A Constraint Programming Approach for Solving Patient Transportation Problems

Quentin Cappart[1,2,3](✉), Charles Thomas[1], Pierre Schaus[1], and Louis-Martin Rousseau[2,3]

[1] Université catholique de Louvain, Louvain-la-Neuve, Belgium
{charles.thomas,pierre.schaus}@uclouvain.be
[2] Ecole Polytechnique de Montréal, Montréal, Canada
quentin.cappart@polymtl.ca
[3] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montréal, Canada
louis-martin.rousseau@cirrelt.ca

Abstract. The Patient Transportation Problem (PTP) aims to bring patients to health centers and to take them back home once the care has been delivered. All the requests are known beforehand and a schedule is built the day before its use. It is a variant of the well-known Dial-a-Ride Problem (DARP) but it has nevertheless some characteristics that complicate the decision process. Three levels of decisions are considered: selecting which requests to service, assigning vehicles to requests and routing properly the vehicles. In this paper, we propose a Constraint Programming approach to solve the Patient Transportation Problem. The model is designed to be flexible enough to accommodate new constraints and objective functions. Furthermore, we introduce a generic search strategy to maximize efficiently the number of selected requests. Our results show that the model can solve real life instances and outperforms greedy strategies typically performed by human schedulers.

1 Introduction

Over the years, there is an increasing demand for transports by disabled and invalid people requiring health care but that do not have the ability to go to hospitals by themselves. In this context, organizations managing the transportation of patients from their home to health centers are present in many cities. Their goal is to provide a door-to-door transportation service to a set of patients on a daily basis. Most of them are non-profit organizations that often have limited resources. Besides, they often do not have an expertise on decision support tools in order to assist them in their operations. This leads to sub-optimal decisions in most cases which has a direct negative impact on the patients and also leads to financial losses. Therefore, minimizing the operational costs while maintaining a sufficient quality of service is highly desirable and both aspects must be properly balanced.

J. Hooker (Ed.): CP 2018, LNCS 11008, pp. 490–506, 2018.
https://doi.org/10.1007/978-3-319-98334-9_32

The problem considered in this paper has been proposed by a non-profit organization operating at Liège (Belgium) which provides a range of home help services. One of them is transportation of people for medical appointments. We refer to it as the Patient Transportation Problem, which is a specific case of the well-known Dial-a-Ride Problem (DARP) [1]. The goal of this last problem consists in designing routes and schedules for a set of users who specify pickup and delivery requests between origins and destinations. It is especially present for the transportation services in the medical domain [2–4].

However, a tremendous amount of variants are possible and have been extensively studied in the literature: the fleet can be composed of several vehicles [5] that can be heterogeneous [6], users can have different characteristics [7], availability of vehicles can be constrained [8], patients can require a return trip [9], several depots can be present [10], etc. A large scope of objective functions can also be considered such as minimizing the waiting time of users or maximizing the number of accepted requests. Multi-objective approaches have also been introduced [11]. Besides, the problem can either be solved offline [12] or online [13]. In the former case, all the requests are known in advance whereas they appear gradually in real-time in the latter. Aforementioned references are only few examples of the broad literature dedicated to DARPs. A good summary of the different variants and methods was nevertheless proposed by Cordeau and Laporte [1]. As a first observation, we can see that most of the approaches are based either on Mixed Integer Programming, Local Search or Dynamic Programming. Conversely, solutions based on Constraint Programming (CP) seem to have been less studied even if some recent works exist [14–18]. However, thanks to its flexibility, we believe that CP can play an important role for solving practical DARPs.

