Journal of Global Optimization (2021) 79:989–1006
https://doi.org/10.1007/s10898-020-00965-1

# The EMS vehicle patient transportation problem during a demand surge

**Farshad Majzoubi[1] · Lihui Bai[2] · Sunderesh S. Heragu[3]**

Received: 20 March 2019 / Accepted: 31 October 2020 / Published online: 4 January 2021


**Abstract**
We consider a real-time emergency medical service (EMS) vehicle patient transportation problem in which vehicles are assigned to patients so they can be transported to hospitals during an emergency. The objective is to minimize the total travel time of all vehicles while satisfying two types of time window constraints. The first requires each EMS vehicle to arrive at a patient's location within a specified time window. The second requires the vehicle to arrive at the designated hospital within another time window. We allow an EMS vehicle to serve up to two patients instead of just one. The problem is shown to be NP-complete. We, therefore, develop a simulated annealing (SA) heuristic for efficient solution in real-time. A column generation algorithm is developed for determining a tight lower bound. Numerical results show that the proposed SA heuristic provides high-quality solutions in much less CPU time, when compared to the general-purpose solver. Therefore, it is suitable for implementation in a real-time decision support system, which is available via a web portal (www.rtdss.org).

**Keywords** Vehicle routing · Meta-heuristics

## 1 Introduction

This paper considers dispatching EMS vehicles in emergencies, e.g., a large scale pandemic viral attack such as COVID-19, when the demand for vehicles is high. It is developed in a way that makes it suitable for use during demand surge as well as during normal operations.

✉ Lihui Bai
lihui.bai@louisville.edu
https://engineering.louisville.edu/faculty/lihui-bai/

Farshad Majzoubi
fmajzoubi@gmail.com

Sunderesh S. Heragu
sunderesh.heragu@okstate.edu

[1] Lyft Inc., San Francisco, USA

[2] Industrial Engineering Department, University of Louisville, Louisville, USA

[3] Industrial Engineering and Management Department, Oklahoma State University, Stillwater, USA

As suggested in [1–3] and as is current practice, EMS vehicles may need to serve up to two patients in these situations.

The EMS vehicle patient transportation problem (EMSVPTP) we consider assigns vehicles to patients so they can be transported to hospitals they choose or those selected by EMS personnel. During normal operations, patients may choose hospitals based on their medical insurance considerations. During a medical surge, hospitals may be chosen by EMS personnel based on availability, wait time, drive time and patient condition. Each EMS vehicle can serve up to two patients. The objective is to minimize the total travel time among all vehicles subject to two time-windows—the time window within which a vehicle must arrive at a patient's location and another time window within which the patient must be transported to the hospital. Two reasons have led to the use of total travel time as the objective function. First, an industry advisory board recommended that models for emergency response should not only be designed for use during a demand surge, but also during normal operations. This will ensure dispatchers trained to use a tool during normal operations can use it effectively when a demand surge occurs due to a pandemic. Second, minimizing total travel distance is a commonly used objective in software adopted by EMS departments in the U.S. and Canada, as they seek to minimize their overall cost while incorporating response time compliance as constraints. The proposed model has been implemented in a real-time decision support system, which is available on a web portal (www.rtdss.org).

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 introduces the EMSVPTP and describes its computational complexity. Section 4 presents a mixed-integer programming (MIP) model of the EMSVPTP. A column generation method to obtain a lower bound for the EMSVPTP is presented in Sect. 5. Next, a simulated annealing heuristic method for the EMSVPTP is presented in Sect. 6. Numerical experiments and computational results with CPLEX [4] are provided in Sect. 7. Conclusions are drawn in Sect. 8.

## 2 Literature review

The problem of dispatching EMS vehicles has been considered in the literature from two main aspects: the dispatching decision to serve current patient requests; and coverage decision in anticipation of serving future patient requests. An overview of these papers can be found in [5–7].

Yang et al. [8] develop an integrated approach for dispatching EMS vehicles, police vehicles, and fire trucks. Their model can be used in real-time to decide a subsequent destination for each vehicle. The objective of their model is to minimize travel times. Penalties are incurred for not reaching a patient in the desired time window and not covering demand nodes. Other researchers have also considered the coverage and dispatching decisions jointly. For example, Andersson and Varbrand [9] study an EMS vehicle dispatching and dynamic relocation problem. Ibri et al. [10] study an integrated dispatch and coverage problem model and develop a method combining ant colony and tabu search techniques for its solution. Majzoubi et al. [11] propose a mathematical model to dispatch and relocate these vehicles when the demand for the EMS vehicles is high. In the current paper, we only deal with dispatching part of the problem and the focus is on utilizing the capacity of EMS vehicles during a demand surge.

Mathematically, the EMS dispatching problem can be considered as a real-time vehicle routing problem (VRP) which is addressed abundantly in the literature. Because each patient

has pickup and drop-off points, it has characteristics that are similar to the VRP with pickup and delivery (VRPPD). Gendreau et al. [12] propose a tabu search heuristic for the VRPPD and use parallel computing for real-time fleet dispatch whenever there is a new request. Bent and Hentenryck [13] develop a hybrid algorithm for the VRPPD. They use simulated annealing for minimizing the number of vehicles and the travel cost. Further, Ropke and Pisinger [14] propose an extended large neighborhood search algorithm for the VRPPD.

Another specific type of VRP that is related to the EMSVPTP is the dial-a-ride (DAR) problem which has also been studied in the past. A survey of models and algorithms for the DAR problem is provided in Cordeau and Laporte [15]. A more recent survey of the DAR literature is provided in Ho et al. [16]. Using the DAR framework, Beaudry et al. [17] consider a problem for transporting patients on a hospital campus. Their problem considers the dynamic arrivals of new transportation requests. The objective in [17] is to minimize a weighted sum of three criteria: total travel time, lateness, and earliness. The authors use insertion movements and tabu search to develop a two-phase heuristic procedure in solving their particular DAR problem. Kergosien et al. [18] consider the problem of transporting patients between care units. They use a tabu search method similar to the method used in [12]. Parragh [19] develops two mathematical models for the DAR problem similar to those in [15] and proposes a branch-and-cut algorithm with a variable neighborhood search to solve the problem. Hu and Chang [20] propose a revised branch-and-price for the DAR problem, while Liu et al. [21] present a branch-and-cut algorithm for the problem. Markovic et al. [22] develop a Mobile Resource Management System for the DAR problem, designing a customized heuristic for dynamic dispatching. They consider a case study in Maryland and show their proposed solution can reduce the expenses for that case by 18%.

## 3 Computational complexity

Krumke et al. [23] study a general vehicle dispatching problem with at most two requests (VDP2), which simply dispatches vehicles so that the total transportation cost is minimized. Note that unlike a $k$-customer VRP (e.g., Haimovich et al. [24]), all vehicles in VDP2 are geographically dispersed and thus are not necessarily based at one depot. Furthermore, Krumke et al. [23] show that the VDP2 is NP-complete. In analyzing the computational complexity for the EMSVPTP, we demonstrate that essentially VDP2 is reducible to the EMSVPTP in polynomial time.

First, we formally introduce the VDP2 in [23]. We then show that a slight variation of VDP2, namely, a vehicle dispatching problem with two requests and a common destination depot (VDP2CDD), is NP-complete. Finally, we establish the proof that the EMSVPTP is NP-complete through a reduction from VDP2CDD.

