European Journal of Operational Research 222 (2012) 596–605

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

# A covering tour approach to the location of satellite distribution centers to supply humanitarian aid

Z. Naji-Azimi [a], J. Renaud [b,c], A. Ruiz [b,c,*], M. Salari [d]

[a] Department of Management, Faculty of Economics and Business Administration, Ferdowsi University of Mashhad, Mashhad, Iran
[b] Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada
[c] Faculté des Sciences de l'administration, Laval University, Canada
[d] Department of Industrial Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

ARTICLE INFO

*Article history:*
Received 23 September 2011
Accepted 1 May 2012
Available online 23 May 2012

*Keywords:*
Vehicle routing
Covering tour
Mathematical model
Heuristics

ABSTRACT

This article concerns the location of *satellite distribution centers* (SDCs) to supply humanitarian aid to the affected people throughout a disaster area. In such situations, it is not possible for the relief teams to visit every single home. Instead, the people are required to go to a satellite distribution center in order to obtain survival goods, provided that these centers are not too far from their homes. The SDCs are usually within walking distance. However, these SDCs need to be supplied from a central depot, using a heterogeneous and capacitated fleet of vehicles. We model this situation as a generalization of the covering tour problem, introduce the idea of split delivery, and propose an efficient heuristic approach to solve it. Numerical experiments on randomly-generated data show that, first, only very small instances can be solved efficiently using the mathematical model and, second, our heuristic produces high-quality solutions and solves real-size instances in a reasonable computing time.



## 1. Introduction

Given the growing number of natural disasters in recent years and the enormous damage that these disasters have caused, the interest of the scientific community in emergency logistics has literally exploded in the last 10 years (Altay and Green, 2006). A humanitarian crisis is a vast, extremely complex situation. Although they are a small part of general crisis management, emergency networks are difficult to design and manage, but their impact on the efficiency of aid delivery is crucial. Emergency logistics is a broad field, which includes such tasks as establishing a rescue command center, collecting information about the disaster area, identifying appropriate sites for shelters, determining the best evacuation routes, delivering relief material, and installing the medical, fire-prevention and emergency construction facilities, as well as setting up evacuation transportation. Thus, these activities may involve both the inflow and outflow of goods and people.

This paper focuses in the inflow logistics and concerns the distribution of survival goods (e.g., food, water, medicine) to the people in the disaster area. Distribution networks for humanitarian aid are often compared to classic industrial distribution networks, replacing suppliers with humanitarian agencies and distribution centers with public sites that are temporarily adapted to store and handle goods (Tzeng and Huang, 2007). The last link in the industrial supply chain – the retailers – is replaced by mobile distribution booths, which are located in any parking lot or any major street intersection so that people can have easy access. Finally, while the objective of general distribution systems is to maximize profit, relief distribution systems try to provide a fair, efficient distribution of aid.

In practice, a logistics network is composed of several *central depots* (CDs), which have been deployed over the affected area, with each CD being responsible of the needs of a given region. In this paper, we will focus on one of these regions, where we assume that a CD has been opened. Thus, the CD location is out of the scope of this paper. However, the interested reader can consult, for example, Rekik et al. (2011) for more details concerning the design of such networks.

In the Canadian province of Quebec, the Civil Protection Act (CPA) was adopted by the government and went into effect on December 20, 2001. According to this CPA, each municipality must develop and update its own emergency preparedness plan, which includes all topics related to emergency logistics. Thus, we assume that the emergency managers have identified several potential distribution sites in the targeted region, where several non-interchangeable products can be made available.

In response operations, the emergency managers must decide which of these potential sites will be used as *satellite distribution centers* (SDCs), depending on the situation. Since the victims have to travel from their homes (i.e., the demand points) to the satellite

* Corresponding author at: Faculté des Sciences de l'administration, Laval University, Canada. Tel.: +1 418 656 7029; fax: +1 418 656 2131.
*E-mail address:* Angel.Ruiz@fsa.ulaval.ca (A. Ruiz).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.05.001

distribution centers, these SDCs must be chosen so that the maximum distance traveled does not exceed a maximum distance set by the emergency managers. In the rest of this paper, we will refer to this maximum distance as the *covering distance*. A demand point (DP) is covered if its distance to an open SDC is shorter than the covering distance.

The SDCs are supplied by a fleet of heterogeneous vehicles located at the central depot. The SDC demand, which corresponds to the demand of the victims assigned to it, may be split and transported in different vehicles. The problem is how to select the locations of the SDCs and how to supply them from the CD using the available vehicle fleet to its best, while covering all the demand points. Fig. 1 shows a network consisting of one central depot, 13 potential satellite distribution centers and 42 demand points (i.e., the victims' homes) to be relieved. The illustrated solution uses two vehicle routes and opens five satellite distribution centers in order to cover all demand points.

The objective of this paper is to provide a tool that supports the emergency managers in designing and operating a satellite distribution center network. To this end, we propose a mathematical model that determines the number and the location of the SDCs, as well as the supply operations plan (i.e., truck route design and the quantities to deliver to each SDC). The rest of this paper is structured as follows. Section 2 defines the problem and proposes a mathematical model. Section 3 reviews and classifies previous studies relevant to the category of covering routing problems. Section 4 presents our heuristic approach. Section 5 reports the results of our extensive numerical experiments to evaluate both the limitations of the mathematical model to solve real-size instances and the quality of the results produced by our heuristic approach. Section 6 offers our conclusions and closes the paper.

## 2. Problem definition and mathematical model

The problem discussed in this paper may be defined as follows. Let $G = (V, A)$ be a complete directed graph in which $V$ represents the vertices and $A$ is the arc set. In the case in question, $V = \{0\} \cup I \cup J$, 0 is the central depot; $I = \{1, \ldots, n\}$ is the set of demand points, where the people affected are located; and $J = \{1, \ldots, m\}$ is the set of potential satellite distribution centers. The number of elements of $J$ that may be visited is free. However, all the demand points of $I$ must be covered. The arc set is defined as $A = \{(v_i, v_j): v_i, v_j \in V\}$, and a distance matrix $c_{ij}$ is defined over $A$. The amount of aid of type $s$ ($s = 1, \ldots, t$) required at demand point $i \in I$ is $d_{is}$ and the weight of each aid unit $s$ is $w_s$. A fleet of $l$ vehicles is available; $Q_k$ represents the capacity (in units) of vehicle $k$=1, …, $l$. Finally, $\alpha = \{\alpha_{ij}\}$ is a $n^*m$ matrix, in which $\alpha_{ij}$ is equal to 1 if demand point $i$ is within the covering distance $\tau$ from SDC $j$, and 0, otherwise. Let us also define the following decision variables:

- $D_{isjk}$ quantity of demand type $s$ at demand point $i$ supplied by vehicle $k$ while visiting SDC $j$;
- $x_{ijk}$ equals 1 if arc $(i, j)$ is used by vehicle $k$, and 0, otherwise;
- $y_{jk}$ equals 1 if SDC $j$ is visited by vehicle $k$, and 0, otherwise; and
- $u_{ik}$ a free variable used in the sub-tour elimination constraints.