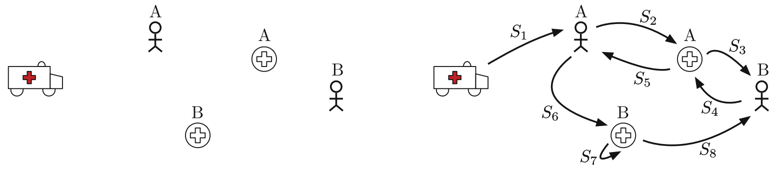
The contribution of this paper is a flexible and efficient approach based on CP for modeling and solving the static Patient Transportation Problem, which is a specific case of the DARP. A general model is first proposed. Several extensions that can be easily integrated to the model are then detailed. A generic search strategy for maximizing the number of selected requests is also proposed. It avoids branching on variables related to a request whenever it is not selected. From a practical point of view, we provide a solution to a problem issued by a non-profit organization, handling the transportation logistic of people requiring health care. The solution we propose is usable in practice, thanks to its efficiency and flexibility to accommodate new situations. Performances of the approach are corroborated by both synthetic and real instances.

This paper is organized as follows. Next section describes the nature of the problem we are considering. Section 3 presents some recent developments related to the problem studied. A core model with its different components is firstly detailed in Sect. 4. Additional features that can be easily integrated in the model are then presented in Sect. 5. Finally, experiments on synthetic and real instances are carried out in Sect. 6.

2 Problem Description

The Patient Transportation Problem (PTP) is a static optimization problem aiming to bring patients to health centers and to take them back home once the care has been delivered. To do so, a fleet of vehicles is available. The fleet is heterogeneous and is mainly composed of ambulances and private drivers operating as volunteers. Each patient has a set of characteristics and is represented by a request. The objective is to satisfy as many requests as possible within a fixed horizon, which is typically bounded by the working hours. Three aspects of decision are considered in the PTP: (1) selecting which requests to service, (2) assigning vehicles to requests and (3) routing and scheduling appropriately the vehicles. An illustration of the PTP on a toy example with two patients (A and B) and a single vehicle is shown in Fig. 1. A possible solution consists in the following sequence: taking A (S_1), bringing A to the hospital (S_2), taking B (S_3), taking back A (S_4), dropping A to its home (S_5), bringing B to the hospital (S_6), waiting for B (S_7) and dropping B to its home (S_8). Some specific characteristics must also be considered in the PTP. Here are some of them:

- Patients can have several constraints such as a maximum travel time or a maximum waiting time at the hospital. The time to embark and disembark a patient must sometimes be considered.
- The set of requests is heterogeneous. Some patients only require to go from their home to a health center, while some of them also need a return trip once the care has been delivered. In the latter case, they must be taken back home, or to another place if requested. It is also possible to have patients requiring only a return trip. Besides, requests can involve more than one passenger at once. For instance, a child can be accompanied by his parents.
- The vehicle fleet is heterogeneous. Vehicles can differ by their capacity, their initial/final location (typically a depot) and their availability. Some patients can only be taken by particular vehicles. For instance, patients in wheelchairs can only be transported by specific vehicles.
- Availability of vehicles can be non continuous. For instance, they can be available from 9am to 1pm and from 3pm to 6pm.



(a) The initial situation. (b) A possible solution.

Fig. 1. Illustration of the PTP with one vehicle and two patients.

Let us finally notice that this version of PTP is static: the whole set of requests is known beforehand and no new request is added in real time. it is used by the organization for designing the first daily schedule given the pool of requests received the previous days.

3 Related Work

To the best of our knowledge, the approach of Liu et al. [18] is the closest and most recent work related to our problem. The authors model and solve the Senior Transportation Problem (STP) using different approaches: CP, MIP and Logic Based Benders Decomposition. The objective is also to maximize the sum of the (weighted) served requests. Their results show that the CP model has the best performances. The STP shares many similarities with our problem but has nevertheless some differences:

- Requests are one-way only and there is no return trip.
- The problem is a transportation problem where the selection of each request is constrained only by the vehicles availability and a maximum travel time, there are no constraints related to the appointment for care.
- There are no constraints linking patients to specific vehicles.

While some constraints are straightforward to add in the STP model, the integration of others would require more modifications. For instance, by properly defining the time windows to make sure the patients arrive on time for their care, appointment constraints for the care can be handled by the STP. However, additional constraints would be necessary to link forward with backward trips and preserve the consistency of the tour. Ensuring that vehicles are the same or can be different for both trips also requires some modifications.

