < Niter$ do
4      $S' \leftarrow S$
5      for $i \in V$ do
6          $S_i \leftarrow \emptyset$
7          Define the center of cluster $i$
8      end
9      while $S' \neq \emptyset$ do
10         Select customer $j \in S'$ and assign to the nearest
             cluster center $i$
11         $S_i \leftarrow S_i \cup \{j\}$
12         $S' \leftarrow S' \setminus \{j\}$
13     end
14     $iter \leftarrow iter + 1$
15 end
16 Calculate the ratio values between two different customers
     including depot using (13)
17 for $i \in V$ do
18     Reorder customers in $S_i$ based on the highest ratio
         values
19     Create a solution representation
20     Select a subset of customers in $S_i$
21     Decode the solution representation into a sequence of
         vehicle $i$
22     Calculate $T^i_{min}, T^i_{max}, T^i_{EOQ}$
23     Define $T^i$
24     Construct the final route of vehicle $i$
25     Update the solution representation of vehicle $i$
26 end
27 Calculate the objective function value (2)
```

In order to define the sequence of customers in vehicle i (line 18), the $ratio_{jj'}$ of two different customers j and j' including depot 0 ($j, j' \in S^+$) (line 16) is calculated by (13). The underlying assumption is to calculate the reward collected per each distance. A node with the higher ratio value is more attractive.

$$ratio_{jj'} = \frac{\lambda_{j'}}{t_{jj'} \times v} \quad \forall j, j' \in S^+ \tag{13}$$

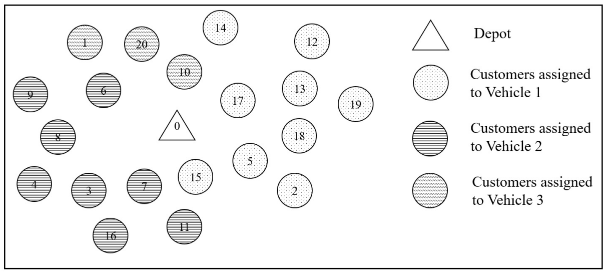
1
20
14
12
10
9
6
17
13
19
0
8
18
5
4
3
7
15
2
16
11
Depot
Customers assigned to Vehicle 1
Customers assigned to Vehicle 2
Customers assigned to Vehicle 3

Fig. 1. An example of the clustering result

Fig. 1 illustrates an example of the clustering result with $|S| = 20$ customers and $|V| = 3$ vehicles. Vehicle 3 considers customers 1, 10 and 20. We start from Depot 0 and find the customer with the highest ratio value, e.g. customer 20. From customer 20, we then compare customers 1 and 10. The final sequence is 20, 10 and 1 (line 18). The sequence of possible customers to be visited for each vehicle is then represented as a string of numbers consisting of a permutation of S_i customers (line 19). In order to separate vehicles, we include -1. Therefore, the number of elements in a particular solution is ($|S| + |V| - 1$), as illustrated in Fig. 2. Using this specially designed solution representation scheme, the routes may be randomly terminated by the values of -1 or by the route capacity constraints [10].

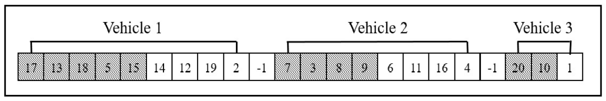

Fig. 2. An example of a solution representation

Since the vehicle may not be able to visit all customers, a subset of customers is taken and treat them as selected customers (line 20), e.g. half of them, for each vehicle, which are shaded in Fig. 2. This solution representation does not represent the final routing sequence of vehicles. It is only utilized for the neighborhood search of SA that will be explained in the next section. From Fig. 2, the solution representation is decoded into a sequence of possible visited customers (line 21), as shown in Fig. 3(a). The cycle time of vehicle i, T^i, is determined by finding the lower bound value T^i_{min} and the upper bound value T^i_{max} [2] (line 22).

T^i_{min} is the total travel time required to visit all selected customers which is based on the solution for the Travelling Salesperson Problem (TSP). The sequence of customers is determined by (13) while T^i_{max} is calculated by (14). It could be possible that T^i_{min} is greater than T^i_{max}. In such a case, we recalculate the values of T^i_{min} and T^i_{max} after removing the customer with the highest demand rate until T^i_{min} is less than T^i_{max}. By removing the customer with the highest demand rate, T^i_{min} will decrease while T^i_{max} will increase simultaneously. The ideal cycle time, denoted as the "economic-order-quantity" T^i_{EOQ} [4], is calculated by (15).

