Computers & Operations Research 40 (2013) 1708–1715

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

The synchronized arc and node routing problem: Application to road marking

M. Angélica Salazar-Aguilar [a,*], André Langevin [b], Gilbert Laporte [a]

[a] CIRRELT and Canada Research Chair in Distribution Management, HEC Montréal, Canada
[b] CIRRELT and Département de mathématiques et de génie industriel, École Polytechnique de Montréal, Canada

ARTICLE INFO

Available online 16 January 2013

Keywords:
Arc routing
Node routing
Road maintenance
Synchronized routes
Adaptive large neighborhood search

ABSTRACT

This paper introduces the synchronized arc and node routing problem, inspired from a real application arising in road marking operations. In this setting, several capacitated vehicles are used to paint lines on the roads and a tank vehicle is used to replenish the painting vehicles. The aim of the problem is to determine the routes and schedules for the painting and replenishment vehicles so that the pavement marking is completed within the least possible time. This must be done in such a way that the routes of the painting and replenishment vehicles are synchronized. An adaptive large neighborhood heuristic is described and evaluated over a large set of artificial instances.

1. Introduction

This work introduces the synchronized arc and node routing problem (SANRP), inspired from a real application arising in road marking operations. A number of applications of the SANRP are encountered in other contexts such as door-to-door delivery of mail, newspapers or promotional material. A common feature of these applications is the existence of multiple capacitated arc routes which require replenishments. In general, replenishments are carried out by a dedicated vehicle. In this work, to make our presentation more concrete, we will describe the SANRP in terms of the road marking application. In this problem, several capacitated vehicles are used to paint lines on the roads and a tank vehicle can replenish the painting vehicles once or several times a day, depending on the quantity of paint they need. This means that the painting vehicles need not return to the depot when they are replenished. As is standard in capacitated arc routing problems [6], some road segments must be painted (they are required), whereas others do not need to be painted (they are non-required). In the SANRP, there are multiple capacitated painting vehicles and a replenishment vehicle initially located at a common depot. The aim is to determine a set of routes for the painting vehicles as well as a route for the replenishment vehicle so that the pavement marking is completed within the least possible time and the painting vehicles never run out of paint. Hence, it is desirable to synchronize the routes of the two vehicle types.

In the SANRP two routing problems must be solved simultaneously: a multi-vehicle capacitated arc routing problem and a node routing problem. The nodes at which the arc routes and the node route intersect are not given a priori, but must be determined together with the routes themselves. Finally, the routes generated should be synchronized so as to reduce the waiting time at the refill nodes. We analyze three replenishment policies: (i) there is no replenishment vehicle and the painting vehicles return to the depot when they need a refill; (ii) the painting vehicles do not return to the depot when they need a refill, but are serviced by the replenishment vehicle; and (iii) a combination of the first two policies, meaning that the painting vehicles can be refilled from the replenishment vehicle or directly from the depot. Policies (ii) and (iii) are compared with policy (i) which is the standard practice.

The SANRP is considerably more difficult to solve than either the CARP or the node routing problem which it integrates. This is so because any change in the solution of one of the two subproblems affects the solution of the other one. It is of course impractical to solve the SANRP exactly for any realistic size. We have designed a powerful adaptive large neighborhood search heuristic (ALNS) in which the solution space is explored by means of several operators. The heuristic was successfully tested over a large set of artificial.

The remainder of the paper is structured as follows. Section 2 summarizes a number of contributions related to the SANRP. Section 3 formally describes the SANRP. The solution procedure is described in Section 4, and computational results are presented in Section 5, followed by conclusions in Section 6.

* Corresponding author. Current address. Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León (UANL), México. Tel.: +52 8111923672.
E-mail addresses: angelica.salazar.a@gmail.com, angysalag@gmail.com (M.A. Salazar-Aguilar), andre.langevin@polymtl.ca (A. Langevin), gilbert.laporte@cirrelt.ca (G. Laporte).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.01.007

2. Related work

Very few studies have been performed on arc routing or vehicle routing problems with synchronization constraints. Amaya et al. [1] have introduced the CARP with refill points (CARP-RP), also in the context of road marking. Their paper considers a problem with a painting vehicle and a replenishment vehicle, which can supply the painting vehicle at any road junction and must return to the depot after each refill. The authors have proposed an integer linear programming formulation and a cutting plane algorithm for this problem and have applied it to instances involving between 20 and 70 nodes, and between 50 and 595 arcs. For larger instances computational times become rather important. An extension of this work, by the same authors [2], is called the capacitated arc routing problem with refill points and multiple loads (CARP-RP-ML). Contrary to what happens in the CARP-RP, in the CARP-RP-ML the replenishment vehicle does not return to the depot after reloading the painting vehicle. The authors describe a heuristic procedure based on the cluster-first-route-second principle. The largest instances solved contain 70 nodes and 600 arcs. Note that the two papers of Amaya et al. [1,2] are rooted in the same context as the SANRP proposed in the present paper, but our problem is more general since it involves several painting vehicles instead of only one, and it considers three replenishment policies. Moreover, in the SANRP the refill nodes location and routing decisions of multiple vehicles are intertwined.

Rosa et al. [16] have introduced a related problem called the arc routing and scheduling problem with transshipment (ARPT). They consider two types of vehicles: small vehicles are used to collect garbage from the streets and to bring it to a transfer station, and large vehicles are used to transport the garbage from the transfer station to a dump site. The ARPT consists of designing a set of routes for the small vehicles and a set of schedules for large ones in order to minimize the total travel time. In this problem, the location of the meeting point (transfer station) for two vehicle types is an exogenous decision, whereas in the SANRP it is endogenous. In addition, in the ARPT the route followed by the large vehicles is predefined; in contrast, in the SANRP the route of the replenishment vehicle must be designed by taking into account the refill nodes chosen for the painting vehicles.