Besides, the modeling and solving parts are also different. In the approach of Liu et al. [18], each decision variable is linked to a location and auxiliary variables are introduced to express that a location is visited by a particular vehicle. In our model, the decision variables are linked to trips instead of visited locations. We express capacity constraints with the standard `cumulative` constraint [19] and can take advantage of efficient propagators [20–24]. Conversely, Liu et al. [18] enforce the capacity constraints of vehicles through renewable resources and *cumul* functions using the `StepAtStart` functions from CP Optimizer. Those abstractions are less standard in CP solvers and modeling languages such as Minizinc or XCSP3 (renewable resources can be modeled with cumulative constraints [25]). Finally we use a custom search strategy combined with a Large Neighborhood Search while Liu et al. rather uses the CP Optimizer default search.

4 Modeling

This section presents a CP model for the PTP, flexible to easily handle different variants of the problem, and efficient enough to solve real instances. The PTP is modeled as a constrained based scheduling problem.

Parameters. Let R be the set of requests and V the set of vehicles. Each request is linked to a patient. The related parameters are depicted in Table 1. Some of them correspond to a location ($start_i$, $dest_i$ and ret_i) and are used for computing a travel time matrix ($T_{i,j}$) from location i to j. A request consists in two trips: a forward trip from a start location to a destination ($start_i$ to $dest_i$) and a backward trip from the previous destination to a return location ($dest_i$ to ret_i).

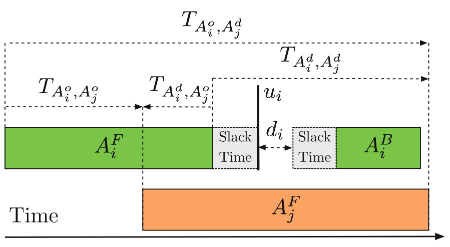
Table 1. Parameters used in the model.

Entity	Parameter	Meaning
Request	$start_i$	Starting place of the patient linked to request i
	$dest_i$	Place where the care is delivered for the patient of request i
	ret_i	Return place of the patient linked to request i
	l_i	Number of places taken by the patient of request i
	u_i	Time at which the health care service begins for request i
	d_i	Time needed to deliver the care for the patient of request i
	p_i	Maximum travel time of the patient linked to request i
	c_i	Category of patient of request i (wheelchair, without, etc.)
Vehicle	k_j	Capacity of vehicle j (i.e. the number of places available)
	C_j	Set of patient categories that vehicle j can take

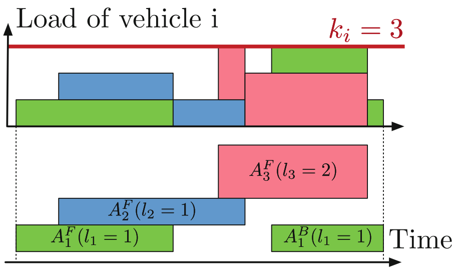
Decision Variables. The problem is to choose which requests will be selected, the vehicles assigned to the requests, the route of the vehicles and their timetable. We model it as a scheduling problem with conditional activities using the formalism proposed by Laborie et al. [26–28]. In the standard form, each conditional activity A_i is modeled with four variables, a start date $s(A_i)$, a duration $d(A_i)$, an end date $e(A_i)$ and a binary execution status $x(A_i)$. If the activity is executed, it behaves as a classical activity that is executed on its time interval, otherwise it is not considered by any constraint. In our case, we also define $v(A_i)$ as the vehicle that has been assigned to an activity A_i. Each request (i) is attached to a forward activity (A_i^F) defining the time slot when the patient is brought from its home to the health center (from $start_i$ to $dest_i$) and to a backward activity (A_i^B) for the time interval of the return trip (from $dest_i$ to ret_i). Furthermore, A_i denotes any activity, either forward or backward, A^F the set of forward activities and A^B the set of backward activities. Equation 1 defines A_i^o and A_i^d as the origin and the destination locations of the activities linked to a request i.