$$T^i_{max} = \frac{\kappa}{max_{j \in S_i} \ d_j} \quad \forall i \in V \tag{14}$$

$$T^i_{EOQ} = \sqrt{\frac{\delta \times v \times T^i_{min} + \sum_{j \in S_j} \phi_j}{\sum_{j \in S_j} \frac{d_j \times \eta_j}{2}}} \quad \forall i \in V \tag{15}$$

There are three possible values of T^i: $T^i = T^i_{EOQ}$ if $T^i_{min} \leq T^i_{EOQ} \leq T^i_{max}$. If $T^i_{EOQ} < T^i_{min}$, then $T^i = T^i_{min}$. If $T^i_{EOQ} > T^i_{max}$, then $T^i = T^i_{max}$ (line 23).

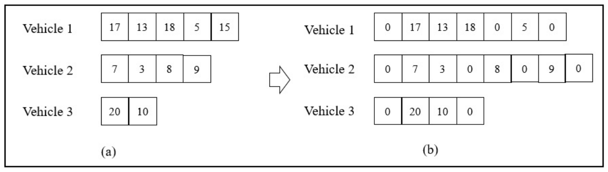

Fig. 3. Decoding results (a) and final routing for each vehicle (b)

Due to the capacity constraint of the vehicle, a vehicle may have multiple trips (e.g. return to the depot again) as long as the total travel time does not exceed the cycle time T^i. As shown in Fig. 3(a), Vehicle 1 visits customer 17 with the number of demand delivered $d_{17} \times T^1$. If the next customer, customer 13, can also be served, Vehicle 1 will continue its journey. Otherwise, it has to return to the depot in order to replenish its capacity and visit customer 13. It is necessary to ensure that the total travel time cannot exceed T^i. For a special case when $T^i_{EOQ} < T^i_{min}$, the problem is solved as a Capacitated Vehicle Routing Problem (CVRP) [2].

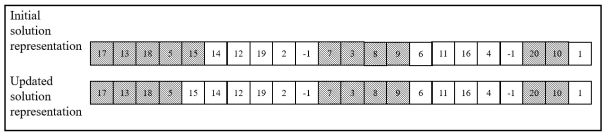

Fig. 4. An example of the updated solution representation

The final routing for each vehicle is generated, as shown in Fig. 3(b) (line 24). Some possible visited customers may not be scheduled anymore, for example, Vehicle 1 does not visit customer 15 although it was planned to be visited in earlier steps. Vehicle 1 needs to return to depot (node 0) after serving customers 17, 13 and 18. The solution representation (Fig. 2) is then updated based on the result shown in Fig. 3(b) (line 25). Fig. 4 illustrates the updated solution representation. Finally, the total objective function value (2) is calculated (line 27).

SA with a random neighborhood structure that features various types of moves, including SWAP, INSERT, INVERSE, ADD and REMOVE, is proposed to improve the initial solution quality (Algorithm 2). Fig. 5 illustrates how all moves are implemented in the solution representation to generate neighborhood solutions. SWAP is performed by randomly selecting two positions and then exchanging the locations of them. INSERT is carried out by randomly selecting one position and inserting it into the position before another randomly selected customer. INVERSE is conducted by randomly selecting two positions and reversing the order of all customers between both. ADD is applied by selecting one unshaded position randomly and converting it into shaded position. REMOVE is the opposite of ADD. All possible moves are randomly selected with the same probability, unless a particular move cannot be performed.

SA use parameters: T_0 refers to the initial temperature. α is a coefficient to control the speed of the cooling schedule ($0 < \alpha < 1$). MAXINNERLOOP refers to the number of iterations at a particular temperature. Let Sol_0, Sol^* and Sol' be the current solution, the best found solution so far and the starting solution at each iteration, respectively. At first, the current temperature $Temp$ equals to T_0 and will be decreased after MAXINNERLOOP iterations by: $Temp = Temp \times \alpha$. FOUNDBESTSOL is initially set as False since a better solution has not been found yet. At a particular value of temperature $Temp$, a possible move is selected randomly in order to explore neighborhoods of Sol_0. Every time one move is selected, we continue with steps explained in lines 21 - 27 of Algorithm 1. For each iteration, the difference between the objective function values of Sol_0 and Sol', denoted as δ, is calculated. If δ is less than 0, which implies that the objective function value is improved, Sol' is replaced by Sol_0. If Sol_0 also improves Sol^*, Sol^* is then replaced by Sol_0 and FOUNDBESTSOL is defined as True. If Sol_0 is worse than Sol', a random number r ($0 < r < 1$) is generated and compared with exp(-$\delta/Temp$). If r is less than exp(-$\delta/Temp$), Sol_0 is accepted and Sol' is updated accordingly; otherwise, we return to Sol' since Sol_0 is not accepted. Sol' is required to keep track of the initial solution at a particular iteration and will be used as the starting point for the next iteration if Sol_0 is rejected. The algorithm will be run until the number of no improvement NOIMPR reaches a threshold LIMIT.