Pia and Filippi [11] have studied a real-life waste collection problem arising in Due Carrera, a town located in Northern Italy. They called this problem the CARP-MD (capacitated arc routing problem with mobile depots). This problem consists of determining meeting points along the vehicle routes so that the smaller vehicles can dump their content into a larger vehicle before resuming their route. In the case study carried out in Due Carrera, a fleet of four vehicles of two types was available to collect the waste: two large ones called compactors, and two small ones called satellites, and there were restrictions on the streets each vehicle type could use. The CARP-MD is related to the SANRP in the following way: each satellite vehicle can be viewed as a painting vehicle and the compactors correspond to the replenishment vehicle. However, there are two main differences between the two problems. First, in the SANRP the replenishment vehicle is exclusively used to refill the tanks of the painting vehicles, which means that it cannot service the streets in the network. In contrast, a compactor in the CARP-MD is used to service the streets and to pick up trash from the satellites. Second, the SANRP is a pure combination of arc routing and node routing problems with synchronization constraints, whereas the CARP-MD has been described as a location-arc routing problem.

In mail delivery, the location of relay boxes along postman routes, studied by Bouliane and Laporte [4], is another application related to the SANRP. It consists of determining the location of relay boxes in such a way that when a postman runs out of mail, he goes to the relay box located on his route in order to replenish his bag. The location of these relay boxes is determined by considering the maximal weight a postman can carry, as well as the distance he travels to reach the relay boxes. These must be loaded in advance by a vehicle. When a postman needs to replenish his bag, he goes to the relay box which contains the mail he needs to deliver along the next segment of his route. A mail bag can be therefore viewed as a painting vehicle in the SANRP. However, no synchronization is required between the replenishment vehicle and the postman in this application.

3. Formal problem description

The SANRP is defined on a directed graph $G=(V,A'\cup A)$, where $V=\{0,\dots,n\}$ is the node set, node 0 represents the depot at which all vehicle routes start and end, A is the set of required arcs (street segments that must be painted), and A' is the set of non-required arcs (street segments that do not have to be painted). All arcs $(i,j)\in A$ must be serviced once in the solution, and any arc $(i,j)\in A'\cup A$ can be *deadheaded* any number of times, i.e., it can be traversed without being serviced. Every arc $(i,j)\in A$ has a non-negative demand p_{ij} which represents the amount of paint required to mark the street segment from i to j. Two kinds of vehicles are available: a replenishment vehicle r_v and a set R of homogeneous vehicles called painting vehicles. Each painting vehicle $r\in R$ has a tank of capacity Q which most of the time is not sufficient to paint the roads assigned to it. Therefore each painting vehicle requires a single or multiple refillings to perform its work. The replenishment vehicle r_v is used to supply paint to the painting vehicles, and we assume its capacity is sufficient to satisfy the total demand of the painting vehicles. Each painting vehicle $r\in R$ travels at speed s (km/h) when it is not painting a street segment, and at speed $s'<s$ when it is painting. Therefore, each arc $(i,j)\in A\cup A'$ has two associated travel times called t_{ij} and t'_{ij} for service and traversal, respectively. Initially, all painting vehicles leave the depot with a full tank, and every node on their route is considered as a potential refilling point. Replenishment can be made preventively before the tanks of the painting vehicles become empty. The refill nodes must be chosen so that the painting vehicles and the replenishment vehicle can meet at the same time without incurring too much waiting. A painting vehicle cannot start servicing an arc if it does not have enough paint for it. The amount of paint required between any two consecutive refill nodes should not exceed Q.

The goal of the SANRP is to determine a set of feasible and synchronized routes minimizing the *makespan*, i.e., the duration of the longest route. This helps generate balanced routes, which are viewed as a desirable feature in many contexts.

Amaya et al. [1] show that even in the simple case, when there is only one painting vehicle and one replenishment vehicle, the problem is NP-hard and the size and complexity of the related formulation are substantial. Our problem is even more difficult because there are several vehicles, preventive refills, and synchronization constraints. Thus formulating it would only be a mathematical exercise and no further insight into our understanding of the problem, and could not be used to solve even relatively small instances. Therefore, the SANRP can now be summarized as follows:

Objective: Minimize the duration of the longest route (painting vehicles).

Constraints:

- The painting vehicles should not run out of paint.
- The refill nodes must be determined together with the routes themselves.
- Preventive refills are allowed.

- The routes generated should be synchronized so as to reduce the waiting time at the refill nodes.
- All required arcs must be served.
- The capacity of the replenishment vehicle is sufficient to satisfy the total demand of the painting vehicles.
- Deadheading is allowed.
- All routes start and end at a common depot.

In our application, the road marking process can be carried out by using one of the two types of paint: epoxy or water paint. Epoxy paint is more resistant and more durable than water paint, but it is also more expensive. Typically, in regions with severe winter conditions, like most parts of Canada, if epoxy paint is used, some arcs in the network do not need to be re-marked after the winter season because the painted lines are still in good condition (they are non-required arcs). In contrast, if the roads are marked with water paint, the entire network has to be re-marked (all arcs are required) after winter is over. The two cases are analyzed in the experimental work of this paper.

We have also implemented and compared three different replenishment policies called D, R, and RD. Under policy D, the replenishment vehicle does not exist, and therefore the painting vehicles have to return to the depot whenever they need a refill. This is the most common policy adopted in practice. Because several painting vehicles may arrive at the depot within a short time interval, a queue is likely to form, which lengthens the replenishment time. In our implementation, we model this operation as a single server queue operating under the first-in first-out discipline. Under policy R, the replenishment vehicle is the only source of replenishment, which means that each time a painting vehicle needs a refill, it has to wait for the replenishment vehicle. Finally, under the RD policy, the painting vehicles can be refilled directly from the depot or from the replenishment vehicle. The choice of going to the depot or waiting for the replenishment vehicle depends on the current location of the replenishment vehicle as well as on the current status of the queue at the depot, and the option yielding the least time is selected.