$$\forall i \in R : \begin{cases} A_i^o = \begin{cases} start_i \text{ if } A_i \in A^F \\ dest_i \ \text{ if } A_i \in A^B \end{cases} \\ A_i^d = \begin{cases} dest_i \text{ if } A_i \in A^F \\ ret_i \ \text{ if } A_i \in A^B \end{cases} \end{cases} \tag{1}$$

Temporal relations between activities are illustrated in Fig. 2a for an arbitrary example. The focus is on activity A_i^F. There are four specific transition times ($T_{x,y}$) with any other activity (A_j^F on this example), they correspond to the time to go from A_i^o to A_j^o, from A_i^o to A_j^d, from A_i^d to A_j^o and from A_i^d to A_j^d. Activity A_i^F must also be completed before the appointment of the request (u_i), and the related backward activity cannot begin before the end of the appointment ($u_i + d_i$). Finally, each activity is executed on a resource, representing the vehicle assigned to the activity. At any moment, the load of the vehicle cannot exceed its capacity. It is illustrated in Fig. 2b by a load profile for an arbitrary set of 4 activities executed on the same vehicle.



(a) Temporal dependencies.



(b) Load constraints.

Fig. 2. Illustration of the parameters of Table 1.

Decision variables related to the selection of requests are depicted in Eq. 2. They are boolean variables defining whether the request is selected or not.

$$\forall i \in R: \; S_i \in \{0,1\} \tag{2}$$

Variables related to the conditional activities are shown in Eq. 3. Patients cannot arrive at the health center after the time at which the appointment begins (forward activity) and cannot leave it before the end of the care (backward activity). Symbol H denotes the time horizon considered. The domain of the vehicle selection variables (v) contains only the vehicles that are compatible with the patient category of the request. Note that the duration variable (d) is not a decision variable as its value depends on the start (s) and the end (e) of the activity. Domains for forward activities implicitly handle the deadline satisfaction for the care for each request. It ensures that the patients arrive to the health center ahead of schedule for their care ($e(A_i^F) \leq u_i$). Similarly, domains for backward activities ensure that patients cannot leave the center before the time at which the care has been delivered ($s(A_i^B) \geq u_i + d_i$).

$$\forall i \in R: \begin{cases} s(A_i^F) \in [0, u_i] \\ e(A_i^F) \in [0, u_i] \\ d(A_i^F) = e(A_i^F) - s(A_i^F) \\ x(A_i^F) \in \{0,1\} \\ v(A_i^F) \in \{j \mid j \in V \wedge c_i \in C_j\} \end{cases} \begin{cases} s(A_i^B) \in [u_i + d_i, H] \\ e(A_i^B) \in [u_i + d_i, H] \\ d(A_i^B) = e(A_i^B) - s(A_i^B) \\ x(A_i^B) \in \{0,1\} \\ v(A_i^B) \in \{j \mid j \in V \wedge c_i \in C_j\} \end{cases} \tag{3}$$

Constraints

Binding Requests to Activities. A request is selected if and only if the forward and backward activities are both completed (Eq. 4).

$$\forall i \in R: \ \big(S_i = 1\big) \equiv \big(x(A_i^F) = 1 \wedge x(A_i^B) = 1\big) \tag{4}$$

Forward and Backward Selection. A forward and backward activity linked to the same request must have the same execution status (Eq. 5). This constraint is redundant with Eq. 4 but can nevertheless be used for a better pruning.

$$\forall i \in R: \ x(A_i^F) = x(A_i^B) \tag{5}$$

Inter-Activity Time Travel Consistency. The start/end of an activity cannot overlap with the start/end of other activities when they are processed by the same vehicle. The time interval between any two locations visited by a same vehicle is at least the time required to travel between these two locations (Eq. 6). It is also referred as *setup time*. It is illustrated in Fig. 2a. The $\vee$ relation is used to consider situations where activity A_i occurs before or after A_j.