**Vehicle dispatching problem with two requests (VDP2)**: Given a set of requests $R$, a set of vehicles $U$ ($|U| \leq 2|R|$), cost function $c: (U \cup R) \times (U \cup R) \mapsto R^+$, the vehicle dispatching problem with two requests (VPD2) finds a tour $t_u = (u, r_{u,1}, r_{u,l(u)})$ for each vehicle $u \in U$ that serves $l(u) \leq 2$ requests, such that each request is served in exactly one tour and the total cost of all tours is minimized.

**Vehicle dispatching problem with two requests and a common destination depot (VDP2CDD)**: Given a set of requests $R$, a set of vehicles $U$ ($|U| \leq 2|R|$), a common destination depot $d$, cost function $c: (U \cup R) \times (U \cup R \cup \{d\}) \mapsto R^+$, the vehicle dispatching problem with two requests and a common destination depot (VDP2CDD)

finds a tour $t_u = (u, r_{u,1}, r_{u,l(u)}, d)$ for each vehicle $u \in U$ that serves $l(u) \leq 2$ requests, such that each request is served in exactly one tour, all tours end at $d$, and the total cost of all tours is minimized.

**Lemma 1** *The VDP2CDD is NP-hard.*

***Proof*** Consider any VDP2 instance $(R, U, c)$. Construct a VDP2CDD instance $(R, U, c, d)$, where $c(r, d) = a \neq 0$ for all $r \in R$ and $a$ is a constant. As described, the construction of VDP2CDD can be done is polynomial time. In addition, any optimal solution to VDP2 solves the above constructed VDP2CDD and vice versa. In Krumke et al. [23], it is shown that the decision version of the VDP2 is NP-complete, thus VDP2 is NP-hard. Therefore, VDP2CDD is also NP-hard.

To prove EMSVPTP's complexity, we restate the EMSVPTP as follows. □

**EMS Vehicle Patient Transportation Problem (EMSVPTP)**: Consider a set of patients $P$, a set of their corresponding requested hospitals $H = \{h(p)|\forall p \in P\}$ ($|H| = |P|$), and a set of vehicles $V$ ($|V| \leq 2|P|$) each with a capacity of $k$ patients. Define $N = P \cup H$ and the cost function $s : (V \cup N) \times (V \cup N) \mapsto R^+$. Then, the EMS vehicle patient transportation problem, denoted as EMSVPTP($V$, $P$, $H$, $s$), finds a tour $t_v = (v, n_{v,i_1}, n_{v,i_2}, \ldots, n_{v,i_{2l(v)}})$ for each vehicle $v \in V$ that serves at most $l(v) \leq k$ patient requests, i.e., picks up each patient and drops each patient off at the requested hospital, such that each patient is served in exactly one tour and the total cost of all tours is minimized.

Note that in the above definition for EMSVPTP, $(i_1, i_2, \ldots, i_{2l(v)})$ is a permutation for $\{1, 2, \ldots, 2l(v)\}$. Further, if patient $p_0$ is the $i_t$th stop for vehicle $v$ in its tour and its associated hospital $h_0 = h(p_0)$ is the $i_q$th stop for the vehicle, i.e., $p_0 = n_{v,i_t}$, $h_0 = n_{v,i_q}$, then $i_q > i_t$.

Theorem 1 establishes the NP-hardness result for the EMSVPTP with $k = 2$ from the reduction of the VDP2CDD problem in Lemma 1 in polynomial time.

**Theorem 1** *The EMS patient transportation problem with $k = 2$ is NP-hard.*

***Proof*** Consider any instance of VDP2CDD $(R, U, c, d)$. To construct an equivalent EMSVPTP with $k = 2$, let $V = U$, $P = R$, $s = c$, and $H = \{d, d, \ldots, d\}$, i.e., all the destination hospitals are at the same location. Clearly, the EMSVPTP($V$, $P$, $H$, $s$) problem can be constructed in polynomial time.

Let $t^* = \{t_u^*\}_{u \in U}$ be an optimal solution to VDP2CDD. Then, for each vehicle $u$, the optimal tour serves either one or two requests.

(i) If $u \in U$ serves one request , i.e., $t_u^* = (u, r_{u,1}^*, d)$, then the corresponding solution to the EMSVPTP with $k = 2$ is $w_v^* = (v, p_{v,1}^* = r_{u,1}^*, h = d)$.
(ii) If $u \in U$ serves two requests , i.e. $t_u^* = (u, r_{u,1}^*, r_{u,2}^*, d)$, the corresponding solution to the EMSVPTP with $k = 2$ is $w_v^* = (v, p_{v,1}^*, p_{v,2}^*, h = d)$.

Note that (ii) is true because both requests have a common destination. We show that $\{w_v^*\}_{v \in V}$ is optimal to the above-constructed EMSVPTP with $k = 2$ by contradiction. If there exists a solution $\bar{w} = \{\bar{w}_{v \in V}\}$ with total cost $\sum_v \bar{s}_v \leq \sum_v s_v^*$, then solution $\bar{w}$ will induce an alternative solution $\bar{t}$ using (i)–(ii) with lower cost for VDP2CDD. This contradicts that $t^*$ is an optimal solution to VDP2CDD. Thus, any optimal solution to VDP2CDD solves the above constructed EMSVPTP. Similarly, it can be shown that any optimal solution $w^*$ to

the EMSVPTP with $k = 2$ solves the VDP2CDD. Thus, the VDP2CDD is equivalent to the constructed EMSVPTP. From Lemma 1, EMSVPTP with $k = 2$ is NP-hard.

Due to the above complexity results for EMSVPTP, we focus on developing efficient heuristic methods for solving EMSVPTP in a real-time decision support system. Because the current paper deals with the EMSVPTP with $k = 2$ as a prototype problem, in subsequent sections we refer to the EMSVPTP with $k = 2$ simply as EMSVPTP without explicitly specifying $k = 2$. □

## 4 The mathematical formulation for the EMSVPTP

To formulate a mixed integer linear program for the EMSVPTP, we define the following sets and mapping functions:

| *Sets* | |
|---|---|
| $K$ | Set of vehicles |
| $K^H \subseteq K$ | Set of vehicles en route to hospitals |
| $P$ | Set of patients |
| $H$ | Set of hospitals |
| $O$ | Set of virtual depots for vehicles |
| *Mapping functions* | |
| $o(k) \in O$ | A function mapping vehicle $k$ to its virtual depot |
| $p(k) \in P$ | A function mapping vehicle $k$ to the patient currently on board, $\forall k \in K^H$ |

In the above notation, $O$ refers to a set of virtual depots for the vehicles. It is assumed that the current location of an EMS vehicle is that of its (virtual) depot. Based on the defined sets and mapping functions, and following the notation in the three-index formulation in [15], we define a network $G = \{N, A\}$, where $N = O \cup P \cup H \cup \{d\}$ is the set of all nodes including virtual depots, patients, hospitals, and a common virtual destination node $d$ for all vehicles. After the last patient is dropped off at a hospital, it is assumed that the EMS vehicle travels to the virtual destination node $d$, for which the travel time is assumed to be zero.

Assuming there are $m$ EMS vehicles and $n$ patients, we index all nodes in $N$ as follows:

$$P = \{1, 2, \ldots, n\},\ H = \{n+1, n+2, \ldots, 2n\},$$
$$O = \{2n+1, 2n+2, \ldots, 2n+m\},\ d = \{2n+m+1\}.$$

Note that there is a distinct hospital node corresponding to each patient node. In reality, two or more of the hospital nodes may correspond to the same hospital, but we formulate our model so that the transfer of each patient to a hospital has a unique arc associated with it, following the convention used in general VRPs with unique pick-up and drop-off points.

In the EMSVPTP, there are six possible vehicle movements at any given time as listed below.