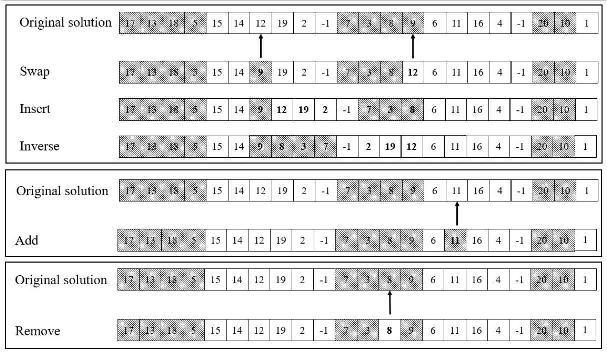

Fig. 5. Examples of neighborhood solutions

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

To the best of our knowledge, benchmark instances for the MV-CIRP are not available; therefore, five sets of SV-CIRP instances [2] are modified by increasing the number of available vehicles $|V|$. Those instances are named according to $|V|$, e.g. 2V-CIRP represents a problem with two vehicles. The capacity for each vehicle is set to a constant value κ.

Algorithm 2: Simulated Annealing

```
1  Input: $(S, V)$
2  $Sol_0 \leftarrow$ Initial Solution Construction
3  $Sol^* \leftarrow Sol_0$
4  $Sol' \leftarrow Sol_0$
5  $Temp \leftarrow T_0$
6  NoImpr $\leftarrow 0$
7  while NoImpr $<$ Limit do
8      InnerLoop $\leftarrow 0$
9      FoundBestSol $\leftarrow$ False
10     while InnerLoop $<$ MaxInnerLoop do
11         $Sol_0 \leftarrow$ select one move randomly
12         $\delta \leftarrow$ obj value of $Sol_0$ - obj value of $Sol'$
13         if $\delta < 0$ then
14             $Sol' \leftarrow Sol_0$
15             if $Sol_0 < Sol^*$ then
16                 $Sol^* \leftarrow Sol_0$
17                 FoundBestSol $\leftarrow$ True
18                 NoImpr $\leftarrow 0$
19             end
20         end
21         else
22             $r \leftarrow rand[0, 1]$
23             if $r < \exp(-\delta/Temp)$ then
24                 $Sol' \leftarrow Sol_0$
25             end
26             else
27                 $Sol_0 \leftarrow Sol'$
28             end
29         end
30         InnerLoop $\leftarrow$ InnerLoop + 1
31     end
32     $Temp \leftarrow Temp \times \alpha$
33     if FoundBestSol = False then
34         NoImpr $\leftarrow$ NoImpr + 1
35     end
36 end
37 Return $Sol^*$
```

SA was coded in C++ and experiments were executed on a PC with Intel Core i7-6700 CPU @ 3.40 GHz processor, 16.0 GB RAM. The quality of the computational results could be influenced by the parameter values [10]. For the initial experiments, we select a subset of instances. A full factorial design with the above-mentioned parameters is implemented. The final parameter values are: T_0 = 10, MaxInnerLoop = 2000, Limit = 100 and α = 0.99, where they will be used for further experiments.

The SV-CIRP instances are first solved. The results are compared against those of the-state-of-the-art algorithms: the combined DC-programming and Steepest Descent Hybrid algorithm (DC-SDHA) [8], the Steepest Descent algorihm (SDA) [9], the SDHA [11], Iterated Local Search (ILS) [4], ILS [2] and the convex optimization [5].

References [8] and [9] provided the optimal solutions (Table I) for Set 1. SA obtains the same solutions within shorter computational times. SA is comparable with ILS [2] and the convex optimization [5] where both obtain the optimal solutions within short computational times as well (less than 10 seconds).

For Sets 2 - 5, only important findings and results are summarized. SV-CIRP and MV-CIRP instances including our solutions are available on the website `https://www.mech.kuleuven.be/en/cib/op/#section-38`. For Set 2, SA outperforms SDHA [11] in terms of CPU time and solution quality. Compared to the results of ILS [2], SA performs better in solving A20-3, A20-6 and A20-9 instances, at the cost of more computational time. In terms of CPU time, SA is faster than the convex optimization [5]. The average gap of SA to the best solutions is 0.21%.