The model is formulated as follows:

$$Min \sum_{i=0}^{m}\sum_{j=0}^{m}\sum_{k=1}^{l} c_{ij}x_{ijk} \quad (1)$$

s. t. :

$$\sum_{i=0}^{m} x_{ijk} = y_{jk} \quad j \in \{1,2,\ldots,m\}, k \in \{1,2,\ldots,l\} \quad (2)$$

$$\sum_{i=0}^{m} x_{jik} = y_{jk} \quad j \in \{1,2,\ldots,m\}, k \in \{1,2,\ldots,l\} \quad (3)$$

$$\sum_{j=0}^{m} x_{0jk} = 1 \quad k \in \{1,2,\ldots,l\} \quad (4)$$

$$\sum_{j=0}^{m} x_{j0k} = 1 \quad k \in \{1,2,\ldots,l\} \quad (5)$$

$$\sum_{j=1}^{m}\sum_{k=1}^{l} \alpha_{ij} D_{isjk} \geqslant d_{is} \quad i \in \{1,2,\ldots,n\}, s \in \{1,2,\ldots,t\} \quad (6)$$

$$\sum_{i=1}^{n}\sum_{s=1}^{t} w_s D_{isjk} \leqslant Q_k y_{jk} \quad k \in \{1,2,\ldots,l\}, j \in \{1,2,\ldots,m\} \quad (7)$$

$$\sum_{s=1}^{t}\sum_{i=1}^{n}\sum_{j=1}^{m} w_s D_{isjk} \leqslant Q_k \quad k \in \{1,2,\ldots,l\} \quad (8)$$

$$u_{ik} - u_{jk} + (m+1)x_{ijk} \leqslant m \quad i,j \in \{1,2,\ldots,m\}, k \in \{1,2,\ldots,l\} \quad (9)$$

$$x_{ijk} \in \{0,1\} \quad i,j \in \{0,1,\ldots,m\}, k \in \{1,2,\ldots,l\} \quad (10)$$

$$y_{jk} \in \{0,1\} \quad j \in \{1,2,\ldots,m\}, k \in \{1,2,\ldots,l\} \quad (11)$$

$$u_{ik} \geqslant 0 \quad i \in \{1,2,\ldots,m\},\ k \in \{1,2,\ldots,l\} \quad (12)$$

$$D_{isjk} \geqslant 0 \quad i \in \{1,2,\ldots,n\},\ s \in \{1,2,\ldots,t\}\ j \in \{1,2,\ldots,m\},\ k \in \{1,2,\ldots,l\} \quad (13)$$

The objective (1) is to minimize the total distance traveled by the vehicle fleet. Constraints (2) and (3) insure that, for each SDC $j$ and for each vehicle $k$, there are either both incoming and outgoing arcs or no arcs at all. Constraints (4) and (5) insure that, for each vehicle of type $k$, there are two arcs connected to the depot. Constraint (6) state that demand type $s$ of demand point $i$ may come from various SDCs $j$ and be delivered with one or more vehicles. Constraint (7) link the distribution variables $D_{isjk}$ to the use of vehicle $k$ for a delivery to demand point $j$. For each vehicle $k$, constraint (8) impose the capacity threshold, and constraint (9) are the classic sub-tour elimination constraints (Miller et al., 1960). (Please note that the right-hand side is $m$, and not $m - 1$, because the starting point is 0.) The remaining constraints are the decision variable definitions.

The following sub-tour elimination constraints can be used as well, since they are equivalent to the constraints in Kara et al. (2004), considering that $\sum_{s=1}^{t}\sum_{h=1}^{n} w_s D_{hsjk}$ is the demand of SDC $j$:

$$u_{ik} - u_{jk} + Q_k x_{ijk} \leqslant Q_k - \sum_{s=1}^{t}\sum_{h=1}^{n} w_s D_{hsjk} \quad i,j \in \{1,2,\ldots,m\} \quad k \in \{1,2,\ldots,l\} \quad (14)$$

Clearly, this model may lack efficiency for solving medium to large instances. This is the reason why we developed a heuristic approach that will be presented in Section 4.

## 3. Literature review

As defined, our problem is closely related to a family of *covering problems*, such as the covering salesman problem, the covering tour problem and the median cycle problem, among others. In this section, we review the most important covering problems and outline the differences between them.

In the *Covering Salesman Problem* (CSP), the objective is to identify the minimum cost tour of a subset of $p$ cities so that every city not on the tour is within some predetermined covering distance of a city that is on the tour. Current and Schilling (1989) used the CSP to elaborate the routes for healthcare teams in developing countries, where the team has to visit a subset of the villages and the rest of the population have to be within a walking distance of the visited sites. They provided an integer linear formulation for

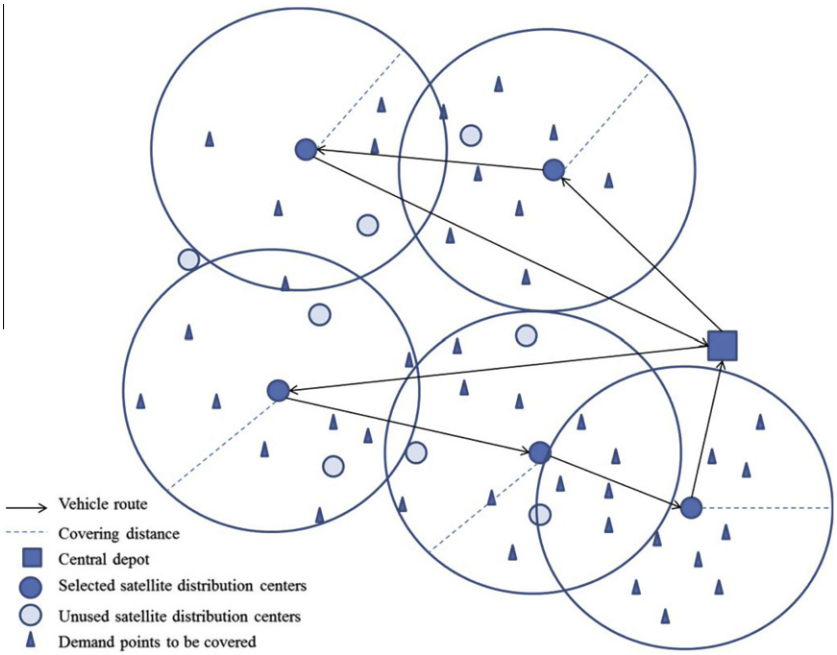


**Fig. 1.** Example of the studied network.

the CSP and a two-steps heuristic based on a set covering problem followed by a traveling salesman problem (TSP).

Current and Schilling (1994) introduced two bi-criteria variants of the CSP, called the *Median Tour Problem* (MTP) and the *Maximal Covering Tour Problem* (MCTP). In both problems, the tour must visit only $p$ of the villages, and the length of this tour must be minimized. In the MTP, the second objective is to minimize the total distance between each unvisited village and the nearest visited village. For the MCTP, the second objective is to maximize the total demand within some prespecified maximal travel distance from a tour stop. Golden et al. (Forthcomming) defined and developed a generalization of this problem and referred to some of the real applications in which satisfying the demand of some customers cannot be met by visiting or covering them for just once and each city $i$ has to be covered $k_i$ times. A local search heuristic based on classic TSP improvement procedures is proposed.