1. From the virtual starting depot to a patient;
2. From first patient to a second patient;
3. From a patient to a hospital;
4. From a hospital to pick up a second patient;
5. From first patient's hospital to the second patient's hospital and vice versa;

6. From hospital to the common virtual destination.

Therefore, the set of valid arcs $A$ is defined as

$$\begin{aligned} A = \{(i,j) |\ & i \in O, j \in P \cup \{d\}; \\ & i \in P \cup H,\ j \in P \cup H, \text{ for } i \neq j \text{ and } i \neq n+j; \\ & i \in H, j = 2n+m+1\}, \end{aligned}$$

where the first set of $(i,j)$'s for $i \in O,\ j \in P$ represents the first type of EMS vehicle movement, the third set of $(i,j)$'s represents the sixth type of vehicle movement, and the second set represents the remaining four types of movements.

Let binary variable $x_{ij}^k$ (where $(i,j) \in A$ and $k \in K$) be 1 if vehicle $k$ goes from node $i$ to $j$; 0 otherwise. Also, let variable $y_{ik}$ represent the time that vehicle $k$ arrives at node $i$. Let $t_{ij}$ be the travel time on arc $(i,j) \in A$ and $\psi_i$ be the desired time by which node $i$ must be visited. Then, similar to [15], the EMSVPTP can be formulated as follows:

$$\min \quad \sum_{(i,j)\in A} \sum_{k\in K} t_{ij} x_{ij}^k \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in N} \sum_{k\in K} x_{ij}^k = 1, \qquad \forall i \in P \tag{2}$$

$$\sum_{i\in N} x_{2n+k,i}^k = \sum_{i\in N} x_{i,2n+m+1}^k = 1, \qquad \forall k \in K \tag{3}$$

$$\sum_{j\in N} x_{ij}^k - \sum_{j\in N} x_{j,n+i}^k = 0, \qquad \forall i \in P,\ \forall k \in K \tag{4}$$

$$\sum_{j\in N} x_{ji}^k - \sum_{j\in N} x_{ij}^k = 0, \qquad \forall i \in P \cup H,\ \forall k \in K \tag{5}$$

$$x_{2n+k,p(k)}^k = 1, \qquad \forall k \in K^H \tag{6}$$

$$\sum_{i\in N} \sum_{j\in P} x_{ij}^k \leq 2, \qquad \forall k \in K \tag{7}$$

$$y_{ik} + t_{i,i+n} \leq y_{i+n,k}, \qquad \forall i \in P,\ \forall k \in K \tag{8}$$

$$y_{jk} \geq y_{ik} + t_{ij} - M(1 - x_{ij}^k), \qquad \forall (i,j) \in A, \forall k \in K \tag{9}$$

$$y_{ik} \leq \psi_i, \qquad \forall i \in N, \forall k \in K \tag{10}$$

Recall that $x_{ij}^k$ is defined only on permissible arc $(i,j) \in A$ for vehicle $k$. Then, in the above formulation (EMSVPTP), the objective function (1) minimizes the total travel time for all the vehicles. Constraint (2) states that each patient must be served by a vehicle. Constraint (3) ensures each vehicle begins at the virtual depot and returns to the virtual destination node. Constraint (4) ensures the same vehicle will pick up a patient and transport them to the associated hospital. In other words, if vehicle $k$ serves patient $i$, vehicle $k$ has to transport that patient to hospital node $n+i$ as well. Constraint (5) is the flow balance equation at either a patient or a hospital node, ensuring the vehicle that comes into node $i$ must leave that node. Constraint (6) ensures that a vehicle en-route to a hospital with a patient is assigned to that patient. Of course, this vehicle may then serve one more patient, as specified by constraint (7). Finally, constraints (8)–(10) ensure the desired time window constraints are satisfied at both patient and hospital nodes.

The above model can be used in a real-time environment for an EMS department with a CAD (computer-aided dispatcher) and AVL (automatic vehicle locations). When latitude and

longitude of a vehicle are provided, the origin-destination matrix $t_{ij}$ can be readily calculated using real-time routing applications. To further reduce processing time, the time between street segment centroids can be pre-calculated and the vehicles can be snapped back to these segments when we receive the AVL data.

The mixed-integer program (1)–(10) is implemented using C# via CPLEX 12.10 [4]. Because the NP-hardness results from Sect. 3, it is not surprising that general-purpose solvers such as CPLEX take excessive amount of CPU time for even small-sized instances, as reported in detail in Sect. 7. Thus, the next section proposes an efficient heuristic method for obtaining a lower bound and a high-quality solution faster.

## 5 Column generation

In this section, we present a column generation approach for obtaining a tight lower bound for the EMSVPTP. We then exploit this method to solve the and find a good feasible solution. First, we present the basics of applying the column generation technique to the general VRP similar to many papers in the literature (e.g., Desrochers et al. [25], Xu et al. [26], Fillet [27], and Ropke and Cordeau [28]). Then we discuss a special pricing algorithm for solving the subproblem in the column generation method for the EMSVPTP.

In the literature, many researchers have used the column generation technique to solve the VRP through a reformulation as a set covering problem (e.g., Xu et al. [26]). When considered as a set covering problem, the VRP is to determine the set of routes with the minimum total travel cost given the entire set of all possible routes.

More formally, let $a_{iku} = 1$, if route $u$ includes a visit of node $i$ by vehicle $k$; 0 otherwise. Let $c_{ku}$ be the cost of route $u$ for vehicle $k$. In addition, let $\Omega$ be the set of all feasible routes and $\Omega_1 \subseteq \Omega$ be an arbitrary subset of $\Omega$. A route is feasible if it satisfies the required time window constraints. Suppose decision variable $\theta_{ku} = 1$ if route $u$ is served by vehicle $k$; 0 otherwise. Then, the VRP can be written as the set covering problem as follows:

$$\text{(VRP-SC)} \quad \min \quad \sum_{k\in V}\sum_{u\in\Omega} c_{ku}\theta_{ku} \tag{11}$$

$$\text{s.t.} \quad \sum_{k\in V}\sum_{u\in\Omega} a_{iku}\theta_{ku} \geq 1, \quad \forall i \in N \tag{12}$$

$$\sum_{u\in\Omega} \theta_{ku} \leq 1, \quad \forall k \in V \tag{13}$$

$$\theta_{ku} \in \{0, 1\} \tag{14}$$

The column generation approach begins with a subset of routes $\Omega_1 \subset \Omega$ to solve the following (restricted) master problem of (VRP-SC).

$$\text{(MP)} \quad \min \quad \sum_{k\in V}\sum_{u\in\Omega_1} c_{ku}\theta_{ku} \tag{15}$$

$$\text{s.t.} \quad \sum_{k\in V}\sum_{u\in\Omega_1} a_{iku}\theta_{ku} \geq 1, \quad \forall i \in N \tag{16}$$

$$\sum_{u\in\Omega_1} \theta_{ku} \leq 1, \quad \forall k \in V \tag{17}$$

$$\theta_{ku} \in \{0, 1\}, \quad \forall k \in V \tag{18}$$

Consider the linear relaxation of the master problem (MP). Let $\lambda_i$ and $\omega_k$ be the dual variables associated with constraints (16) and (17), respectively. We solve the subproblem of minimizing the reduced cost for all routes $u \in \Omega$. If the minimum reduced cost is zero, then the optimal route is found; otherwise, we update $\Omega_1$ by including the route with the minimum reduced cost and resolve the master problem (MP). Therefore, the subproblem is commonly referred to as the "pricing" problem. Rather compactly, the subproblem can be written as:

$$\text{(PP)}: \qquad \min_{u\in\Omega}\{-\sum_{i\in N} a_{iku}\lambda_i + \omega_k + c_{ku}\}$$