Name	DC-SDHA [8]		SDA [9]		SA	
	Cost	CPU (s)	Cost	CPU (s)	Cost	CPU (s)
A15-0	-328.5	3168	-328.5	116	-328.5	10
A15-1	-295.2	1621	-295.2	308	-295.2	8
A15-2	-283.9	3902	-283.9	173	-283.9	9
A15-3	-386.9	1440	-386.9	812	-386.9	9
A15-4	-360.9	1838	-360.9	363	-360.9	9
A15-5	-348.6	5748	-348.6	724	-348.6	10
A15-6	-399.9	2233	-399.9	271	-399.9	10
A15-7	-347.1	2911	-347.1	291	-347.1	10
A15-8	-393.9	2827	-393.9	574	-393.9	10
A15-9	-316.7	4110	-316.7	166	-316.7	10
Average		2979.8		379.8		9.5

TABLE I
Experimental Results for Set 1 (SV-CIRP)

When comparing against the solution quality of ILS [2] for Set 3, SA performs better in solving A25-1, A25-2, A25-6 and A25-8 instances. The average gap to the best solution is around 0.48% while ILS's gap to the best solution is 0.80%. The convex optimization requires high computational times (e.g. two hours) in order to reach the gap of 0.07% to the best known solutions. The average CPU time of SA is only 76.3 seconds. SA improves one best known solution, A25-8. For Set 4, SA performs better than ILS [2], [4] for all instances at the cost of more computational time. Five new best known solutions are found by SA and the average gap is -1.46%. SA is also 16 times faster with respect to the convex optimization. For Set 5, the solutions of SA are better than those of ILS [2] except for one instance, Ys-VcapL-7 30. SA is only 2 times slower than ILS [2]. SA improves 6 solutions of the convex optimization.

For MV-CIRP instances, SA results are compared against the ones obtained by the commercial software, BARON 18.5.8 as a general nonlinear optimizer. BARON uses the CPU time of 5 hours. The performance of BARON deteriorates as $|V|$ increases, none of the instances can be solved to optimality. Only the best found solutions are reported. BARON cannot even provide the best found solutions for some instances.ILS is also implemented in order to compare with SA using the same moves and CPU times. The average results for different number of vehicles are summarized in Table II. Both SA and ILS perform better than BARON in terms of CPU time and quality of results. SA outperforms ILS in solving all instances with the average gaps range from -0.11% to -25.20%. Based on the statistical tests, the results obtained by SA are significantly better than those obtained by BARON and ILS.

The impact of increasing $|V|$ towards the number of visited

Problem	Set	BARON		ILS		SA		Gap (%)	
		Average Cost	Average CPU (s)	Average Cost	Average CPU (s)	Average Cost	Average CPU (s)	Baron	ILS
2V-CIRP	Set 1	-513.9	18000	-525.4	14	-565.0	14	-10.04	-7.71
	Set 2	-927.9	18000	-897.2	24	-1044.4	24	-13.04	-17.57
	Set 3	-2039.4	18000	-2094.9	86	-2111.1	86	-3.52	-0.77
	Set 4	-1180.4	18000	-1232.7	599	-1522.8	599	-28.93	-25.20
	Set 5	-1454.4	18000	-1750.7	643	-1945.7	643	-34.10	-11.34
3V-CIRP	Set 1	-607.7	18000	-648.3	15	-658.8	15	-8.53	-1.70
	Set 2	-1094.4	18000	-1112.6	38	-1204.2	38	-10.07	-8.93
	Set 3	-1972.2	18000	-2010.9	105	-2031.0	105	-2.99	-1.00
	Set 4	-1248.8*	18000	-1478.1	690	-1794.7	690	-34.23	-22.26
	Set 5	-1855.3*	18000	-2100.2	808	-2311.1	808	-14.90	-9.84
4V-CIRP	Set 1	-598.7	18000	-630.9	18	-634.8	18	-6.16	-0.65
	Set 2	-1186.9	18000	-1170.7	54	-1238.5	54	-4.68	-6.49
	Set 3	-1891.6	18000	-1936.2	118	-1950.7	118	-3.13	-0.75
	Set 4	-1350.1*	18000	-1727.2	910	-1978.4	910	-30.28	-14.86
	Set 5	-1807.7*	18000	-2349.8	849	-2489.3	849	-16.97	-5.52
5V-CIRP	Set 1	-565.0	18000	-590.1	24	-590.7	24	-4.73	-0.11
	Set 2	-1168.2	18000	-1155.5	79	-1199.7	79	-2.84	-4.03
	Set 3	-1815.1	18000	-1852.2	111	-1869.2	111	-2.99	-0.92
	Set 4	-1148.5*	18000	-1896.3	937	-2115.5	937	-47.39	-11.28
	Set 5	-1844.7*	18000	-2445.9	897	-2548.4	897	-7.00	-3.73

* some instances cannot be solved

TABLE II
EXPERIMENTAL RESULTS FOR MV-CIRP