Gendreau et al. (1997) studied the *Covering Tour Problem* (CTP) in which the vertices are divided as follows: $W_1$ is a set of vertices that can be visited; $\overline{W} \subset W_1$ is a set of vertices that must be visited, including the depot; and $W_2$ is a set of vertices that must be covered. The objective of the CTP is to minimize the length of the Hamiltonian cycle over a subset of $W_1$ in such a way that the tour contains all vertices $\overline{W}$, and every vertex of $W_2$ is covered by the tour (i.e., it lies within a specified distance from a vertex of the tour). To solve this problem, they proposed first an exact branch-and-cut algorithm and second, as Current and Schilling (1989), an algorithm combining both a set covering and a TSP heuristics. Baldacci et al. (2005) proposed three scatter search algorithms to solve the CTP. Hachicha et al. (2000), and previously Hodgson et al. (1998) in the case of a mobile healthcare facilities planning application, have studied the multi-vehicle CTP. Jozefowiez et al. (2007) studied the *Bi-Objective Covering Tour Problem* (BOCTP), in which the second objective consists of minimizing the greatest distances of the covered nodes. They proposed a two-phase cooperative heuristic that combines a multi-objective evolutionary algorithm with a branch-and-cut algorithm. They also developed an $\varepsilon$-constraint approach to determine optimal Pareto sets. An extended and stochastic version of the BOCTP was studied by Tricoire et al. (2012).

Nolz et al. (2010) proposed a *Multi-Objective Covering Tour Problem* (MOCTP) in which, given a central depot and a set of identical vehicles, the demand of each node has to be satisfied by exactly one vehicle. The goal of the problem is to minimize the following objectives: (1) the combination of the *mini-sum* facility location criterion (i.e., the sum of distances between all nodes and their nearest open facility) and the maximal covering location criterion (i.e., the number of nodes unable to reach a facility within a predefined maximum distance), (2) the total tour length, and (3) the latest arrival time at a node. They solve a bi-objective problem by considering the objectives (1) and (2) and then the objectives (1) and (3). To this end, a metaheuristic encompassing variable neighborhood search, path relinking, and a genetic algorithm, is proposed.

The covering tour problem is related to the *Prize-Collecting Traveling Salesman Problem* (PCTSP) and to the *Selective Traveling Salesman Problem* (STSP). In these problems, a non-negative profit $p_i$ is associated with each vertex $i$. In the PCTSP, the objective is to minimize the tour length through a subset of the vertices so that the profit $p_i$ collected on the subtour is at least equal to a given value (Fischetti and Toth, 1988). On the contrary, in the STSP, the objective is to search for a subtour with the highest profit, and a length not exceeding a preset value (Laporte and Martello, 1990). In both cases, authors provided linear integer formulations, and solved them by branch-and-bound algorithms.

Another related problem is the *Median Cycle Problem* (MCP), which is studied in two versions. In the first version called MCP1, the sum of routing distance of the cycle and the assignment distance of the vertices not in the cycle to their nearest vertex in the cycle are minimized. This problem is also called the ring star problem (Kedad-Sidhoum and Hung Nguyen, 2010). In the second version called MCP2, the routing distance is minimized, subject to an upper bound on the assignment distance. Moreno Pérez et al. (2003) and Renaud et al. (2004), solved both versions of the MCP by using, respectively, a variable neighborhood tabu search heuristic, and an evolutionary algorithm.

Table 1 presents the characteristics of the major related problems appearing in the literature. To the best of our knowledge, none of the published papers integrates simultaneously all the characteristics of the situation studied in this paper. Our problem

**Table 1**
Characteristics of major related problems.

| Problem name | Authors | Objective function | No. of vertices in the subtour | Kinds of node | Covering distance | Nodes with demand | Nodes with profit | No. of products | No. of vehicles |
|---|---|---|---|---|---|---|---|---|---|
| Covering Salesman Problem – CSV | Current and Schilling (1989) | Minimize distance | Fixed, $p$ | One | Yes | No | No | 1 | 1 |
| Median Tour Problem – MTP | Current and Schilling (1994) | Minimize $Z_1$ = distance and $Z_2$ = assignment cost | Fixed, $p$ | One | No | Yes | No | 1 | 1 |
| Maximal Covering Tour Problem – MCTP | Current and Schilling (1994) | Minimize $Z_1$ = distance and maximize and $Z_2$ = demand within a covering distance | Fixed, $p$ | One | Yes | Yes | No | 1 | 1 |
| Covering Tour Problem – CTP | Gendreau et al. (1997) and Baldacci et al. (2005) | Minimize distance while covering nodes of $W_2$ | Free | $W_1$ can be visited, some must be visited and $W_2$ must be covered | Yes | No | No | 1 | 1 |
| Bi-Objective Covering Tour Problem – BOCTP | Jozefowiez et al. (2007) | Minimize distance of visited nodes and the maximum distance of covered nodes | Free | $W_1$ can be visited, some must be visited and $W_2$ must be covered | Yes | No | No | 1 | 1 |
| Multi-Objective Covering Tour Problem – MOCTP | Nolz et al. (2010) | Combination of two objectives choosen between three | Free | One | Yes | Yes | No | 1 | $m$ |
| Multi-Vehicle Covering Tour Problem – $m$-CTP | Hachicha et al. (2000) | Minimize distance while covering nodes of $W_2$, subject to maximum number of nodes and maximum distance per route | No more than $p$ | $W_1$ can be visited, some must be visited and $W_2$ must be covered | Yes | No | No | 1 | $m$ |
| Prize Collecting Traveling Salesman Problem – PCTSP | Fischetti and Toth (1988) | Minimize distance subject to a minimum profit collected | Free | One | No | No | $p_i$ | 1 | 1 |
| Selective Traveling Salesman Problem – STSP | Laporte and Martello (1990) | Maximize profit subject to a maximum distance | Free | One | No | No | $p_i$ | 1 | 1 |
| Median Cycle Problem – MCP1 | Moreno Pérez et al. (2003), Kedad-Sidhoum and Hung Nguyen (2010), and Renaud et al. (2004) | Minimize distance and assignment cost | Free | One | No | No | No | 1 | 1 |
| Median Cycle Problem – MCP2 | Moreno Pérez et al. (2003) and Renaud et al. (2004) | Minimize distance subject to a maximum assignment cost | Free | One | No | No | No | 1 | 1 |
| Current contribution | | Minimize distance | Free | $J$ can be visited and $I$ must be covered | Yes | Yes | No | Many | $m$ |

extends the multi-vehicle covering tour problem to include multiple commodities, heterogeneous capacitated fleet, and split deliveries. These characteristics are needed to model humanitarian aid distribution more accurately.

## 4. A multi-start heuristic

This section presents our heuristic approach for solving the defined problem. The procedure starts by running a *Preprocessing* step on the problem data in order to identify whether or not one or several SDCs need to be opened at specific locations to guarantee that a feasible solution can be reached. After running the preprocessing step, an initial feasible solution is produced by the *Initialization* step, and then a *Local Search* (LS) is conducted. This local search searches several neighborhoods and uses several procedures that try to avoid the search being trapped by the local optima. The LS was embedded into a multi-start mechanism so that, for a specific number of times, a solution is constructed and improved iteratively. The goal of this mechanism is to increase the robustness of the LS, with respect to the initial solution, as well as to improve its ability to explore non-visited regions of the solution space. The algorithmic structure of our heuristic approach is shown in Fig. 2, and each procedure is described in the following subsections.