Let $b_{ijku} = 1$ if route $u$ uses vehicle $k$ to visit arc $(i, j)$ and 0 otherwise. Then, $a_{iku} = \sum_{j\in N} b_{ijku}$ and $c_{ku} = \sum_{(i,j)\in(N\cup O)\times N} b_{ijku}t_{ij} = \sum_{i,j\in N} b_{ijku}t_{ij} + \sum_{j\in N} b_{o(k)jku}t_{o(k)j}$. Thus, the objective function in (PP) can be rewritten as

$$c_{ku} - \sum_{i\in N} a_{iku}\lambda_i + \omega_k = \sum_{i,j\in N} b_{ijku}(t_{ij} - \lambda_i) + \sum_{j\in N} b_{o(k)jku}t_{o(k)j} + \omega_k \tag{19}$$

In other words, the travel cost between nodes is $(t_{i,j} - \lambda_i)$ for $i, j \in N$ and $(t_{o(k),j} + \omega_k)$ for $(o(k), j) \in O \times N$. Therefore, the subproblem (PP) can be considered as an elementary shortest path problem with resource constraints (ESPPRC) (see e.g., Desrochers et al. [25] and Ropke and Cordeau [28]), where the resource constraints correspond to the time window constraints in the underlying route selection.

Finally, we make two notes regarding the implementation of the pricing problem (PP). First, because each vehicle cannot serve more than two patients in our problem, the maximum length of a route (excluding depots) is four. Therefore, we simply enumerate all routes and add the best route to $\Omega_1$. Second, as detailed in Sect. 7, our implementation adds $M$ routes/columns (with the $M$ most negative reduced costs) at a time, where $M$ is a predefined parameter representing the maximum number of columns to add in the column generation algorithm. This process is continued until we are not able to find any negative cost routes.

## 6 A simulated annealing algorithm for the EMSVPTP

In this section, we present a meta-heuristic method for the EMSVPTP using the simulated annealing approach. We define a pairwise exchange type of neighborhood function/search in Subroutine 1, followed by Subroutine 2 which describes how we optimally sequence the nodes visited by a vehicle after the selections of nodes are determined for all vehicles. Finally, Subroutine 3 conducts a specialized local search attempting to introduce more diverse solutions at later stages (with lower temperatures) of the SA procedure. We refer to the combination of this local search with the pairwise exchange neighborhood search as the 'hybrid' method in this and subsequent sections.

In particular, Subroutine 1 performs a pairwise type of exchange to obtain a neighborhood solution from any given solution. It first randomly chooses two vehicles in the current solution and randomly selects one patient from each vehicle. The patients are then swapped between the two vehicles. Note that a randomly selected vehicle may not have any patient, thus there are three possible outcomes for the exchange/swap: 1) swapping two patients between the two chosen vehicles, each has at least one patient; 2) moving one patient from a vehicle with at least one patient to the other that has no patient; 3) doing nothing when neither vehicle has a patient. Subroutine 1 below excludes all swaps with outcome 3).

**Subroutine 1: The Pairwise Exchange Neighborhood Function**
$\bar{X} = \mathcal{N}(X, i, j)$

Step 0: (Initialize the neighbor solution)
$\bar{X} = X$;
Step 1: (Select the patients to swap from vehicles $i$ and $j$)
**For** $k = i, j$
if $k \in K^H$, then choose the only patient in vehicle $k$ as patient $p_k$;
if $k \in K \setminus K^H$, then randomly choose any patient in vehicle $k$ as patient $p_k$;
**EndFor**
Step 2: (Swap selected patients in vehicles $i$ and $j$)
**If** both $p_i$ and $p_j$ are non-null, then then swap them between two vehicles $i$ and $j$
**ElseIf** one of $p_i$ and $p_j$ is null, then simply move the non-null patient to the other vehicle
Step 3: (Optimize the routes in vehicles $i$ and $j$)
**For** $k = i, j$
if $k$ is idle or en-route to a patient find the best possible sequence from all possible six sequences(see Subroutine 2);
if $k$ is transporting a patient, find the best possible sequence from all possible three sequences (see Subroutine 2);
**EndFor**
return($\bar{X}$)

Subroutine 2 OptimizeSequence determines the best sequence of all nodes to be visited by vehicle $k$ when it serves two patients. In this case, at most six possible sequences exist because there are, at most, two patients and two hospitals to be visited by a vehicle, and each patient is visited before the associated hospital. In particular, let $1 \leq \rho \leq 6$ be the sequence index, $p_1$ and $p_2$ be the two patients and $h_1$ and $h_2$ be their respective hospitals. Then sequences 1 through 6 are shown in the table below. For example, in sequence 3, vehicle $k$ visits patient $p_1$ first, followed by that patient's hospital $h_1$, then patient $p_2$, and finally the hospital $h_2$ of patient 2.

| Sequence index $\rho$ | The actual sequence |
|---|---|
| 1 | $(k, p_1, p_2, h_1, h_2)$ |
| 2 | $(k, p_1, p_2, h_2, h_1)$ |
| 3 | $(k, p_1, h_1, p_2, h_2)$ |
| 4 | $(k, p_2, p_1, h_1, h_2)$ |
| 5 | $(k, p_2, p_1, h_2, h_1)$ |
| 6 | $(k, p_2, h_2, p_1, h_1)$ |

Using this notation, Subroutine 2 examines all applicable sequences $l$ through $u$ ($l \leq u$) for vehicle $k$ in serving patient $j$, with $X$ being the associated solution. If improvement is found, the subroutine updates $X$ with the improved sequence. Note that for vehicle $k \in K \setminus K^H$, all six sequences are feasible, hence $l = 1$ and $u = 6$. However, for vehicle $k \in K^H$, because only sequences 1 through 3 are feasible, $l = 1$ and $u = 3$.

**Subroutine 2: OptimizeSequence**($\bar{X},\ k,\ j,\ l,\ u$)
  **Loop** $\rho = l, \cdots, u$ // $l$ through $u$ are the applicable sequence indices
   Calculate times to reach the first and second patient $g^1(\rho)$ and $g^2(\rho)$;
   Calculate times to transport the first and second patient to their respective hospitals $g^3(\rho)$ and $g^4(\rho)$;
   **If** $g^l(\rho), l = 1, 2, 3, 4$ are all feasible, **Then**
    Evaluate the new objective value associated with sequence $\rho$, i.e., Obj($\rho$);
    **If** Obj($\rho$) < BestObj **Then**
     Assign patient $j$ to vehicle $k$ using sequence $\rho$ and update solution $\bar{X}$;
     BestX=$\bar{X}$;
     BestObj=Obj($\rho$);
  **EndLoop**
Return BestX

In addition to the pairwise exchange neighborhood search in Subroutine 1, we propose a nested vehicle search (NVS) algorithm to be used in later stages of the SA process when the temperature is relatively low. In essence, the NVS in Subroutine 3 attempts to diversify the solution space when the SA process begins to concentrate its search. In particular, beginning with the current incumbent solution in the SA process, the NVS first randomly select a patient ($j$) and cancels the assignment of patient $p_1$ to its current hospital $h_1$. Second, a local greedy search of vehicle is performed for this patient $j$. This search checks all vehicles with three possible scenarios. In the first scenario where vehicle $i$ is assigned to only one patient, step 1.2.1 checks if patient $j$ can be reached by vehicle $i$ within the specified time window. In addition, Subroutine 2 will be applied to optimize the sequence for this vehicle to visit two patients considering six alternative sequences. In the second scenario where vehicle $i$ is transporting a patient to a hospital, step 1.2.2 checks all three possible sequences when performing sequence optimization. This is because the current patient has to be the first patient, thus making $\rho = 4, 5$ and 6 infeasible in Subroutine 2. In the third scenario where a vehicle $i$ is assigned to serve two patients, let $j_1$ be the immediate patient and $j_2$ the second patient. Then step 1.2.3 checks if swapping current patient $j$ with the second patient $j_2$ can lead to a better solution. In addition, step 1.2.3 examines if $j_2$ can be served by either a vehicle which is assigned to a patient (step 1.2.1) or a vehicle en route to a hospital (step 1.2.2). If so, the swap will be accepted. After considering all three scenarios for patient $j$, the best assignment is obtained. This process repeats for all patients other than $j$ before a best local solution is identified.