$$\forall i, j \in R \mid i \neq j: \begin{cases} (v(A_i) = v(A_j)) \rightarrow \big((s(A_j) - s(A_i) \geq T_{A_i^o, A_j^o}) \vee (s(A_i) - s(A_j) \geq T_{A_j^o, A_i^o})\big) \\ (v(A_i) = v(A_j)) \rightarrow \big((s(A_j) - e(A_i) \geq T_{A_i^o, A_j^d}) \vee (s(A_i) - e(A_j) \geq T_{A_j^o, A_i^d})\big) \\ (v(A_i) = v(A_j)) \rightarrow \big((e(A_j) - s(A_i) \geq T_{A_i^d, A_j^o}) \vee (e(A_i) - s(A_j) \geq T_{A_j^d, A_i^o})\big) \\ (v(A_i) = v(A_j)) \rightarrow \big((e(A_j) - e(A_i) \geq T_{A_i^d, A_j^d}) \vee (e(A_i) - e(A_j) \geq T_{A_j^d, A_i^d})\big) \end{cases} \tag{6}$$

An alternative way to enforce the travel times is to use a `NoOverlap` with transition time constraint imposed on activities created at each location [18]. In particular, the propagator proposed by Dejemeppe et al. [29] could possibly be extended to handle optional activities. But the decomposition approach relying on reification and binary constraints is arguably the most portable formulation for other solvers and modeling languages.

Intra-Activity Time Travel Consistency. The duration of each activity cannot be lesser than the time required to go from the origin to the destination (Eq. 7).

$$\forall i \in R: \ d(A_i) \geq T_{A_i^o, A_i^d} \tag{7}$$

Maximum Travel Time. It is also suitable to constraint the maximal travel time of patients. It prevents situations where a patient stays too long in a vehicle. To do so, the duration of each activity is constrained (Eq. 8).

$$\forall i \in R: \ d(A_i) \leq p_i \tag{8}$$

Cumulative Resource. At any moment, the number of places occupied by patients in a same vehicle j cannot exceed its capacity k_j (Eq. 9). This behaviour is illustrated in the arbitrary example of Fig. 2b. This constraint is referred in the literature as the cumulative resource global constraint [19]. In our case, each activity A_i consumes l_i resources. We use the filtering algorithm of Gay et al. [20]. The vehicle of a non-executed activity is not considered by the constraint.

$$\forall j \in V : \ \texttt{cumulative}\Big(\Big\{(A_i, l_i) \mid i \in R \wedge v(A_i) = j\Big\}, k_j\Big) \tag{9}$$

Objective Function. The first criterion considered for the objective function is the satisfaction of requests. We want to maximize the number of served requests (Eq. 10). Other objective functions can be considered. For instance, we could be interested in minimizing the accumulated travel time for all the patients (Eq. 11). The travel time of a request corresponds to the duration of its activities. It is also possible to minimize the maximum travel time (Eq. 12). To do so, the maximal duration of the whole set of activities has to be minimized. Other objective functions are also proposed by Cordeau and Laporte [1]. They can be used together inside the same model using either a lexicographic ordering or a Pareto multi objective criterion [30].

$$\max\Big(\sum_{i \in R} S_i\Big) \tag{10}$$

$$\min\Big(\sum_{i \in R} d(A_i)\Big) \tag{11}$$

$$\min\Big(\max_{i \in R} d(A_i)\Big) \tag{12}$$

Search Phase. The search tree is explored using a standard branch and bound depth first search. The decision variables are divided into two categories: the *request variables* (Eq. 2) and the *activity variables* (Eq. 3). Given the main objective of the problem (maximizing the number of served patients), our primal heuristic is to select patients on the left branches ($S_i = 1$) and discard them on the right branches ($S_i = 0$). Whenever a patient has been selected in a search node, all its related activity variables are subsequently assigned (start time, duration and end time and vehicle) before considering again the next patient selection variable. On the contrary, whenever a patient is not selected ($S_i = 0$ on the right branch), there is no need to consider the other decision variables related to this patient. The idea is to branch on the activity variables only if the related request variable has been selected ($S_i = 1$). Otherwise, no search is performed on the activity variables. We denote this search strategy as the *Maximum Selection Search*. The main asset of this search is that activity variables are branched on only when they are relevant to a solution. It drastically reduces the size of the search tree. An example of search tree is illustrated in Fig. 3.