### 4.1. Preprocessing

This first step of the algorithm is executed only once, although of the number of restarts (*#_restarts*) selected may be more. The goal of preprocessing is to identify demand points (DPs) that have a single SDC within the maximum covering distance $\tau$. In other words, it seeks for DPs that can only be covered from one single SDC and selects these SDCs to be in any feasible solution. These SDCs will be denoted "essential" in the rest of the paper. For each SDC $j$, the subset $T_j$, including all the demand points within a distance $\tau$ of $j$, is constructed.

### 4.2. Initialization

The *Initialization* step finds an initial feasible solution. At the beginning of this step, all the demand points $i \in I$ are tagged as

"unassigned", and the remaining capacity of each vehicle $k \in K$, denoted $R_k$, is set to $Q_k$. The *Initialization* step randomly draws a vehicle $k$ and a SDC $j$ from the lists of available vehicles and potential SDCs, respectively. Then, a demand point $i \in T_j$ is randomly selected. If the weight of the total demand of DP $i$ for every type of demand $s$ (i.e., $\sum_s d_{is} w_s$) is less than or equal to $R_k$, then the demand point $i$ is "assigned" to SDC $j$. $R_k$ is decreased appropriately, and another unassigned DP in $T_j$ is considered. If all the DPs in $T_j$ are assigned, then the next SDC in route $k$ (i.e., $j+1$) is selected as the closest SDC to $j$ for which $T_{j+1}$ contains at least one unassigned demand point. The DPs, and therefore the SDCs, are added to the current route $k$ as long as $R_k$ remains positive.

If $R_k$ cannot satisfy the volume required by the incumbent DP, the algorithm assigns as many products as possible (the order in which products are assigned is irrelevant), and the route $k$ is completed by returning the vehicle to the depot. Then, a new vehicle $k+1$ is randomly selected from the vehicle list, the current SDC being its first stop, and the unsatisfied demand of the incumbent DP is assigned to it. As a result, the demand of this DP is split. The algorithm assigns demand points, adds SDC and builds new routes (i.e., selects new vehicles) until all the DPs have been assigned. The feasible solution produced by the initialization step is then used as the initial solution for the local search. (Please note that, in the rest of this paper, we use "routes" and "vehicles" as synonyms because only active vehicles are considered and each route is performed by a single vehicle.)

### 4.3. Local search

The *Local Search* contains several mechanisms procedures that have been designed to tackle specific parts or characteristics of an incumbent solution. In particular, the search process executes consecutively the following procedures:

1. the *Delete-Redundant-SDC* procedure, which tries to eliminate unnecessary SDCs (or stops) on the routes;
2. the *Swap* procedure, which tries to reduce the route length by applying a 2-Opt exchange on each route and between different routes;
3. the *Drop* & *Add* procedure, which tries to replace an opened SDC by one or more SDCs that are not visited in the current solution;
4. the *Extraction–Insertion* procedure, which tries to move a SDC from its current route to another one, so as to better use the vehicle capacity.

The procedure is repeated for a given number of iterations (*#_LocalSearch_iterations*).

The following subsections are devoted to the comprehensive description of each procedure.

#### 4.3.1. Delete-Redundant-SDC procedure

An SDC $j$ is called redundant if, after removing it from a route, the solution remains feasible, meaning that all the demand points currently satisfied by $j$ can be reassigned to another open SDC on the same route. This procedure is particularly useful because the *Initialization* step tends to produce solutions that contain too many SDCs. The *Delete-Redundant-SDC* procedure considers the routes in the current solution consecutively, and, for each route, evaluates whether or not the solution remains feasible after removing each SDC. As the triangle inequality holds true, removing a SDC from a route reduces the route's length; in the worst case, the route's length will remain the same.

Please note that the feasibility after removing a specific SDC $j$ can be tested by verifying if all the DPs served by $j$ are within the covering distance ($\tau$) of the other SDCs visited in the same route. If not, this SDC cannot be removed. Once a SDC $j$ is selected, the algorithm tries to assign as many demand points as possible to the closest SDC to SDC $j$. If any of the DPs covered by SDC $j$ remain uncovered, then the algorithm tries the second closest SDC to $j$, and so on until all the DPs originally assigned to $j$ are assigned to the other SDCs on the same route, in which case $j$ may be removed, or until all the SDCs on the route have been tested, in which case the algorithm concludes that $j$ cannot be removed.

#### 4.3.2. Swap procedure

The *Swap* procedure tests whether or not a different SDC visiting sequence will lead to a shorter route. First, the procedure considers exchanging positions between two SDCs on the same route. Once all the possible exchanges have been examined, it considers exchanges involving SDCs on two different routes. These exchanges are repeated as long as the solution improves.

Swapping SDCs within a single route is rather simple as the resulting route is always feasible and the assignment of demand points to the SDC remains unchanged. Each pair of SDCs on a given route is considered, and the positions of these two SDCs are exchanged. The resulting tour lengths are calculated, and the exchange producing the largest improvement, if any, is implemented. Fig. 3 illustrates a swap between $j_1$ and $j_2$ on route $k_2$, where $j_2$ is visited simultaneously by routes $k_1$ and $k_2$. The procedure is repeated for each route as long as the total routes length is improved.

Swapping SDCs belonging to two different routes is slightly more complicated because the vehicle capacities and the demand delivered at the studied SDCs need to be taken into account. In fact, each SDC serves different sets of demand points, thus entailing different demands for each product type, so the required vehicle capacity is also different. It follows that swapping two SDCs on different routes is not always possible due to vehicle capacity.

To better illustrate this point, let us consider the swap between SDC $j_1$ and SDC $j_2$ visited currently by vehicles $k_1$ and $k_2$, which have remaining capacities $R_1$ and $R_2$, respectively. Let us also consider the capacity required to satisfy demand served by $j_1$ and $j_2$, denoted $d_1$ and $d_2$, respectively ($d_1 = \sum\sum D_{isj_1k_1} w_s$, and $d_2 = \sum\sum D_{isj_2k_2} w_s$ for all $s$ and $i$). A swap between $j_1$ and $j_2$ is only possible if the following swap feasibility condition is satisfied: $min(R_1 + d_1 - d_2; R_2 + d_2 - d_1) \geqslant 0$.

Split delivery causes another difficulty to deal with. If a SDC considered for a swap is visited by $n$ different routes (i.e., a split SDC), then $n$ different swaps are possible. Let us assume that $j_1$, which is visited by vehicle $k_1$, is selected to be swapped with $j_2$, which is visited by vehicles $k_2$ and $k_3$. In this case, two possible exchanges are possible: (1) $j_1$ is visited by $k_2$, with $j_2$ being visited by $k_1$ and $k_3$, and (2) $j_1$ is visited by $k_3$, with $j_2$ being visited by $k_1$ and $k_2$. This situation is shown in Fig. 4, in which (a) illustrates a feasible solution before the swap, and (b) and (c) illustrate the two possible swaps described above.