**Subroutine 3: The Nested Vehicle Search Neighborhood Function**
$BestX = NVS(\bar{X})$

Step 0: // Initialize
Bestx=$\bar{X}$
Step 1: //Assign vehicles to patients selected by SA based on a greedy selection
**Loop** $j = p_1, p_2$ going through patients selected by SA
**If** ($j \neq$ null) **Then**
Step 1.1: Un-assign patient $j$ from its current vehicle
Step 1.2: Select a vehicle for patient $j$ using the following procedures
**Loop** $k = 1, \cdots, m$ //searching through all vehicles
Step 1.2.1: //first, examine vehicles en route to a patient and not yet being assigned to a second patient
**If** $k \in K \setminus K^H$ and it is assigned to only one patient **Then**
Obtain $\bar{X}$ resulted from assigning patient $j$ to vehicle $k$ using $OptimizeSequence(\bar{X}, k, j, 1, 6)$
**EndIf**
Step 1.2.2: // second, examine vehicles en route to a hospital
**If** $k \in K^H$ and no patient is assigned to it **Then**
Obtain $\bar{X}$ resulted from assigning patient $j$ to vehicle $k$ using $OptimizeSequence(\bar{X}, k, j, 1, 3)$
**EndIf**
Step 1.2.3: // third, examine vehicles that are assigned to two patients with different requested hospitals,
**If** $k \in K \setminus K^H$ and it is assigned to two patients with two different hospitals **Then**
Examine if the second patient of vehicle $k$ can be served by a different vehicle $k'$ using similar processes
as in Steps 1.2.1 and 1.2.2;
**If** a $k'$ is found **Then**
Obtain $\bar{X}$ resulted from assigning patient $j$ to vehicle $k$ using $OptimizeSequence(\bar{X}, k, j, 4, 6)$
**EndIf**
**EndIf**
**EndLoop** // going through vehicle $k$
Step 1.3:
**If** $obj \leq$BestObj **Then**
BestX=$\bar{X}$, BestObj=$Obj$;
**EndIf**
**EndIf**
**EndLoop** // going through patient $j$

## 7 Numerical experiments

To evaluate the proposed simulated annealing heuristic, we used C# to implement the SA heuristic, as well as the mixed-integer model for EMSVPTP via the ILOG/Concert technology by CPLEX 12.10 [4]. The CPU times reported here are from an Acer Intel(R) Core(TM) i7-6500 U CPU dual-core processor with 2.5 GHz and 2.6 GHz and 8 GB RAM on a 64-bit Windows operating system.

Preliminary pilot studies are conducted to fine-tune the parameters in the proposed SA heuristic. From the pilot runs (see Majzoubi [11]), the best parameter setting recommendations are as follows. The initial temperature $T_{init} = 20$ which equates to approximately 80% probability of accepting an uphill transition initially. The choice of the exponential cooling function is $T_k = T_0\alpha^k$ with $\alpha = 0.9$. The total epoch length and total length are set to

$L = 10{,}000$ and $L_{total} = 100{,}000$, respectively. For the column generation method, we set the maximum number of columns to add in one iteration to be $M = 800$.

In our numerical experiments, all test instances are randomly generated in a $50 \times 50$ mile grid. The location of patients and vehicles are chosen randomly using a uniform distribution within the grid.

To evaluate the efficacy of allowing up to two patients in a vehicle, we run 20 instances with the numbers of patients and vehicles varying from 15 to 25. We compare two dispatching strategies: (a) a traditional dispatching where the closest vehicle is dispatched to serve a single patient, and (b) the proposed method allowing an EMS vehicle to serve up to two patients. For the first strategy, the average time for a patient to arrive at their hospital is approximately 28.5 min whereas the same service time reduces to 19.6 min under the second strategy. This demonstrates the effectiveness of the proposed model. A more thorough analysis regarding the benefit of increasing EMS vehicle's capacity to two is done in [11].

To measure the quality of our proposed algorithm, we first compare the performance of the SA heuristic for EMSVPTP against that of the MIP model in Sect. 4 by CPLEX [4] on 20 test instances. These test instances are relatively small-sized, with the numbers of patients and vehicles ranging between 15 and 25. Multiple instances are created for each combination of patient and vehicle sizes. The results are compared and summarized in Table 1 which displays the numbers of patients and EMS vehicles for each group of instances in column 2; the objective function values and the computer processing unit (CPU) times for SA in columns 3 and 4; and the objective function value as well as CPU times for CPLEX in columns 5 and 6. Column 7 represents the percentage gap between SA and CPLEX. Also reported in Table 1 are the average objective function values and CPU times for SA and CPLEX.

From Table 1, it can be seen that for the small-sized EMSVPTP instances, the average CPU time for SA is 0.9 seconds compared to 118.38 seconds for CPLEX while the quality of the SA solution is rather high with an average gap of 0.09%. More specifically, in four of the 20 instances, the CPLEX solution is better than those provided by SA. On the other hand, for 16 examples, CPLEX and SA found the same optimal solution. In all of these instances, less CPU time is required for the SA than CPLEX. In particular, SA was 130 times faster than CPLEX, on average. Thus, we conclude that the proposed SA algorithm is efficient and robust, because for most instances it was able to provide the same solution as CPLEX in much less CPU time.

Next, we generated medium-sized EMSVPTP instances where the numbers of vehicles and patients range between 70 and 80, and between 55 and 95, respectively. These 50 instances are too large to be solved by CPLEX, therefore we compare the heuristic solution obtained using SA to the lower bound obtained using the column generation method described in Sect. 5.

In addition to the proposed algorithms, we compare our results with a greedy heuristic algorithm for bench-marking purposes. Subroutine 4 presents an outline of this algorithm.