4. Adaptive large neighborhood search metaheuristic

Several constraints must be handled simultaneously when solving the SANRP, which poses a methodological challenge. We have therefore opted for the development of an adaptive large neighborhood search (ALNS) metaheuristic which can easily handle multiple constraints of different natures. This methodology was introduced by Ropke and Pisinger [15] and Pisinger and Ropke [12] in the context of the deterministic vehicle routing problem and several of its extensions. It has since been applied to a variety of problems including stochastic arc routing [8], vehicle scheduling [3,10], districting [9], inventory-routing [5], and fixed charged network flow problems [7].

Our algorithm generates a first set of arc routes for the painting vehicles and a node route for the replenishment vehicle, and then attempts to improve them by means of several destroy and repair operators. In our algorithm these operators are coupled. The choice of a destroy/repair operator is controlled dynamically according to its past performance. Given an initial solution, the ALNS heuristic is applied for a given number of iterations. The selection of an operator and the iterative process are similar to those of Salazar-Aguilar et al. [17]. Several initial solutions are successively used as an input and the best known solution identified during the entire procedure is reported.

In the SANRP, one of the hardest constraints is route synchronization. We have tackled this constraint by following a constructive procedure combined with GRASP, a metaheuristic put forward by Resende and Ribeiro [13]. This algorithm is a multi-start metaheuristic for combinatorial problems, in which each iteration consists of two phases: construction of a greedy randomized solution and local search. In our GRASP implementation the greedy randomized solutions are generated by adding elements (refill nodes) to the route of the replenishment vehicle. At each iteration, the choice of the next refill node to be added is determined by ordering all elements in a candidate list C with respect to a greedy function which measures the cost of selecting the element. The GRASP is adaptive because the cost associated with the elements of C are updated at each iteration of the construction phase to reflect the changes brought by the selection of the previous element. The probabilistic feature of a GRASP is operationalized by randomly choosing one of the top candidates in the list, but not necessarily the best one. In order to diversify the search, the element to be added is randomly chosen from a restricted candidate list (also known as RCL), which contains only well-ranked candidate elements lying within a range determined by a quality parameter α. If $\alpha = 1.0$, the candidate selection is completely random; at the other extreme, if $\alpha = 0.0$, the selection is purely greedy.

The ALNS developed in this work is based on the replenishment policy R. The application of policies D and RD is carried out starting with the best solution (P^*,φ^*) generated under policy R, where P^* is the set of routes for the painting vehicles and φ^* is the set of refill nodes which define the route of the replenishment vehicle. In another words, the routes of the painting vehicles and the refill nodes (not the order in which they are visited) are the same for all the three policies. For policy D, at each iteration, the vehicle returning to the depot is the first that needs a refill. When all refills have been performed, the makespan is computed as the time at which the last vehicle returns to the depot. For policy RD, the replenishment vehicle is initially located at the depot. We identify the first painting vehicle needing a refill. We then evaluate the time this vehicle must wait for the replenishment vehicle, as well as the time it would take to go to the depot and return to its current position on its route. The option yielding the least time is selected. This process continues iteratively until no more refills are required, and the makespan is computed as in the previous policies.

4.1. Construction phase

Our constructive procedure (Algorithm 1) works as follows. The $|R|$ painting routes are initialized by choosing $|R|$ spatially dispersed seed arcs and are gradually extended by applying a route balancing criterion. To keep the balance, an arc is added to the shortest route at each iteration. Once the painting routes have been generated, the potential refill nodes on each of them are identified (see Algorithm 2), and the route of the replenishment vehicle is then constructed by means of a GRASP (Algorithm 3).

Algorithm 1. Construction phase of the ALNS heuristic.

```
Input: $G = (V, A' \cup A)$: Instance graph
  $R$: Set of painting vehicles
  $Q$: Tank capacity for any $r \in R$
Output: $(P^*, \varphi^*)$ where $P^*$ corresponds to the set of routes for
all painting vehicles and $\varphi^*$ is the set of refill nodes defining
the route of the replenishment vehicle

1: Set $m = |R|, \varphi = \emptyset$, $P = \{P_1, \ldots, P_m\}$ where $P_k = \emptyset, \forall k = 1, \ldots, m$
2: Identify $m$ seed arcs (with maximum dispersion):
   $\sigma = \{(i_1, j_1), \ldots, (i_m, j_m)\} \subset A$
3: Initialization of routes, $P_k \leftarrow (i_k, j_k), k = 1, \ldots, m$
4: Set $A \leftarrow A \backslash \sigma$
5: while $A \neq \emptyset$ do
6:   Set $\beta = \arg\min_{k=1,\ldots,m}$ {path length $(P_k)$}
```

```
7:      Let $(g,h)$ be the first arc of $P_\beta$ and $(u,v)$ be the last arc of $P_\beta$
8:      Let $E_\beta$ be the union of the required incoming arcs to $(g,h)$
        and outgoing arcs from $(u,v)$.
9:      if $E_\beta \neq \emptyset$ then
10:         Choose randomly $(a,b) \in E_\beta$
11:     else
12:         Choose $(a,b) \in A$ such that the path length between $(a,b)$
            and $P_\beta$ is minimized
13:     end if
14:     $P_\beta \leftarrow P_\beta \cup \{(a,b)\}$
15:     $A \leftarrow A \setminus \{(a,b)\}$
16: end while
17: $\Pi = \text{PotentialRefillNodes}(P)$
18: for $\alpha = 0.0, 0.2, \ldots, 1.0$ do
19:     $\varphi_\alpha = \text{ReplenishmentVehicleRoute}(P,\Pi,\alpha)$ {GRASP strategy}
20:     if $\alpha = 0.0$ then
21:         Set $P^* = P$ and $\varphi^* = \varphi_\alpha$ {Best initial solution}
22:     end if
23:     if $(\text{Makespan}(P,\varphi_\alpha) < \text{Makespan}(P^*,\varphi^*))$ then
24:         Set $P^* = P$ and $\varphi^* = \varphi_\alpha$
25:     end if
26: end for
27: return $P^*,\varphi^*$
```