It follows that, if $j_1$ is visited by $n$ vehicles and $j_2$ is visited by $m$ vehicles, then $n^*m$ different possible *swaps* need to be evaluated. Taking into account all these particularities, the *swaps* between SDCs on the different routes is done as follows:

1. The algorithm starts by choosing arbitrarily the first route (or vehicle) $k_1$, and its first stop on the route, $j_1$.
2. The algorithm builds a list containing the $\phi$ SDCs closest to $j_1$, but not on the same route. Restricting the number of SDC to be swapped limits the computational effort required to evaluate the potential swaps.
3. The algorithm evaluates all the swaps between $j_1$ and each of the SDCs in the list. To this end, the feasibility test is performed. If the test is satisfied, the total length of the two routes concerned, before and after swap, are computed and stored. If the SDC being considered in the swap is visited by more than one route, all the potential swaps need to be evaluated.

```
Preprocessing
For r = 1 to #_restarts
    Initialization
    For i = 1 to #_iterations
        For j=1 to #_LocalSearch_iterations
            Delete-Redundant-SDC
            Swap
            Drop&Add
            Extraction-Insertion
        next j
        Diversification
    next i
next r
end
```

**Fig. 2.** Algorithmic structure of our heuristic.

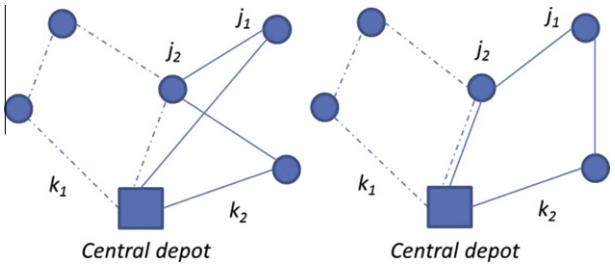


**Fig. 3.** Swapping with a SDC visited by two routes.

The procedure is repeated for all the stops in route $k$, and then applied in the exact same way to the other routes in the solution. The parameter $\phi$, which defines the size of the neighborhood to explore, needs to be set carefully by the decision-maker. In fact, as $\phi$ increases, more potential swaps are evaluated. However, as $\phi$ increases, the SDCs that are at a greater distance are considered, but their chances of reducing the total distance decreases.

#### 4.3.3. Drop & Add procedure

It could happen that, for a current non-optimal route, visiting more and different SDCs leads to a shorter route length. In addition, when considering a SDC visited by two or more routes, replacing this SDC by one or more neighboring SDCs that are not in the current solution may also improve the solution's total route length. These two situations are illustrated in Fig. 5. In Fig. 5a, SDC $j_1$ is replaced by $j_2$ and $j_3$, thus reducing the total length of the route in Fig. 5b. In Fig. 5c, $j_1$ is visited by $k_1$ and $k_2$. After the *Drop* & *Add* procedure, in Fig. 5d, $j_1$ is replaced by $j_2$ and $j_3$, thus reducing the total distance of $k_1 + k_2$.

The goal of the *Drop* & *Add* procedure is to remove a SDC and replace it with a subset of unvisited SDCs that decrease the route length. To this end, a visited SDC is selected and removed from the current solution. Thus, some DPs will eventually become uncovered, so the algorithm tries to assign them to other closest SDCs visited on the same route. Still, if demand points remain uncovered, the algorithm tries to cover them by adding one or more unvisited SDCs to the solution. The algorithm considers unvisited SDCs to add according to their increasing distance to SDC $j$, which has been removed. If the closest SDC $j'$ is able to cover at least one uncovered DP, $j'$ is added to the route so that the new route's length is minimized; then, as many uncovered DPs as possible are assigned to $j'$. The algorithm adds unvisited SDCs until the solution becomes feasible again.

This procedure is performed for every SDC in the current solution. The replacement leading to the shortest total length is implemented, even if the total length of the current route is increased with the less damaging exchange.

#### 4.3.4. Extraction–Insertion procedure

The *Extraction–Insertion* procedure moves one SDC from one route to another. To this end, the algorithm selects a SDC $j$, currently visited by vehicle $k$, and tries to evaluate if $j$ could be served by another vehicle $k'$. This is done by checking if the remaining capacity of each of the other vehicles $R_{k'}$ is equal or greater than the volume required to transport the total demand covered by SDC $j$, denoted $d_j (d_j = \sum_i \sum_s D_{isjk} w_s)$. If transferring $j$ from route $k$ to route $k'$ is possible, then the algorithm inserts $j$ into $k'$ so that $k'$ route length increase is minimized. For each SDC, the algorithm evaluates every possible transfer and implements the one resulting in the lowest route length, if it improves the current solution. The procedure is repeated as long as improvements are found.

### 4.4. Diversification procedure

The goal of the *Diversification* procedure is to perform a soft diversification of the incumbent solution by adding $r$ unvisited SDCs to the solution. The rationale behind this procedure is to give the solution some flexibility that will, hopefully, allow the local search to perform changes in the solution structure at the following iteration. To this end, each time the local search loop is finished, the *Diversification* procedure selects randomly $\varphi$ unvisited SDCs and inserts each of them into the routes of the current solution so that the insertion cost (i.e., the increasing route length of the solution) is minimized.

This is done in the following manner. After selecting an unvisited SDC $j_1$ with uniform probability, the procedure identifies $j_2$,

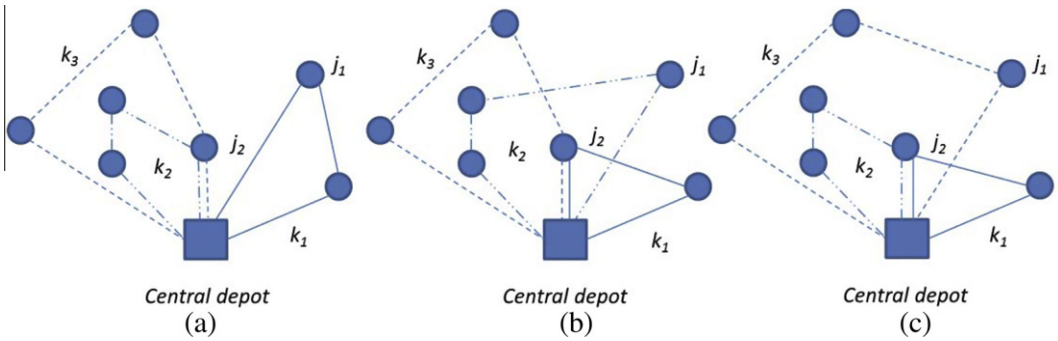


**Fig. 4.** Possible swaps when one of the selected SDCs is visited by two routes.

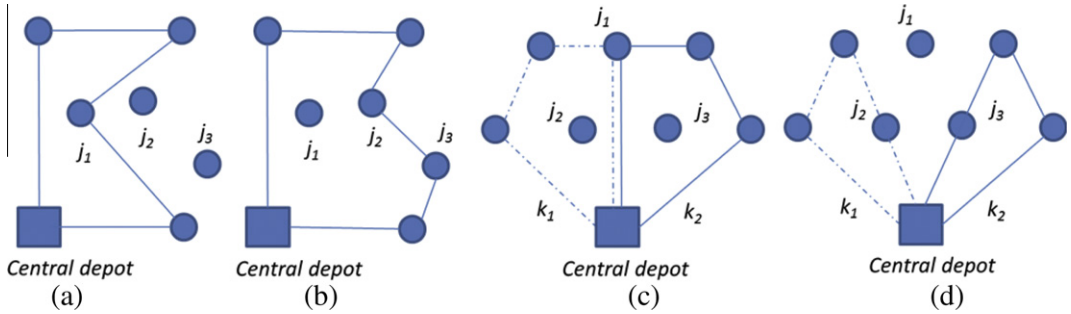


**Fig. 5.** Examples of potential improvements by *Drop* & *Add*.

the closest SDC to SDC $j_1$ already used in the current solution. Then, the algorithm evaluates the insertion of $j_1$ before and after $j_2$, and the solution that minimizes the total route length of the solution is retained. If $j_2$ is visited by several vehicles, then $j_1$ will be visited by the same vehicles and, for each of these vehicles, the relative position of $j_1$ with respect to $j_2$ is given by the minimal cost insertion rule. Finally, the algorithm tries to reallocate as many demand points as possible from their current assignment to $j_1$.