This meta-search strategy for optional activities can be combined with any existing variable-value heuristic or used for similar applications such as packing as most rectangles as possible. As a variable heuristic on the request variables we use a Conflict Ordering Search heuristic (COS) [31]. A conflict is recorded on a request only when it is impossible to assign in the sub-tree all its other activity variables. The fallback heuristic combined with COS is to select the next requests with the highest *minimum slack*, defined as the sum of the minimum duration multiplied by the patient load for its forward and backward activities. The subsearch on the other activity variables follows a *min-domain first fail* strategy for the variable selection and a custom greedy value heuristic based on the type of the corresponding variable which can be a time-related decision or a vehicle choice. In the former case, the heuristic selects the closest time to the corresponding appointment. In the latter case, the vehicle that has the most remaining places is selected.

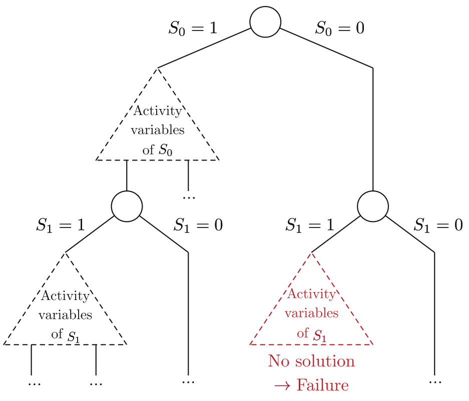


Fig. 3. Canonical shape of the search tree for two request variables (S_0 and S_1).

Large Neighborhood Search. In order to boost the performances on large instances, a Large Neighborhood Search (LNS) [32] is also used. At each iteration, a set of request variables is chosen randomly and then relaxed. The other variables are fixed to their value in the last solution. For the request variables that are selected ($S_i = 1$), the corresponding activity variables are also fixed based on the current solution. The remaining unbound variables form a smaller search space which is explored using the search defined earlier. A new iteration is started when the reduced search space is completely explored or once a fixed number of backtracks is reached.

5 Extensions of the Model

One of the main asset of this model is its flexibility to easily accommodate new constraints depending on the situation. This section presents some variants of the problem and how they can be integrated in the core model.

Mandatory Requests. It is possible to enforce the selection of some requests (Eq. 13). Parameter m_j is a boolean value indicating if a request j is mandatory.

$$\forall i \in \{j \mid j \in R \wedge m_j = 1\} : \ S_i = 1 \tag{13}$$

Maximum Waiting Time. The time that a patient has to wait at the health center, either before or after his care, is often constrained. Parameter w_i indicates the maximum amount of time that a patient can wait. It is handled by adapting the definition of domains in Eq. 3. It avoids situations where patients are dropped to the health center too early or taken back too late (Eq. 14).

$$\forall i \in R : \begin{cases} s(A_i^F) \in [0, u_i - \boldsymbol{w_i}] \\ e(A_i^F) \in [0, u_i - \boldsymbol{w_i}] \\ (...) \end{cases} \quad \begin{cases} s(A_i^B) \in [u_i + d_i + \boldsymbol{w_i}, H] \\ e(A_i^B) \in [u_i + d_i + \boldsymbol{w_i}, H] \\ (...) \end{cases} \tag{14}$$

Integrating Service Time. Most often, the time required to embark or disembark a patient is negligible. However in some cases, it could be more representative to consider it. For instance, embarking a patient with a wheelchair can take a significant amount of time. Such a dependency can be integrated in the inter-activity time travel consistency constraints defined in Eq. 6.