Algorithm 2. PotentialRefillNodes(P).

```
Input: $P$: Set of routes for the painting vehicles
Output: $\Pi = \{\pi_1, \ldots, \pi_m\}$: set of potential refill nodes for all
$P_k \in P, k = 1, \ldots, m$

1:  for all $k = 1, \ldots, m$ do
2:      $\pi_k = \emptyset$: Ordered set of potential refill nodes on route $P_k$
3:      if $\sum_{(i,j) \in P_k} p_{ij} \geq Q$ then
4:          Let $(a,b)$ be the first serviced arc in $P_k$
5:          Let $(y,z)$ be the last serviced arc in $P_k$
6:          Extract $\overline{P}_k \subset P_k : \overline{P}_k = \{(a,b), \ldots, (y,z)\}$ sequence of arcs
            from $(a,b)$ to $(y,z)$ in $P_k$
7:          Let $\pi_k$ be the set of nodes $i$ such that $(i,j)$ or $(j,i)$ is a
            non-serviced arc in $\overline{P}_k$
8:          Let $\Delta$ be the set of sequences of consecutive serviced
            arcs in $\overline{P}_k$
9:          Set $f = |\Delta|$
10:         for all $t = 1, \ldots, f$ do
11:             Set $e = |\delta_t|$: number of arcs in the sequence $\delta_t \in \Delta$,
                where $\delta_t = \{(i_1,j_1), \ldots, (i_e,j_e)\}$
12:             if $(e > 1)$ and $(\sum_{r=1}^{e} p_{i_r j_r} > Q)$, $(i_r,j_r) \in \delta_t$ then
13:                 Find the maximum $\lambda$ such that
                    $\sum_{r=1}^{\lambda} p_{i_r j_r} \leq Q, (i_r,j_r) \in \delta_t$
14:                 {Identification of critical nodes, forward direction}
15:                 Set $\pi_k \leftarrow \pi_k \cup \{j_\lambda\}$
16:                 Find the minimum $\lambda > 1$ such that
                    $\sum_{r=\lambda}^{e} p_{i_r j_r} \leq Q, (i_r,j_r) \in \delta_t$
17:                 {Identification of critical nodes, backward direction}
18:                 Set $\pi_k \leftarrow \pi_k \cup \{i_\lambda\}$
19:             end if
20:         end for
21:     end if
22: end for
23: return $\Pi$
```

Note that there are two types of potential refill nodes (Algorithm 2): those that belong to a non-serviced arc, and those referred to as critical nodes where the painting vehicle could run out of paint. Critical nodes are identified in sequences of consecutive serviced arcs for which the required amount of paint exceeds Q. A critical node is the first node of the sequence at which the painting vehicle does not have enough paint to service the next arc. In our algorithm the critical nodes are identified by following the routes in each of their two directions (see Algorithm 2).

Algorithm 3. ReplenishmentVehicleRoute(P,Π,α).

```
Input: $P = \{P_1, \ldots, P_m\}$: Set of routes for the painting vehicles
    $\Pi = \{\pi_1, \ldots, \pi_m\}$: Potential refill nodes for all routes in $P$
    $\alpha$: Quality parameter used in the restricted candidate list
(RCL)
Output: $\varphi$: Set of refill nodes defining the route of the
replenishment vehicle

1:  $\varphi = \emptyset$, $H = \emptyset$, $C = \emptyset$, $l = 0$ {location of the replenishment
    vehicle}
2:  Let $r(i)$ be the index of a refill for vehicle $i$, $i = 1, \ldots, m$
3:  Set $r(i) = 1, \forall i = 1, \ldots, m$
4:  Let $\overline{\pi}_{r(i)} \subset \pi_i$ be the set of potential refill nodes for the $r(i)^{th}$
    refill in $P_i$
5:  Set $H = \bigcup_{i=1}^{m} \overline{\pi}_{r(i)}$
6:  if $H \neq \emptyset$ then
7:      For each $j \in H$ compute $\vartheta(l,j)$ and add $j$ in the candidate
        list $C$
8:      Compute the RCL for the $\alpha$ given
9:      Random selection of a refill node $x \in \text{RCL}$
10:     Set $l = x$ {new location of the replenishment vehicle}
11:     Let $k$ be the index of the route associated with $x$
12:     Set $r(k) = r(k) + 1$ {go for the next refill in $P_k$}
13:     Set $\varphi = \varphi \cup \{x\}$
14: else
15:     return $\varphi$
16: end if
```

The GRASP applied to create the route of the replenishment vehicle is described in Algorithm 3. The goal of this procedure is to reduce the waiting time at the refill nodes. If the replenishment vehicle is located at node i and the next refill node has to be selected, then the set H of potential refill nodes for all painting routes is determined, and the cost (waiting time) associated with the choice of a node j is computed by means of a greedy function $\vartheta(i,j)$. All elements in H constitute the list C of candidates for the next refill node. If $j \in H$ belongs to the route of painting vehicle r, then the cost of choosing j as the next refill node is given by

$$\vartheta(i,j) = |(t_f(i) + \overline{t}_{ij} - t_r(j)|, \tag{1}$$

where, $t_f(i)$ is the time at which the replenishment vehicle finishes its service at node i, $\overline{t}_{ij}$ is the travel time required by the replenishment vehicle to go from i to j, and $t_r(j)$ is the target arrival time of the painting vehicle r at node j. If the tank of painting vehicle r contains an amount of paint greater than half of its capacity, then the greedy function is twice the value computed by (1). We adopt this strategy to give a higher priority to those vehicles for which a refill is crucial to continue their operations.