**Table 1** Comparison between the SA and CPLEX solutions

| Instance | (Vehicles, patients) | SA method | | CPLEX | | Gap (%) |
|---|---|---|---|---|---|---|
| | | Objective function | CPU time (s) | Objective function | CPU time (s) | |
| 1 | | 808.54 | 1.02 | 808.54 | 7.63 | 0 |
| 2 | | 785.50 | 0.64 | 785.50 | 4.61 | 0 |
| 3 | (15, 15) | 777.32 | 0.65 | 777.32 | 9.65 | 0 |
| 4 | | 733.41 | 0.65 | 733.41 | 4.06 | 0 |
| 5 | | 734.76 | 0.65 | 713.87 | 6.94 | 0 |
| 6 | | 801.76 | 0.44 | 831.93 | 1.10 | 0 |
| 7 | (25, 5) | 701.01 | 0.46 | 701.01 | 0.75 | 0 |
| 8 | | 767.80 | 0.75 | 767.80 | 1.26 | 0 |
| 9 | | 913.04 | 0.56 | 913.04 | 1.79 | 0 |
| 10 | (25,10) | 847.58 | 0.64 | 847.58 | 1.08 | 0 |
| 11 | | 944.23 | 0.80 | 944.23 | 2.56 | 0 |
| 12 | | 898.14 | 1.03 | 898.14 | 7.26 | 0 |
| 13 | (25,15) | 1030.70 | 0.71 | 1030.70 | 10.78 | 0 |
| 14 | | 1000.25 | 0.66 | 1000.25 | 8.59 | 0 |
| 15 | | 1114.78 | 1.28 | 1113.73 | 19.61 | 0 |
| 16 | (25, 20) | 1059.30 | 1.26 | 1055.47 | 199.30 | 0.36 |
| 17 | | 1059.88 | 1.25 | 1059.88 | 85.76 | 0 |
| 18 | | 1183.19 | 1.48 | 1178.52 | 437.16 | 0.4 |
| 19 | (25,25) | 1109.18 | 1.49 | 1109.18 | 762.11 | 0 |
| 20 | | 1115.84 | 1.51 | 1105.48 | 795.66 | 0.94 |
| Average | | 919.78 | 0.90 | 918.78 | 118.38 | 0.09 |

**Subroutine 4: A greedy heuristic algorithm**
*//Initialization:*
Randomly assign each patient to a vehicle while satisfying vehicle's capacity;
BestObj=$\infty$;
*//Greedy search:*
**While** better solution found
**Loop** $j$ in set of patients
**Loop** $i$ in set of vehicles
**Repeat** for any patient (maximal of two) in vehicle $i$
1. swap patient $j$ with this patient in vehicle $i$
2. Let its objective value be NewObj
3. if the new solution is not feasible then penalize the NewObj with the time window constratints violated
4. if NewObj<BestObj, then update the best solution as BestX
**EndRepeat**
**EndLoop**
**EndLoop**
**EndWhile**
Return BestX

The results of these algorithms are provided in Table 2. All associated master and pricing problems for the set covering problem and all routes for the column generation algorithm

**Table 2** Comparison between the SA and the lower bound from column generation

| Instance | LB column generation | | SA | | | Greedy | | |
|---|---|---|---|---|---|---|---|---|
| | ObjFun | CPU time (s) | ObjFun | CPU time (s) | Gap (%) | ObjFun | CPU time (s) | Gap (%) |
| 1 | 2225.43 | 45.76 | 2284.49 | 0.7 | 2.65 | 2381.82 | 0.09 | 7.03 |
| 2 | 2204.68 | 70.69 | 2267.76 | 0.73 | 2.86 | 2426.89 | 0.03 | 10.08 |
| 3 | 2554.43 | 38.23 | 2582.49 | 0.71 | 1.10 | 2710.54 | 0.04 | 6.11 |
| 4 | 2568.34 | 12.71 | 2620.11 | 0.73 | 2.02 | 2664.86 | 0.07 | 3.76 |
| 5 | 2185.01 | 21.77 | 2312.26 | 0.78 | 5.82 | 2331.68 | 0.06 | 6.71 |
| 6 | 2851.62 | 15.15 | 2919.64 | 0.75 | 2.39 | 2986.26 | 0.05 | 4.72 |
| 7 | 2628.69 | 57.93 | 2701.64 | 0.79 | 2.78 | 2802.65 | 0.11 | 6.62 |
| 8 | 2792.14 | 20.37 | 2835.5 | 0.74 | 1.55 | 2890.29 | 0.08 | 3.52 |
| 9 | 2603.75 | 47.91 | 2665.71 | 0.77 | 2.38 | 2738.57 | 0.08 | 5.18 |
| 10 | 2701.36 | 26.78 | 2773.2 | 0.77 | 2.66 | 2861.3 | 0.17 | 5.92 |
| 11 | 3143.8 | 53.26 | 3154.99 | 0.8 | 0.36 | 3317.94 | 0.1 | 5.54 |
| 12 | 2987.25 | 26.47 | 3044.46 | 0.83 | 1.92 | 3109.44 | 0.16 | 4.09 |
| 13 | 2576.01 | 77.75 | 2640.2 | 0.8 | 2.49 | 2742.82 | 0.06 | 6.48 |
| 14 | 3068.42 | 34.61 | 3140.16 | 0.82 | 2.34 | 3244.96 | 0.09 | 5.75 |
| 15 | 2647.58 | 118.77 | 2725.54 | 0.83 | 2.94 | 2828.99 | 0.1 | 6.85 |
| 16 | 3186.92 | 60.23 | 3235.2 | 0.85 | 1.51 | 3359.06 | 0.1 | 5.40 |
| 17 | 2906.36 | 148.87 | 2963.38 | 0.88 | 1.96 | 3172.17 | 0.11 | 9.15 |
| 18 | 3379.72 | 86.92 | 3440 | 0.86 | 1.78 | 3647.1 | 0.1 | 7.91 |
| 19 | 2929.12 | 176.38 | 3070.7 | 0.88 | 4.83 | 3187.79 | 0.07 | 8.83 |
| 20 | 2960.34 | 139.35 | 3049.97 | 0.89 | 3.03 | 3119.43 | 0.07 | 5.37 |
| 21 | 3119.3 | 360.95 | 3201.16 | 0.95 | 2.62 | 3332.83 | 0.12 | 6.85 |
| 22 | 3048.19 | 689.85 | 3138.13 | 0.96 | 2.95 | 3375.76 | 0.13 | 10.75 |
| 23 | 3666.84 | 437.49 | 3684.11 | 0.91 | 0.47 | 3956.2 | 0.18 | 7.89 |
| 24 | 3374.45 | 292.76 | 3436.85 | 0.95 | 1.85 | 3692.37 | 0.13 | 9.42 |
| 25 | 3268.15 | 276.42 | 3373.73 | 0.98 | 3.23 | 3592.97 | 0.13 | 9.94 |
| 26 | 2417.99 | 28.79 | 2546.59 | 0.75 | 5.32 | 2511.59 | 0.05 | 3.87 |
| 27 | 2627.1 | 17.2 | 2671.19 | 0.73 | 1.68 | 2727.56 | 0.05 | 3.82 |
| 28 | 2365.88 | 20.08 | 2470.36 | 0.76 | 4.42 | 2556.42 | 0.05 | 8.05 |
| 29 | 2536.33 | 13.12 | 2629.17 | 0.75 | 3.66 | 2682.83 | 0.08 | 5.78 |
| 30 | 2740.9 | 7.67 | 2752.73 | 0.74 | 0.43 | 2795.88 | 0.08 | 2.01 |
| 31 | 2619.2 | 35.35 | 2721.4 | 0.82 | 3.90 | 2761.56 | 0.1 | 5.44 |
| 32 | 2784.25 | 27.1 | 2840.17 | 0.83 | 2.01 | 2869.84 | 0.09 | 3.07 |
| 33 | 2709.97 | 44.15 | 2771.54 | 0.83 | 2.27 | 2853.3 | 0.06 | 5.29 |
| 34 | 2873.35 | 20.5 | 2917.83 | 0.77 | 1.55 | 2966.19 | 0.1 | 3.23 |
| 35 | 2820.52 | 25.24 | 2891.54 | 0.8 | 2.52 | 2955.05 | 0.07 | 4.77 |
| 36 | 3054.78 | 88.25 | 3098.47 | 0.84 | 1.43 | 3209.24 | 0.07 | 5.06 |
| 37 | 2842.25 | 171.97 | 2931.74 | 0.85 | 3.15 | 3044.62 | 0.08 | 7.12 |
| 38 | 2928.83 | 52.64 | 2975.92 | 0.84 | 1.61 | 3117.45 | 0.08 | 6.44 |
| 39 | 3207.93 | 113.1 | 3267.08 | 0.87 | 1.84 | 3381 | 0.12 | 5.40 |
| 40 | 3098.45 | 68.51 | 3172.37 | 0.87 | 2.39 | 3224.98 | 0.09 | 4.08 |