The *Diversification* procedure completes the main search algorithm (Fig. 2). Thus, assuming that the predefined number of iterations (#_iterations) has not yet been reached, the next procedure to be executed will be the *Delete-Redundant-SDC* procedure, which may eliminate the SDC just added by the *Diversification* procedure. To prevent the algorithm from doing so, the $r$ newly SDC are put a kind of tabu-list for the next application of the *Delete-Redundant-SDC* procedure.

## 5. Computational results

This section has three objectives: (1) identify the parameter combination that produces the best results, (2) identify the limits of the mathematical model, and (3) evaluate the quality of our heuristic approach in terms of both computational time and objective function. To this end, we built a set of numerical experiments based on randomly-generated data. The instances are characterized by the number of demand points ($n$), the number of potential SDCs ($m$), the number of different products to distribute ($t$), and the number of vehicles available ($l$). For each DP and each potential SDC, the coordinates were uniformly generated within a [0, 100] square. Distances between each pair of sites $c_{ij}$ were computed as the Euclidean integer distance. All the models were coded in Java, and the branch-and-bound algorithm of CPLEX 11.0 (with its default parameters) was used to solve the instances on a 3.00 GHz Intel Core 2 Duo PC with a 4.00 Go RAM. All the computation times in the rest of this paper are reported in seconds.

### 5.1. Setting the heuristic's parameters

Our heuristic is based on the parameters that influence the quality of the final solution. In order to evaluate the impact of these parameters, we generated 20 instances with $n = 100$ demand points, $m = 20$ satellite distribution centers, $t = 2$ types of product and $l = 2$ vehicle types. We fixed the number of restarts to 3000 and considered the following parameter combinations:

- Number of iterations (#_iterations) : 10, 20
- Number of local search iterations (#_LocalSearch_iterations): 10, 15, 20
- Number of SDCs considered in the swap ($\phi$): 4, 6, 8
- Number of SDCs considered in the diversification ($\varphi$): 3, 5, 7

Table 2 presents the numerical results associated to the 54 combinations that we used to calibrate the parameters. Our heuristic is not too sensitive to the parameters, since the worst average solution over 20 instances was 1288.15, and the best average solution was 1265.00. The most important parameter seems to be the

**Table 2**
Heuristic parameters setting.

| #_iterations | #_LocalSearch_iterations | $\varphi$ | $\phi$ | | |
|---|---|---|---|---|---|
| | | | 4 | 6 | 8 |
| 10 | 10 | 3 | 1283.95 | 1274.45 | 1277.35 |
| | | 5 | 1278.50 | 1272.15 | 1268.90 |
| | | 7 | 1284.30 | 1276.00 | 1274.20 |
| | 15 | 3 | 1285.25 | 1274.20 | 1273.45 |
| | | 5 | 1280.35 | 1279.45 | 1270.85 |
| | | 7 | 1277.00 | 1273.75 | 1275.95 |
| | 20 | 3 | 1283.85 | 1277.80 | 1279.20 |
| | | 5 | 1278.55 | 1278.55 | 1271.85 |
| | | 7 | 1279.35 | 1276.90 | 1273.75 |
| 20 | 10 | 3 | 1288.15 | 1270.10 | 1272.35 |
| | | 5 | 1277.30 | 1270.10 | *1265.00* |
| | | 7 | 1278.45 | 1278.75 | 1276.55 |
| | 15 | 3 | 1280.70 | 1277.50 | 1274.70 |
| | | 5 | 1279.40 | 1277.70 | 1277.20 |
| | | 7 | 1286.05 | 1273.20 | 1272.40 |
| | 20 | 3 | 1273.05 | 1282.45 | 1272.00 |
| | | 5 | 1270.70 | 1277.60 | 1270.40 |
| | | 7 | 1277.30 | 1279.10 | 1272.60 |

The numbers in italics correspond to the best average solution.

**Table 3**
Numerical results for the small instances ($n = 20$ DPs).

| Set | SDC | Products | Vehicles | Exact | | Heuristic | | |
|---|---|---|---|---|---|---|---|---|
| | | | | Cost | Seconds | Cost | Gap (%) | Seconds |
| 1 | 4 | 2 | 2 | 284.00 | 0.31 | 284.00 | 0.00 | 3.29 |
| 2 | 4 | 2 | 3 | 226.20 | 0.22 | 226.20 | 0.00 | 3.30 |
| 3 | 4 | 2 | 4 | 212.60 | 0.57 | 212.60 | 0.00 | 3.43 |
| 4 | 4 | 3 | 2 | 464.80 | 8.57 | 465.60 | 0.17 | 6.51 |
| 5 | 4 | 3 | 3 | 417.60 | 1.98 | 421.60 | 0.96 | 5.29 |
| 6 | 4 | 3 | 4 | 313.60 | 1.37 | 317.00 | 1.08 | 4.05 |
| 7 | 4 | 4 | 2 | 813.60 | 52.58 | 813.60 | 0.00 | 12.19 |
| 8 | 4 | 4 | 3 | 628.00 | 19.30 | 628.00 | 0.00 | 9.12 |
| 9 | 4 | 4 | 4 | 497.40 | 5.08 | 497.40 | 0.00 | 6.79 |
| 10 | 6 | 2 | 2 | 208.60 | 5.22 | 208.60 | 0.00 | 4.59 |
| 11 | 6 | 2 | 3 | 183.80 | 8.77 | 183.80 | 0.00 | 4.62 |
| 12 | 6 | 2 | 4 | 134.40 | 2.89 | 134.40 | 0.00 | 4.32 |
| 13 | 6 | 3 | 2 | 435.60 | 100.87 | 435.60 | 0.00 | 8.03 |
| 14 | 6 | 3 | 3 | 347.40 | 46.48 | 347.40 | 0.00 | 6.73 |
| 15 | 6 | 3 | 4 | 265.80 | 10.23 | 265.80 | 0.00 | 5.88 |
| 16 | 6 | 4 | 2 | 694.40 | 734.94 | 694.40 | 0.00 | 15.20 |
| 17 | 6 | 4 | 3 | 564.00 | 746.87 | 564.00 | 0.00 | 12.26 |
| 18 | 6 | 4 | 4 | 444.20 | 266.69 | 444.20 | 0.00 | 9.79 |
| 19 | 8 | 2 | 2 | 301.40 | 367.82 | 301.40 | 0.00 | 5.49 |
| 20 | 8 | 2 | 3 | 265.40 | 363.98 | 265.40 | 0.00 | 5.11 |
| 21 | 8 | 2 | 4 | 228.60 | 31.01 | 228.60 | 0.00 | 5.08 |
| 22 | 8 | 3 | 2 | 430.60 | 783.29 | 430.60 | 0.00 | 10.06 |
| 23 | 8 | 3 | 3 | 356.00 | 469.30 | 355.80 | *−0.06* | 8.84 |
| 24 | 8 | 3 | 4 | 286.80 | 313.87 | 286.80 | 0.00 | 8.04 |
| 25 | 8 | 4 | 2 | 722.80 | 942.80 | 713.40 | *−1.30* | 15.12 |
| 26 | 8 | 4 | 3 | 542.80 | 805.38 | 538.20 | *−0.85* | 12.16 |
| 27 | 8 | 4 | 4 | 432.20 | 731.40 | 432.20 | 0.00 | 10.00 |
| 28 | 10 | 2 | 2 | 265.40 | 364.25 | 265.40 | 0.00 | 5.09 |
| 29 | 10 | 2 | 3 | 230.20 | 42.76 | 230.20 | 0.00 | 5.25 |
| 30 | 10 | 2 | 4 | 218.60 | 41.76 | 218.60 | 0.00 | 5.72 |
| 31 | 10 | 3 | 2 | 355.00 | 1180.55 | 354.80 | *−0.06* | 9.95 |
| 32 | 10 | 3 | 3 | 296.00 | 426.89 | 296.60 | 0.20 | 8.31 |
| 33 | 10 | 3 | 4 | 245.80 | 488.41 | 245.80 | 0.00 | 8.09 |
| 34 | 10 | 4 | 2 | 624.40 | 1672.34 | 616.40 | *−1.28* | 14.31 |
| 35 | 10 | 4 | 3 | 517.60 | 1156.36 | 516.60 | *−0.19* | 11.70 |
| 36 | 10 | 4 | 4 | 406.60 | 813.56 | 406.60 | 0.00 | 9.72 |
| Average | | | | | 361.35 | | −0.11 | 7.87 |
| Minimum | | | | | 0.22 | | −1.30 | 3.29 |
| Maximum | | | | | 1672.34 | | 1.08 | 15.20 |