Following the GRASP strategy, an RCL is defined as the set of the most attractive refill nodes determined by the quality parameter $\alpha \in [0,1]$. More specifically, the RCL is computed as follows:

$$\vartheta_{\min} = \min_{j \in C} \vartheta(i,j), \tag{2}$$

$$\vartheta_{\max} = \max_{j \in C} \vartheta(i,j), \tag{3}$$

$$\text{RCL} = \{j \in C : \vartheta(i,j) \in [\vartheta_{\min}, \vartheta_{\min} + \alpha(\vartheta_{\max} - \vartheta_{\min})]\}. \tag{4}$$

In our implementation, the elements in the RCL are ordered by following an increasing order of their evaluation in the greedy function (1). Then an element (refill node) of the RCL is chosen and included in the route of the replenishment vehicle. The process continues until there are no more refills to perform. The replenishment vehicle then returns to the depot. We have used six values of α to diversify the initial solutions (see Algorithm 3).

In the construction phase (Algorithm 1) several initial sets of routes for the painting vehicles are created and for each set, the GRASP generates six different solutions by using the values of α (Algorithm 3). The best solution among these six trials is then selected as an input to the improvement phase of our ALNS.

4.2. Improvement phase

The improvement phase is applied to each initial solution until a stopping criterion is reached (Algorithm 4). We have developed seven destroy/repair operators which consist of removing and inserting of sequences of arcs, interchanging of arc statuses, and reordering of nodes (refill nodes) in the route followed by the replenishment vehicle. Each destroy/repair operator $\omega \in \Omega = \{o_1,\ldots,o_7\}$ has a given weight, ρ_ω, which is adjusted dynamically during the search. At the first iteration, all operators $\omega \in \Omega$ have the same weight $\rho_\omega = 1/|\Omega|$. These weights are then adjusted dynamically, based on the past performance of the operator. That is, $\rho(\omega) = S(\omega)/U(\omega)$ for each $\omega \in \Omega$, where $U(\omega)$ is the number of times the operator ω has been selected, and $S(\omega)$ is the number of times for which ω has improved the current solution in the past iterations of the segment. At each iteration, the choice of a destroy/repair operator is based on a roulette-wheel selection principle and the probability of selection is computed as $\tau_\omega = \rho_\omega / \sum_{o \in \Omega} \rho_o$, $\forall \omega \in \Omega$. In our implementation, the maximum number of iterations is 1500, and a search segment is composed of 50 iterations. The operators are now described.

Algorithm 4. Improvement phase of the ALNS heuristic.

```
Output:
  $(P,\varphi)$: Initial feasible solution
  $\Omega$: Set of destroy/repair operators
Output: $(P^*,\varphi^*)$: Incumbent solution
  Initialization
  $P^* \leftarrow P$, $\varphi^* \leftarrow \varphi$, $U(\omega)=0$, $S(\omega)=0$, $\rho_\omega = 1/|\Omega|$, for all $\omega \in \Omega$
  $\Psi = \emptyset$ {Set of non-improving solutions accepted to diversify the search}
  while Stopping criterion is not met do
    Choose a destroy/repair operator $\omega \in \Omega$ using the roulette wheel selection mechanism based on current weights $\rho_\omega$
    while Three consecutive non-improving solutions are not found do
      Set $U(\omega) \leftarrow U(\omega)+1$
      Generate $(P',\varphi')$ from $(P,\varphi)$ by applying operator $\omega$.
      if $Makespan(P',\varphi') < Makespan(P,\varphi)$ then
        Set $P \leftarrow P'$, $\varphi \leftarrow \varphi'$, $S(\omega) \leftarrow S(\omega)+1$
      else
        if $Makespan(P',\varphi') \leq 1.10 \times Makespan(P^*,\varphi^*)$ then
          Generate a number $\theta \in [0,50]$ according to a discrete uniform distribution
            if $|\Psi| < \theta < 50$ then
              Set $P \leftarrow P'$, $\varphi \leftarrow \varphi'$, $\Psi \leftarrow \Psi \cup (P',\varphi')$
            end if
        end if
      end if
      if $Makespan(P',\varphi') < Makespan(P^*,\varphi^*)$ then
        Update the incumbent solution, set $P^* \leftarrow P'$ and $\varphi^* \leftarrow \varphi'$
      end if
    end while
    if the end of the search segment is reached then
      $\rho_\omega = 1/|\Omega|$
    else
      $\rho_\omega = S(\omega)/U(\omega)$
    end if
  end while
return $(P^*,\varphi^*)$
```

o_1. *Complete reordering of refill nodes*: The order in which the replenishment vehicle refills the painting vehicles is highly important for route synchronization and has a direct effect on the makespan. The role of this operator is to change the positions at which the refill nodes of the longest route are visited by the replenishment vehicle. For example, let r^* be the index of the painting vehicle with the longest route and $r_1^*, r_2^*, r_3^* \in V$ be the refill nodes on r^*. Suppose that the route followed by the replenishment vehicle is $\varphi = \{0,a,b,r_1^*,c,d,e,r_2^*,f,g,h,i,r_3^*,\ldots,0\}$, where $a,b,c,d,e,f,g,h,i \in V$, and 0 is the depot. Observe that the first refill (r_1^*) cannot be carried out before the second refill (r_2^*) takes place. In a similar way, the third refill (r_3^*) cannot be performed if the second one has not been performed. A new position for refill nodes r_1^*, r_2^*, and r_3^* in φ is randomly chosen while respecting these precedence relations. After this operator has been applied, the route of the replenishment vehicle as well as the times related to the painting vehicles are updated.

o_2. *Random reordering of refill nodes*: This operator is similar to the previous one. The only difference is that each refill node of the longest route has a probability of 50% of changing its position in the route followed by the replenishment vehicle.