**Table 2** continued

| Instance | LB column generation | | SA | | | Greedy | | |
|---|---|---|---|---|---|---|---|---|
| | ObjFun | CPU time (s) | ObjFun | CPU time (s) | Gap (%) | ObjFun | CPU time (s) | Gap (%) |
| 41 | 3059.52 | 158.3 | 3143.05 | 0.9 | 2.73 | 3242.26 | 0.13 | 5.97 |
| 42 | 3017.16 | 244.35 | 3152.88 | 0.92 | 4.50 | 3289.02 | 0.14 | 9.01 |
| 43 | 3000.85 | 208.37 | 3109.38 | 0.94 | 3.62 | 3286.24 | 0.08 | 9.51 |
| 44 | 3399.81 | 108.24 | 3448.66 | 0.89 | 1.44 | 3627.41 | 0.13 | 6.69 |
| 45 | 3270.9 | 121.82 | 3305.53 | 0.89 | 1.06 | 3478.31 | 0.17 | 6.34 |
| 46 | 3551.11 | 222.79 | 3643.72 | 0.93 | 2.61 | 3750.56 | 0.36 | 5.62 |
| 47 | 3586.77 | 175.49 | 3626.74 | 0.95 | 1.11 | 3859.89 | 0.15 | 7.61 |
| 48 | 3475.31 | 113.96 | 3582.6 | 0.96 | 3.09 | 3832.74 | 0.1 | 10.28 |
| 49 | 3566.04 | 306.76 | 3640.45 | 0.98 | 2.09 | 3807.77 | 0.1 | 6.78 |
| 50 | 3563.45 | 235.11 | 3620.6 | 0.96 | 1.60 | 3840.89 | 0.15 | 7.79 |
| Average | 2933.93 | 119.32 | 3003.86 | 0.84 | 2.45 | 3122.95 | 0.10 | 6.28 |

were solved by CPLEX via Concert technology and C#. In Table 2, the size of the instances are (70, 55) for instances 1–5, (70, 65) for instances 6–10, (70, 75) for instances 11–15, (70, 85) for instances 16–20, (70, 95) for instances 21–25, (80, 55) for instances 26–30, (80,65) for instances 31–35, (80, 75) for instances 36–40, (80, 85) for instances 41–45 and (80, 95) for instances 46–50 wherein the first argument is the number of vehicles and the second argument is the number of patients i.e., (# of Patients, # of Vehicles). Columns 2 and 3 show the objective function values and CPU times for the column generation algorithm. Columns 4 and 5 show the objective values and CPU times for the proposed hybrid simulated annealing algorithm. The gap between the SA and the lower bound solutions are provided in Column 6. Finally, Columns 7, 8, and 9 show the same information for the greedy algorithm introduced earlier in this section.

As shown in Table 2, the average gap of the solutions to the lower bound is 2.45% indicating that the SA algorithm was able to provide a solution very close to the lower bound. In contrast, the greedy algorithm is on average 6.28% worse than the lower bound. Thus, Table 2 shows that the proposed SA algorithm provides good quality solutions for large scale EMSVPTP instances.

In another experiment, we tested our algorithm for larger sized problems, specifically, more than 200 patients and 150 vehicles. Because none of the algorithms besides our proposed SA algorithm was able to find useful solutions, we do not show those results.

Finally, we compare the results obtained using the pure and hybrid SA methods. The pure SA method refers to the SA method using purely pairwise exchange neighborhood function in Subroutine 1, and the hybrid SA method refers to the one using Subroutine 1 and the nested vehicle search neighborhood function in Subroutine 3. As mentioned in Sect. 6, the tests on the hybrid method in Majzoubi et al. [11] suggest that the benefit of including the nested vehicle search is only significant after the temperature in SA is relatively low. Thus the hybrid SA method activates the nested vehicle search when the algorithm is in the last 5% of its iterations.

Table 3 shows the results of this hybrid method on the same 50 instances as in Table 2. Overall, this table suggests that the hybrid method can provide better solutions for 33 out of

**Table 3** Comparison between the pure SA and hybrid SA

| Instance | Hybrid SA | | Pure SA | | Gap (%) | Instance | Hybrid SA | | Pure SA | | Gap (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Obj function | CPU time | Obj function | CPU time | | | Obj function | CPU time | Obj function | CPU time | |
| 1 | 2284.49 | 0.76 | 2331.23 | 0.73 | 2.01 | 26 | 2497.20 | 1.32 | 2520.65 | 1.33 | 0.93 |
| 2 | 2291.18 | 0.77 | 2331.12 | 0.76 | 1.71 | 27 | 2694.79 | 1.47 | 2712.14 | 1.25 | 0.64 |
| 3 | 2595.02 | 0.70 | 2647.60 | 0.67 | 1.99 | 28 | 2479.32 | 1.83 | 2456.78 | 2.14 | −0.92 |
| 4 | 2615.07 | 0.69 | 2637.15 | 0.68 | 0.84 | 29 | 2653.02 | 1.57 | 2688.40 | 1.44 | 1.32 |
| 5 | 2340.10 | 0.70 | 2397.30 | 0.67 | 2.39 | 30 | 2750.18 | 1.20 | 2756.66 | 1.38 | 0.23 |
| 6 | 2872.90 | 0.74 | 2915.08 | 0.71 | 1.45 | 31 | 2713.36 | 1.46 | 2763.43 | 1.40 | 1.81 |
| 7 | 2696.37 | 0.89 | 2690.71 | 1.05 | −0.21 | 32 | 2801.25 | 1.48 | 2836.22 | 1.71 | 1.23 |
| 8 | 2885.25 | 0.83 | 2816.04 | 0.75 | −2.46 | 33 | 2763.29 | 0.78 | 2799.56 | 0.75 | 1.30 |
| 9 | 2654.33 | 0.78 | 2661.80 | 0.80 | 0.28 | 34 | 2947.37 | 0.77 | 2938.76 | 0.75 | −0.29 |
| 10 | 2761.02 | 0.77 | 2739.06 | 0.76 | −0.80 | 35 | 2914.64 | 0.76 | 2889.20 | 0.74 | −0.88 |
| 11 | 3191.91 | 0.87 | 3209.00 | 0.77 | 0.53 | 36 | 3102.35 | 0.84 | 3126.09 | 0.81 | 0.76 |
| 12 | 2989.80 | 0.89 | 3045.28 | 0.85 | 1.82 | 37 | 2950.78 | 0.85 | 2949.64 | 0.80 | −0.04 |
| 13 | 2617.64 | 0.93 | 2632.78 | 0.89 | 0.58 | 38 | 3021.86 | 0.87 | 2957.44 | 0.81 | −2.18 |
| 14 | 3100.01 | 0.92 | 3129.63 | 0.85 | 0.95 | 39 | 3265.34 | 0.91 | 3292.00 | 0.79 | 0.81 |
| 15 | 2748.26 | 1.21 | 2742.48 | 1.14 | −0.21 | 40 | 3164.02 | 0.85 | 3143.65 | 0.80 | −0.65 |
| 16 | 3240.49 | 0.90 | 3262.68 | 0.90 | 0.68 | 41 | 3184.16 | 0.89 | 3100.51 | 0.88 | −2.70 |
| 17 | 2977.08 | 1.45 | 2956.91 | 1.44 | −0.68 | 42 | 3115.91 | 0.98 | 3122.50 | 0.86 | 0.21 |
| 18 | 3453.92 | 1.47 | 3458.78 | 1.58 | 0.14 | 43 | 3099.65 | 0.99 | 3065.02 | 0.95 | −1.13 |
| 19 | 3072.14 | 1.41 | 3077.00 | 1.41 | 0.16 | 44 | 3426.76 | 0.88 | 3431.26 | 0.85 | 0.13 |
| 20 | 2998.24 | 1.43 | 3026.02 | 1.57 | 0.92 | 45 | 3350.66 | 0.88 | 3331.95 | 0.85 | −0.56 |
| 21 | 3220.79 | 1.74 | 3200.91 | 1.67 | −0.62 | 46 | 3592.79 | 0.94 | 3628.06 | 0.92 | 0.97 |
| 22 | 3121.17 | 1.86 | 3126.41 | 1.84 | 0.17 | 47 | 3696.13 | 0.92 | 3646.07 | 0.89 | −1.37 |
| 23 | 3764.94 | 1.80 | 4258.06 | 1.89 | 11.58 | 48 | 3566.52 | 0.95 | 3555.18 | 0.91 | −0.32 |
| 24 | 3432.99 | 1.66 | 3488.45 | 2.23 | 1.59 | 49 | 3655.50 | 0.95 | 3662.95 | 0.91 | 0.20 |
| 25 | 3348.14 | 1.89 | 3363.94 | 1.72 | 0.47 | 50 | 3615.45 | 0.94 | 3631.98 | 0.89 | 0.46 |
| Average | Hybrid SA—Obj: 3005.91, CPU time: 1.09 | | Pure SA—Obj: 3023.03, CPU time: 1.08 | | | Gap: 0.50% | | | | | |