**Table 4**
Results for instances with $n = 30$ DP.

| Set | SDC | Products | Vehicles | Exact | | | Heuristic | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Cost | Gap (%) | Seconds | Cost | Gap (%) | Seconds |
| 1 | 9 | 3 | 3 | 528.20 | 14.45 | 4924 | 526.40 | −0.34 | 17.72 |
| 2 | 9 | 3 | 4 | 925.60 | 34.83 | 5829 | 906.00 | −2.12 | 31.53 |
| 3 | 9 | 4 | 3 | 421.40 | 7.37 | 3127 | 421.20 | −0.05 | 15.14 |
| 4 | 9 | 4 | 4 | 665.80 | 28.31 | 5784 | 675.60 | 1.47 | 24.29 |
| 5 | 12 | 3 | 3 | 504.80 | 28.49 | 7200 | 503.20 | −0.32 | 18.83 |
| 6 | 12 | 3 | 4 | 903.80 | 50.90 | 7200 | 857.20 | −5.16 | 32.31 |
| 7 | 12 | 4 | 3 | 419.20 | 17.90 | 6002 | 418.00 | −0.29 | 15.97 |
| 8 | 12 | 4 | 4 | 691.60 | 39.02 | 7200 | 662.60 | −4.19 | 24.92 |
| Average | | | | | 27.66 | 5908 | | −1.37 | 22.59 |

**Table 5**
Results for instances with $n = 40$ DP.

| Set | SDC | Products | Vehicles | Exact | | | Heuristic | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Cost | Gap (%) | Seconds | Cost | Gap (%) | Seconds |
| 1 | 12 | 3 | 3 | 655.80 | 47.58 | 7200 | 632.20 | −3.60 | 31.76 |
| 2 | 12 | 3 | 4 | 1191.80 | 59.86 | 7200 | 1120.80 | −5.96 | 54.74 |
| 3 | 12 | 4 | 3 | 503.60 | 53.34 | 7200 | 487.20 | −3.26 | 26.90 |
| 4 | 12 | 4 | 4 | 1011.80 | 57.81 | 7200 | 961.00 | −5.02 | 46.34 |
| 5 | 16 | 3 | 3 | 652.60 | 58.88 | 7200 | 478.00 | −26.75 | 30.75 |
| 6 | 16 | 3 | 4 | 976.00 | 65.74 | 7200 | 931.00 | −4.61 | 60.65 |
| 7 | 16 | 4 | 3 | 463.60 | 46.78 | 7200 | 384.00 | −17.17 | 26.55 |
| 8 | 16 | 4 | 4 | 791.80 | 64.89 | 7200 | 769.00 | −2.88 | 50.29 |
| Average | | | | | 58.86 | 7200 | | −8.66 | 41.00 |

**Table 6**
Results for instances with $n = 50$ DP.

| Set | SDC | Products | Vehicles | Exact | | | Heuristic | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Cost | Gap (%) | Seconds | Cost | Gap (%) | Seconds |
| 1 | 12 | 3 | 3 | 843.60 | 67.35 | 7200 | 775.80 | −8.04 | 50.00 |
| 2 | 12 | 3 | 4 | 660.60 | 63.70 | 7200 | 582.20 | −11.87 | 41.89 |
| 3 | 12 | 4 | 3 | 1432.00 | 75.01 | 7200 | 1350.40 | −5.70 | 90.86 |
| 4 | 12 | 4 | 4 | 1089.60 | 73.08 | 7200 | 996.00 | −8.59 | 69.22 |
| 5 | 16 | 3 | 3 | 807.80 | 71.23 | 7200 | 720.80 | −10.77 | 57.29 |
| 6 | 16 | 3 | 4 | 627.00 | 63.74 | 7200 | 561.40 | −10.46 | 49.10 |
| 7 | 16 | 4 | 3 | 1310.00 | 81.67 | 7200 | 1146.40 | −12.49 | 88.42 |
| 8 | 16 | 4 | 4 | 984.80 | 79.77 | 7200 | 861.40 | −12.53 | 72.74 |
| Average | | | | | 71.94 | 7200 | | −10.06 | 64.94 |

number of iterations (i.e., the number of times that the global loop, including the diversification, is applied). The performance does not seem to be linked to the number of local search iterations, which means that the best solution may be obtained early on. Consequently, in all the forthcoming tests, we used the following parameters: #_iteration = 20, #_LocalSearch_iterations = 10, $\varphi = 5$ and $\phi = 8$.

### 5.2. Computational results

In this sub-section, our heuristic's performance is compared to the performance of Cplex in order to evaluate the optimality gap produced by the heuristic. We generated some other instance sets in order to establish independence from the instances used to calibrate our parameters. These new instances are smaller so that they can be managed by Cplex. For these new instances, four categories of vehicles are considered. Their capacities are {50, 75, 100, 150} units, respectively. The instances with different vehicle types ($l = 2$, $l = 3$ and $l = 4$) have vehicle capacities of {50, 75}, {50, 75, 100}, and {50, 75, 100, 150} units, respectively. For a given instance, the vehicles are added until their total capacity is equal or greater than the total demand, multiplied by a factor of 1.2.