o_3. *Double-random reordering of refill nodes*: This operator modifies the route of the replenishment vehicle. A painting route is randomly chosen and the refill nodes associated to this route then randomly change their positions in the route followed by the replenishment vehicle.

o_4. *Best arcs sequence removal-best insertion*: The goal of this operator is to reduce the ending time of the longest route. A sequence of arcs is removed from the longest route and the serviced arcs in the sequence are iteratively inserted in other routes. The cardinality of this sequence must be at most 40% of the number of arcs between the first and the last serviced arcs in the route. For each serviced arc in the sequence the algorithm performs the best reinsertion in every other route. A new route for the replenishment vehicle is then computed.

o_5. *Best interchange of arc status* (*serviced-traversed*): This destroy/repair operator attempts to intensify the search. An arc (i,j) serviced by the longest route is randomly chosen. If another route uses this arc just for traversal, the statuses of this arc in the two routes are interchanged. The routes are re-optimized by computing shortest paths affected by the status change of arc (i,j). This may lead to the computation of a new route for the replenishment vehicle.

o_6. *Random interchange of arc status* (*serviced-traversed*): This operator attempts to diversify the search. A required arc (i,j) is randomly selected and the route servicing this arc is identified. If there is another route traversing this arc, then an interchange of statuses for arc (i,j) is carried out in both routes.

o_7. *Reconstruction of replenishment vehicle route*: This operator generates a new route for the replenishment vehicle based on the current routes of the painting vehicles. The new refill nodes used to construct the route of the replenishment vehicle are those for which the painting vehicle does not have enough paint to continue its job. This selection of potential refill nodes is achieved by scanning the route of each painting vehicle forward and backward. Once the potential refill nodes have been identified, Algorithm 3 is applied to generate the new route for the replenishment vehicle.

5. Computational results

The algorithm just described was coded in C++ and compiled on a 2.40 GHz Intel(R) Xeon(R) CPU E5530 under the Linux operating system. The experimental work was carried out over a large set of artificial instances.

We have performed several analyses on the artificial instances in order to study: (i) the impact on computation time and solution quality yielded by the number of initial solution sets created in the construction phase (Section 5.1.1); (ii) the computation time used in each phase of our ALNS procedure (Section 5.1.2); (iii) the performance of the improvement phase over the initial solutions (Section 5.1.3); (iv) the performance of each replenishment policy D, R, and RD (Section 5.1.4); (v) the consumed time per activity of the painting vehicles (Section 5.1.5); (vi) the effect of the depot location on the makespan value (Section 5.1.6); and (vii) the impact on computation time and solution quality yielded by variations in the percentage of required arcs (Section 5.1.7).

The effect of the location of the depot was analyzed by considering three different locations: random location, best location (min), and remote location (max). We have solved a 1-median problem on G to determine the best location of the depot, and for the remote location, we have maximized the 1-median objective. In practice, the percentage of required arcs is related to the use of epoxy paint or water paint for road marking.

5.1. *Computational results on randomly generated instances*

We have first tested our ALNS on several sets of randomly generated instances. We have generated three instance sets S60, S80, and S100, with 60, 80, and 100 nodes, respectively. Each set is made up of 20 instances with around 200, 280, and 350 arcs, respectively. In addition, we have considered two larger sets with 200 and 400 nodes (S200 and S400) and for each of them 10 instances with around 720 and 1500 arcs were generated. All instances were generated on a planar graph in the $[0,5000] \times [0,5000]$ square. Specifically, we have used the generator developed by Ríos-Mercado and Fernández [14] to generate planar graphs which initially have undirected connections between nodes. However, since the SANRP is defined on a directed graph, all edges in the planar graphs were replaced by two opposite arcs, and the length of each arc (i,j) was computed as the Euclidian distance between i and j.

On the basis of real data from the Quebec Ministry of Transportation, we have used traveling and servicing speeds of 80 and 10 km/h, respectively, as well as a constant refill set-up time of five minutes and a speed of refill equal to 37.5 l/min. A liter of paint is enough to cover 2.3 m^2, and the width of the lines is set to 15 cm. The tank capacity for the painting vehicle is 1500 l, and the replenishment vehicle has enough capacity to satisfy the requirements of all painting vehicles. For the instance sets S60, S80, and S100, we have considered three, four, and five painting vehicles, respectively. For the sets S200 and S400, we have used six and seven painting vehicles, respectively. The 80 instances were solved under the three replenishment policies D, R, and RD, and by using three different locations of the depot. Additionally, we have solved the 80 instances under the three policies by considering a random location of the depot and three different percentages of required arcs, 40%, 70%, and 100%, respectively. Our ALNS metaheuristic was capable of solving all instances within reasonable computation times. All tables report average values over all instances of each set.

5.1.1. *Number of initial solutions*

As explained in Section 4.1, the construction procedure generates several initial solutions which are iteratively used as an input for the improvement phase of the ALNS metaheuristic. The first issue we have investigated was therefore to determine how many times the construction procedure has to be called in order to generate good solutions without incurring excessive computation time. Let $I_s \in \{5,10,15,20,25\}$ be the number of times the construction phase is called. Then for each I_s we have solved 80 instances by considering that all arcs are required and the depot has a random location. Table 1 displays the average time computed for each instance set, as well as the average value of the makespan. Observe that the computation time is proportional to I_s. However, solution quality does not always improve with I_s. The best compromise we have identified between solution quality and computation time is achieved for $I_s = 20$. We have therefore used this value in all subsequent tests. Recall that each time the construction phase is called, six different trial solutions are generated by using $\alpha \in \{0,0.2,\ldots,1.0\}$. The best of these six solutions is used as an input for the improvement phase. This means that in total 120 initial solutions are generated and 20 of them are filtered to the improvement phase.