50 instances with an average improvement on the objective value of 0.5%. For instance 23, the improvement is greater than 10%, showing the value of the nested local search algorithm.

## 8 Conclusions

In this paper, we consider the EMSVPTP problem during a demand surge. It minimizes the total time traveled by all vehicles while ensuring all patient requests are served within pre-specified time window constraints. To better utilize the EMS vehicles' capacity, each EMS vehicle can serve up to two patients. We present a mixed-integer programming formulation for the EMSVPTP problem, similar to the vehicle routing problem with pickup and delivery, based on the three-index formulation in [15]. We show the problem is NP-hard, and thus propose a simulated, we propose a simulated annealing heuristic method for its solution for real-world applications. Our numerical results show that the SA method is efficient in solving large-scale instances for EMSVPTP, when compared to exact solvers such as CPLEX. Future research could involve developing a customized exact algorithm for EMSVPTP using branch-and-price techniques.

## References


1. Russo, T.: Pandemic planning. Emerg. Med. Serv. **35**(10), 51–56 (2006)
2. Recommended Actions for EMS Providers to Prepare for and Respond to Pandemic. Los Angeles County Emergency Medical Services Agency (2009)
3. Emergency Medical Services and Non-Emergent (Medical) Transport Organizations Pandemic Influenza Planning Checklist. Department of Health and Human Services, USA (2006)
4. CPLEX, IBM ILOG.: V12. 1: User's manual for CPLEX. International Business Machines Corporation 46.53, p. 157 (2009)
5. Owen, S., Daskin, M.: Strategic facility location: a review. Eur. J. Oper. Res. **111**, 423–447 (1998)
6. Brotcorne, L., Laporte, G., Semet, F.: Ambulance location and relocation models. Eur. J. Oper. Res. **147**(3), 451–463 (2003)
7. Goldberg, J.: Operations research models for the deployment of emergency services vehicles. EMS Manag. J. **1**(1), 20–39 (2004)
8. Yang, S., Hamedi, M., Haghani, A.: Integrated approach for emergency service location and assignment problem. J. Transp. Res. Board **1882**, 184–192 (2004)
9. Andersson, T., Varbrand, P.: Decision support tools for ambulance dispatch and relocation. J. Oper. Res. Soc. **58**, 195–201 (2007)
10. Ibri, S., Nourelfath, M., Drias, H.: A parallel hybrid ant-Tabu algorithm for integrated emergency vehicle dispatching and covering problem. Int. J. Innov. Comput. Appl. **2**(4), 226–236 (2010)
11. Majzoubi, F., Bai, L., Heragu, S.S.: An optimization approach for dispatching and relocating EMS vehicles. IISE Trans. Healthc. Syst. Eng. **2**(3), 211–223 (2012)
12. Gendreau, M., Guertin, F., Potvin, J.Y., Seguin, R.: Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. Transp. Res. Part C Emerg. Technol. **14**(3), 157–174 (2006)
13. Bent, R., Hentenryck, P.V.: A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Comput. Oper. Res. **33**(4), 875–893 (2006)
14. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**(4), 455–472 (2006)
15. Cordeau, J., Laporte, G.: The dial-a-ride problem: models and algorithms. Ann. Oper. Res. **153**(1), 29–46 (2007)
16. Ho, S.C., Kuo, Y.H., Leung, J.M.Y., Petering, M., Tou, T.W.H.: A survey of dial-a-ride problems: literature review and recent developments. Transp. Res. Part B **111**, 395–421 (2018)
17. Beaudry, A., Laporte, G., Melo, T., Nickel, S.: Dynamic transportation of patients in hospitals. OR Spectr. **32**, 77–107 (2010)

18. Kergosien, Y., Lente, Ch., Piton, D., Billaut, J.C.: A Tabu search heuristic for the dynamic transportation of patients between care units. Eur. J. Oper. Res. **214**, 442–452 (2011)
19. Parragh, S.: Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. Transp. Res. Part C Emerg. Technol. **19**(5), 912–930 (2011)
20. Hu, T.-Y., Chang, C.-P.: A revised branch-and-price algorithm for dial-a-ride problems with the consideration of time-dependent travel cost. J. Adv. Transp. **49**(6), 700–723 (2014)
21. Liu, M., Luo, Z., Lim, A.: A branch-and-cut algorithm for a realistic dial-a-ride problem. Transp. Res. Part B Methodol. **81**, 267–288 (2015)
22. Markovic, N., Nair, R., Schonfeld, P., Miller-Hooks, E., Mohebbi, M.: Optimizing dial-a-ride services in Maryland: benefits of computerized routing and scheduling. Transp. Res. Part C Emerg. Technol. **55**, 156–165 (2015)
23. Krumke, S., Saliba, S., Vredeveld, T., Westphal, S.: Approximation algorithms for a vehicle routing problem. Math. Methods Oper. Res. **68**(2), 333–359 (2008)
24. Haimovich, M., Rinnooy Kan, A.: Bounds and heuristics for capacitated routing problems. Math. Oper. Res. **10**(4), 527–542 (1985)
25. Desrochers, M., Desrosiers, J., Solomon, M.: A new optimization algorithm for the vehicle routing problem with time windows. Oper. Res. **40**(2), 342–354 (1992)
26. Xu, H., Chen, Z.L., Rajagopal, S., Arunapuram, S.: Solving a practical pickup and delivery problem. Transp. Sci. **37**(3), 347–364 (2003)
27. Feillet, D., Dejax, P., Gendreau, M., Gueguen, C.: An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. Networks **44**(3), 216–229 (2004)
28. Ropke, S., Cordeau, J.F.: Branch and cut and price for the pickup and delivery problem with time windows. Transp. Sci. **43**(3), 267–286 (2009)