Table 3 shows the results obtained for these small instances, with only $n = 20$ demand points. We generated five instances for each combination of the number of SDC ($m = 4$, 6, 8, and 10), the number of products ($t = 2$, 3, and 4), and the number of vehicles ($l = 2$, 3, and 4). This lead to 36 sets of five instances each, for a total of 180 instances. Each line in Table 3 reports the average cost over five instances.

The results of the mathematical model (Section 2) are provided under the header, *Exact*, and our heuristic approach (Section 4) under the header, *Heuristic*. Column *Gap* (%) refers to the average cost percentage of the heuristic solution over the exact solution (or the best integer solution, if proof of optimality was not obtained). For these small instances, Cplex was allowed to run for up to 1800 s. Within this time limit, it was able to give proof of optimality in 152 out of 180 cases. The average computational times range from only a fraction of a second up to more than 1600 s, but, overall, Cplex is efficient solving these kinds of instances.

The figures in Table 3 confirm the excellent performance for our heuristic. In fact, for 26 out of 36 sets, the heuristic average gap was 0%, which means that, for the five instances in each of these 26 sets, the heuristic found the best known solutions. For the other six sets (italics in Table 3), the average gap of the heuristic was negative, meaning that the heuristic produces a better solution than Cplex in the allotted time. Globally, the heuristic average gap is −0.11, with an average computing time of 7.8 s.

In Tables 4–6, we report the results of the larger instances in order to evaluate the ability of our heuristic to solve real problems efficiently. We generated 24 new sets of five instances each, with different combinations of numbers of DP, SDC, products and vehicle types. The instances – which had up to 50 DP, 20 SDC, four products and four vehicle types – were solved by running Cplex for up to 7200 s for each instance. For these new 120 instances, Cplex was only able to find eight proven optimal solutions, all of them for $n = 30$ DP instances, shown in Table 4. However, the average optimality gap of Cplex was 27.66%. For Tables 4–6, the gap for the heuristic was computed against Cplex's best-known integer solution. Table 4 shows that the heuristic improves Cplex results by 1.37%, using an average of only 23 s of computing time.

Results with $n = 40$ and 50 clients are presented in Tables 5 and 6. For these instances, Cplex always reaches the maximum time limit of 7200 s per instance, resulting in optimality gap of 56.86 for 40 DP and of 71.94% for 50 DP. Clearly, Cplex has become inefficient at solving these instances.

For the 40-DP instances, our heuristic improves the Cplex results by 8.66% in only 41.00 s. The results for the 50-DP instances are reported in Table 6: the average improvement is 10.06% in only 64 s of computing time. These results clearly demonstrate that the heuristic produces high-quality solutions in very short computing times.

## 6. Conclusions

This article tackles the location of satellite distribution centers (SDCs) used to deliver humanitarian aid to the population in a disaster area. We model this situation as a generalization of the covering tour problem. Our numerical experiments on randomly-generated data confirm that only very small instances can be solved efficiently using the mathematical model. We proposed a heuristic approach to solve instances whose size could be of practical interest. Our multi-start heuristic produces high-quality solutions and solves realistic instances in reasonable computing times. Both in small and large instances, our heuristic produced better average results than Cplex in only a fraction of the computing time.

## Acknowledgements

This research was partially supported by Grants [OPG 0293307 and OPG 0172633] from the Canadian Natural Sciences and Engineering Research Council (NSERC). This support is gratefully acknowledged. The authors are grateful to the anonymous referees for their useful and constructive comments and suggestions.

## References

Altay, N., Green III, W.G., 2006. OR/MS research in disaster operations management. European Journal of Operational Research 175, 475–493.

Baldacci, R., Boschetti, M.A., Maniezzo, V., Zamboni, M., 2005. Scatter search methods for the covering tour problem. In: Sharda, R. et al. (Eds.): Metaheuristic Optimization via Memory and Evolution. Operations Research/Computer Science Interfaces Series, vol. 30, pp. 59–91.

Current, J.R., Schilling, D.A., 1989. The covering salesman problem. Transportation Science 23, 208–213.

Current, J.R., Schilling, D.A., 1994. The median tour and maximal covering tour problems: formulations and heuristics. European Journal of Operational Research 73, 114–126.
Fischetti, M., Toth, P., 1988. An additive approach for the optimal solution of the prize-collecting traveling salesman problem. In: Golden, B.L., Assad, A.A. (Eds.), Vehicle Routing: Methods and Studied. North-Holland, Amsterdam, pp. 319–343.
Gendreau, M., Laporte, G., Semet, F., 1997. The covering tour problem. Operations Research 45, 568–576.
Golden B., Naji-Azimi Z., Raghavan S., Salari M., Toth P., Forthcomming. The generalized covering salesman problem. INFORMS Journal on Computing.
Hachicha, M., Hodgson, M.J., Laporte, G., Semet, F., 2000. Heuristics for the multi-vehicle covering tour problem. Computers and Operations Research 27, 29–42.
Hodgson, M.J., Laporte, G., Semet, F., 1998. A covering tour model for planning mobile health care facilities in Suhum district, Ghana. Journal of Regional Science 38, 621–639.
Jozefowiez, N., Semet, F., Talbi, E.-G., 2007. The bi-objective covering tour problem. Computers and Operations Research 34, 1929–1942.
Kara, I., Laporte, G., Bektas, T., 2004. A note on the Lifted Miller–Tucker–Zemlin subtour elimination constraints for the capacitated vehicle routing problem. European Journal of Operational Research 158, 793–795.
Kedad-Sidhoum, S., Hung Nguyen, V., 2010. An exact algorithm for solving the ring star problem. Optimization 59, 125–140.
Laporte, G., Martello, S., 1990. The selective traveling salesman problem. Discrete Applied Mathematics 26, 193–207.
Miller, C.E., Tucker, A.W., Zemlin, R.A., 1960. Integer programming formulations and traveling salesman problems. Journal of the Association for Computing Machinery 7, 326–329.
Moreno Pérez, J.A., Moreno-Vega, J.M., Rodriguez Martin, I., 2003. Variable Neighborhood tabu search and its application to the median cycle problem. European Journal of Operational Research 51, 365–378.
Nolz, P.C., Doerner, K.F., Gutjahr, W.J., Hartl, R.F., 2010. A bi-objective metaheuristic for disaster relief operation planning. In: Coello, C.A. et al., (Eds.) A dv. in Multi-Objective Nature Inspired Computing. SCI 272, pp. 167–187.
Rekik, M., Ruiz, A., Renaud, J., Berkoune, D., 2011. A decision support system for distribution network design for disaster response. Working paper CIRRELT-2011-36, Interuniversity research centre on enterprise networks, logistics and transportation.
Renaud, J., Boctor, F., Laporte, G., 2004. Efficient heuristics for median cycle problems. Journal of the Operational Research Society 55, 179–186.
Tricoire, F., Graf, A., Gutjahr, W.J., 2012. The bi-objective stochastic covering tour problem. Computers & Operations Research 39, 1582–1592.
Tzeng, G.-H., Cheng, H.-J., Huang, T.D., 2007. Multi-objective optimal planning for designing relief delivery systems. Transportation Research Part E 43, 673–686.