5.1.2. *Computation time*

The computation time used by our ALNS is detailed in Table 2. Again, the depot is randomly located and all arcs are required. The second column displays the average time used to generate 120 initial solutions and the third column shows the average time needed to improve the 20 best initial solutions. The average total time of our ALNS metaheuristic is given in the fourth column. Observe that for all tested instances the improvement phase is the most time consuming one.

5.1.3. *Improvement phase performance*

To evaluate the performance of our destroy/repair operators we compute the average improvement of the makespan after the destroy/repair operators have been applied to the initial solutions. For each instance in the different tested sets, 20 initial solutions are used and the improvement phase is applied to each of them. The percentage of improvement from each initial solution is computed by 100 [Makespan (Initial) − Makespan (Final)]/Makespan (Initial). Hence, the percentage of improvement for a given instance is the average of 20 improvements and the improvement of each instance set is the average of improvements from all instances in the set. Table 3 shows the average improvement

Table 1
Impact of the number I_s of initial solutions for the painting vehicles under policy R. The depot is randomly located and all arcs are required.

Measure	$\|V\|$	Number of initial solutions				
		$I_s=5$	$I_s=10$	$I_s=15$	$I_s=20$	$I_s=25$
Computation time (s)	60	5.3	11.5	17.7	25.1	31.7
	80	10.1	21.4	33.2	47.3	59.7
	100	20.6	41.0	61.7	87.3	107.7
	200	138.1	340.3	497.3	663.8	822.3
	400	1125.8	2981.4	4315.7	5763.4	7222.3
Makespan	60	1183.9	1168.6	1156.9	1142.4	1132.2
	80	1380.9	1351.4	1332.5	1330.1	1330.1
	100	1573.8	1541.8	1536.8	1529.5	1528.6
	200	3401.4	3365.6	3346.8	3343.0	3335.9
	400	7026.9	6927.9	6904.1	6868.6	6853.7

Table 2
Computation times. The depot is randomly located and all arcs are required.

$\|V\|$	Computation time (s)		
	Construction	Improvement	Total
60	2.8	22.3	25.1
80	7.9	39.4	47.3
100	14.7	72.6	87.3
200	235.9	427.9	663.8
400	2690.7	3072.7	5763.4

Table 3
Average percentage of improvement. The depot is randomly located and all arcs are required. Policy RD.

$\|V\|$	Improvement (%)
60	17.5
80	28.5
100	31.4
200	25.9
400	22.0

obtained for different instance sets. We observe that the minimum percentage of makespan improvement is 17.5% for the smallest instances. In the case of the 100-node instances, the percentage of improvement reaches 31.4%. These results show that the proposed destroy/repair operators are very useful.

5.1.4. *Replenishment policies*

To analyze and compare the three replenishment policies, we have solved all instances with a random depot location and we have assumed that all arcs are required. Table 4 summarizes the results obtained in this test. The first column displays the size of the instance sets, and columns 2–4 give the average makespan obtained under each replenishment policy. Observe that policy RD, which combines refills by the replenishment vehicle and returns to the depot, is the best policy in all instances tested. The fifth and sixth columns contain the number of times replenishment policy R or RD that yield better makespan values than policy D which is common in practice. Observe that when $|V|=60$, replenishment policy R yields better results than policy D in 12 out of 20 instances; this is also true in four out of the 20 instances with $|V|=80$. However, for larger instances, policy R never yields better solutions than policy D. In contrast, policy RD generates better solutions than policies R and D on all instances tested except, for four instances with $|V|=60$ where replenishment policy R is better than RD. The last three columns of the table give the percentage of improvement between pairs of replenishment policies. Observe that on an average, the makespan is reduced from 11.6% to 25.7% if replenishment policy RD is adopted instead of D (RD vs D), and the makespan reduction lies between 3.1% and 43.2% if replenishment policy RD is adopted instead of R (RD vs R). In other words, it is clearly better to adopt a mixed replenishment policy.

5.1.5. *Time consumed per activity of the painting vehicles*

To evaluate the element of synchronization presented in this problem, we analyze how the painting vehicles use their time on their routes. In Table 5 we analyze the three replenishment policies R, D, and RD. Because the painting vehicles perform the same route under each of the three policies, the painting and deadheading times are also the same (see the second and third columns). Note that the waiting time takes place when a refill is performed. In policy D (fourth column) it represents the time in

Table 4
Comparison of the three replenishment policies. The depot is randomly located and all arcs are required.

$\|V\|$	Makespan			Absolute frequency			Improvement (%)		
	D	R	RD	R < D	RD < D	RD < R	R vs D	RD vs D	RD vs R
60	1142.4	1320.4	1107.4	19	20	18	12.6	15.5	3.1
80	1330.1	1237.3	1087.9	1	20	20	−8.0	11.6	18.1
100	1529.5	1335.7	1166.0	2	20	20	−15.0	12.7	23.6
200	3343.0	2656.7	2129.5	0	10	10	−25.8	19.9	36.1
400	6868.6	5249.2	3901.1	0	10	10	−31.0	25.7	43.2

Table 5
Time (min) consumed per activity of the painting vehicles. The depot is randomly located and all arcs are required.

$\|V\|$	All policies		Waiting time			Returning to D	
	Painting	Deadheading	D	R	RD	D	RD
60	2176.9	185.8	184.0	202.8	21.6	408.1	76.8
80	2808.9	233.4	347.2	1137.6	72.5	283.1	91.5
100	3484.4	331.7	702.8	2134.8	161.3	310.5	123.5
200	7153.5	715.4	2409.5	7974.1	314.6	1173.1	486.4
400	14 536.1	1444.7	7783.2	21 831.3	850.3	2744.7	1180.8

Table 6
Comparison of depot locations. All arcs are required under policy RD.