Instance	Objective Function Value					# visited customers					# customers
	SV-CIRP	2V-CIRP	3V-CIRP	4V-CIRP	5V-CIRP	SV-CIRP	2V-CIRP	3V-CIRP	4V-CIRP	5V-CIRP	
A15-0	-328.5	-524.7	-649.5	-628.4	-585.2	8	12	15	15	15	15
A20-4	-480.5	-878.1	-1038.6	-1120.5	-1200.5	7	13	16	18	20	20
Ys-VcapS-7 30	-888.3	-1169.4	-1338.9	-1425	-1381.9	18	24	28	30	30	30
Ys-VcapL-5 67	-1581.4	-2191.3	-2711.7	-3024.6	-3444.6	32	43	52	57	64	67

TABLE III
ANALYSIS OF MV-CIRP RESULTS

customers and the objective function value is analyzed for selected instances (Table III). When $|V|$ is increased, the number of visited customers also increases. As long as not all customers are visited, the objective function value also improves. The collected reward from additional customers is much larger than the costs occurred. However, when all customers are visited, adding vehicles will only increase the costs, and therefore the objective function value will be worse. The current MV-CIRP model enforces $|V|$ vehicles to be used.

VI. CONCLUSIONS

This work studies the Multi-Vehicle Cyclic Inventory Routing Problem (MV-CIRP) as the extension of the Single-Vehicle CIRP (SV-CIRP). The proposed Simulated Annealing algorithm performs comparably to the state-of-the-art algorithms for solving the SV-CIRP. It is able to find 12 new best known solutions. For the MV-CIRP, high-quality solutions obtained can be used as the baseline for future research. Possible future research could be considered: not all vehicles need to be used, demand and delivery lead times uncertainties and applying other metaheuristics or hybridizing other algorithms.

REFERENCES

[1] M. Waller, M. Johnson, and T. Davis, “Vendor-managed inventory in the retail supply chain,” *Journal of Business Logistics*, vol. 20, pp. 183–203, 1999.

[2] P. Vansteenwegen and M. Mateo, “An iterated local search algorithm for the single-vehicle cyclic inventory routing problem,” *European Journal of Operational Research*, vol. 237, pp. 802–813, 2014.

[3] B. Raa and E.-H. Aghezzaf, “A practical solution approach for the cyclic inventory routing problem,” *European Journal of Operational Research*, vol. 192, pp. 429–441, 2009.

[4] Y. Zhong and E.-H. Aghezzaf, “Effective local search approaches for the single-vehicle cyclic inventory routing problem,” *International Journal of Services Operations and Informatics*, vol. 7, pp. 260–279, 2012.

[5] W. Lefever, E.-H. Aghezzaf, and K. Hadj-Hamou, “A convex optimization approach for solving the single-vehicle cyclic inventory routing problem,” *Computers and Operations Research*, vol. 72, pp. 97–106, 2016.

[6] E.-H. Aghezzaf, B. Raa, and H. V. Landeghem, “Modeling inventory routing problems in supply chains of high consumption products,” *European Journal of Operational Research*, vol. 169, pp. 1048–1063, 2006.

[7] B. Raa and W. Dullaert, “Route and fleet design for cyclic inventory routing,” *European Journal of Operational Research*, vol. 256, no. 2, pp. 404–411, 2017.

[8] Y. Zhong and E.-H. Aghezzaf, “Combining dc-programming and steepest-descent to solve the single-vehicle inventory routing problem,” *Computers and Industrial Engineering*, vol. 61, pp. 313–321, 2011.

[9] E.-H. Aghezzaf, Y. Zhong, R. Birger, and M. Mateo, “Analysis of the single-vehicle cyclic inventory routing problem,” *International Journal of Systems Science*, vol. 43, no. 11, pp. 2040–2049, 2012.

[10] V. Yu, S.-W. Lin, W. Lee, and C.-J. Ting, “A simulated annealing heuristic for the capacitated location routing problem,” *Computers and Industrial Engineering*, vol. 58, pp. 288–299, 2010.

[11] Y. Zhong, “Exact and heuristic methods for the cyclic inventory routing problem with side-constraints,” Ph.D. dissertation, Ghent University, Ghent, Belgium, 2012.