$\|V\|$	Makespan		Increment (%)
	Min	Max	
60	1092.5	1157.3	5.7
80	1076.4	1176.9	9.9
100	1145.4	1234.3	8.5
200	2061.4	2205.6	7.3
400	3800.1	4028.3	6.1

the queue at the depot and in policy R (fifth column) it represents the time that the painting vehicle waits for the replenishment vehicle. In policy RD, it is the sum of these two waiting times. Finally, under policies D and RD, time is also wasted when a painting vehicle goes to the depot to get a refill and returns to resume painting; these values are provided in columns 7 and 8. Observe that most of the time, the vehicles are performing a painting task and the deadheading is less than the waiting time. These data again suggest that the best policy is RD.

5.1.6. *Depot location*

In this part of our experimental work we have analyzed the effect of the depot location on the makespan value. The 80 instances with all arcs required were solved with three depot locations: random location, 1-median location (min), and remote location (max). We compute the percentage of increment when the depot is remotely located (max) rather than in the middle of the network (min). In Table 6, the second and third columns display the average makespan on each instance set, observe that the best makespan values are obtained when the depot is well located. The last column shows that the increment of the makespan lies from 5.7% to 9.9% when the depot is remotely located instead of in the middle of the network.

5.1.7. *Percentage of required arcs*

Finally, we have analyzed the effect of the percentage of required arcs on the computation time and on the makespan.

Table 7
Effect of the percentage of required arcs.

$\|V\|$	40%		70%		100%	
	Time (s)	Makespan	Time (s)	Makespan	Time (s)	Makespan
60	6.5	470.0	13.6	812.5	25.1	1107.4
80	10.7	484.1	24.7	820.3	47.3	1087.9
100	16.4	468.9	40.4	816.0	87.3	1166.0
200	98.4	896.8	283.4	1525.2	663.8	2129.5
400	783.1	1688.7	2413.2	2923.4	5763.4	3901.1

This part of our experimental work was inspired by the case where epoxy paint is used for road marking and therefore only some of the arcs have to be marked. Additionally, we have analyzed the case where the road marking is performed with water paint which is less durable than epoxy paint, and therefore all arcs have to be marked. For all instance sets we have considered that 40%, 70%, and 100% of the arcs are required. Table 7 shows that computation time increases with the percentage of required arcs and, as expected, the makespan is greater when all arcs are required. Note that these results are based on the replenishment policy RD which is the best policy we have identified.

6. Conclusions

We have introduced the SANRP (synchronized arc and node routing problem) which has not been previously addressed in the operations research literature. We have presented the problem in the context of road marking, but there exist other applications of this problem. We have developed an ALNS metaheuristic combining seven destroy/repair operators. The performance of our heuristic was evaluated over a large set of artificial instances. Three replenishment policies were compared. We have shown that the policy RD which allows replenishments from the depot and from the replenishment vehicle is the best. The introduction of the SANRP opens new challenges in the field of combined arc and node routing, namely in what concerns the computation of lower bounds and the development of exact algorithms.

Acknowledgements

This work was supported by CONACYT under the program of postdoctoral training abroad 2011–2012 and by FQRNT (Fonds Québécois de la recherche sur la nature et les technologies) under the program of fellowships for foreign students, and by NSERC under grants RGPIN41978 and 39682-10. This support is gratefully acknowledged. Thanks are due to the referees for their valuable comments.

References

[1] Amaya C, Langevin A, Trépanier M. The capacitated arc routing problem with refill points. Operations Research Letters 2007;35:45–53.
[2] Amaya C, Langevin A, Trépanier M. A heuristic method for the capacitated arc routing problem with refill points and multiple loads. Journal of the Operational Research Society 2010;61:1095–103.
[3] Bartodziej P, Derigs U, Malcherek D, Vogel U. Models and algorithms for solving combined vehicle and crew scheduling problems with rest constraints: an application to road feeder service planning in air cargo transportation. OR Spectrum 2009;31:405–29.
[4] Bouliane J, Laporte G. Locating postal relay boxes using a set covering algorithm. American Journal of Mathematical and Management Sciences 1992;12:65–74.
[5] Coelho LC, Cordeau J-F, Laporte G. The inventory-routing problem with transshipment. Computers and Operations Research 2012;39:2537–48.
[6] Golden BL, Wong RT. Capacitated arc routing problems. Networks 1981;11: 305–15.
[7] Hewitt M, Nemhauser GL, Savelsbergh MWP. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. INFORMS Journal on Computing 2010;22:314–25.
[8] Laporte G, Musmanno R, Vocaturo F. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. Transportation Science 2010;44:125–35.
[9] Lei H, Laporte G, Guo B. Districting for routing with stochastic customers. EURO Journal on Transportation and Logistics 2012;1:67–85.
[10] Pepin A-S, Desaulniers G, Hertz A, Huisman D. A comparison of five heuristics for the multiple depot vehicle scheduling problem. Journal of Scheduling 2009;12:17–30.
[11] Pia AD, Filippi C. A variable neighborhood descent algorithm for a real waste collection problem with mobile depots. International Transactions in Operational Research 2006;13:125–41.
[12] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. Computers and Operations Research 2007;34:2403–35.
[13] Resende MGC, Ribeiro CC. Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G, editors. Handbook of Metaheuristics. International Series in Operations Research and Management Science, vol. 57. New York: Springer; 2003. p. 219–49.
[14] Ríos-Mercado RZ, Fernández E. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. Computers and Operations Research 2009;36:755–76.
[15] Ropke S, Pisinger D. An adaptive large neighbourhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 2006;40:455–72.
[16] Rosa BD, Improta G, Ghiani G, Musmanno R. The arc routing and scheduling problem with transshipment. Transportation Science 2002;36:301–13.
[17] Salazar-Aguilar MA, Langevin A, Laporte G. Synchronized arc routing problem for snow plowing operations. Computers and Operations Research 2012;39: 1432–40.