Vehicles Availability. Vehicles can also have constraints on their availability. They are available during a period and cannot leave their initial position (i.e. a depot) before the period. Similarly, they have to go back to the depot before the end of the period. Let us introduce $start_j$ the starting location of a vehicle j, $dest_j$ its return destination and $[b_j, r_j]$ its availability window. The travel time matrix (T) defined previously is extended in order to take into account these new locations. We define $D_i^o = start_{v(A_i)}$ and $D_i^d = dest_{v(A_i)}$ as the origin/destination location of the vehicle linked to activity A_i as defined in Eq. 3. Constraints on vehicle availability are expressed in Eq. 15. It states that an activity cannot begin before the availability of its vehicle plus the time required to go from the initial depot to the patient place. Similarly, the vehicles must have enough time to return to their depot in order to stay in the availability window.

$$\forall i \in R : \begin{cases} s(A_i) \geq b_{v(A_i)} + T_{D_i^o, A_i^o} \\ e(A_i) \leq r_{v(A_i)} - T_{A_i^d, D_i^d} \end{cases} \tag{15}$$

Finally, some vehicles can have non continuous availability. For instance, they can be available from 9am to 1pm and from 3pm to 6pm. We handle this specificity by duplicating the vehicles for each continuous interval. The availability of each vehicle is then composed by a unique interval. In practice, vehicles are duplicated at most once (morning and afternoon shift).

Same Vehicle Forward/Backward. The forward and the backward trips can be constrained in order to be handled by the same vehicle (Eq. 16). Parameter q_j is a boolean value indicating if the forward and the backward trip of request j must be handled by the same vehicle.

$$\forall i \in \big\{j \mid j \in R \wedge q_j = 1\big\} : \; v(A_i^F) = v(A_i^B) \tag{16}$$

Empty Locations. Some patients only require to go from their home to a health center without return trip. It is also possible to have patients needing only a trip from the health center to their home. A location can then be empty. When the start location is empty the request has no forward trip. Similarly, there is no backward trip when the return location is empty. This variant is handled by extending the notion of requests. A request has no forward activity when the start location is empty and no backward activity when the return location is empty. In such cases, some constraints of the previous are simplified or removed in order to consider only situations involving a forward or a backward activity. More specifically, Eq. 4 is adapted as follows (Eq. 17, $\vee$ instead of $\wedge$) and constraint in Eq. 5 does not hold anymore.

$$\forall i \in R : \; \big(S_i = 1\big) \equiv \big(x(A_i^F) = 1 \vee x(A_i^B) = 1\big) \tag{17}$$

6 Experimental Results

This section evaluates the performance of the model on synthetic and real instances. The model tested is referred as the *Scheduling with Maximal Selection Search* (`SCHED+MSS`) approach. It corresponds to the core model described in Sect. 4 with the following extensions: *maximum waiting time*, *integrating service time*, *vehicles availability* and *empty locations*. No constraints on the *maximum travel time* were asked by the partner organization. Finally, the objective considered is to maximize the number of requests satisfied (Eq. 10).

Approaches Considered. Our model is compared with four other approaches: a greedy search, the same CP model without the maximal selection search, a similar scheduling model implemented in CP Optimizer and a successor model, more standard for solving routing problems with CP.

Greedy Search (`GREEDY`). It mimics the manual decision process used by the non-profit organization. It consists in selecting first the requests having the smallest starting time and choosing for them the closest compatible vehicle. The idea is to minimize the time between the trips of each vehicle across the requests. Each trip is inserted at the earliest possible time such that later trips can be inserted with more flexibility. If a trip cannot be inserted, the request is discarded.

Successor Model (SUCC). As an alternative to our approach, a successor model was considered. Similar models were used for solving DARPs using CP [15,17]. Each trip is represented by two stops which correspond to the place where the patient is loaded and the place where they are unloaded. Each request has then two or four stops depending on whether it is a single trip or a double trip. The successor and the predecessor of each stop are both modeled by a variable indicating the next and the previous stop. As in [17], ride time and vehicle capacity constraints are modeled via auxiliary variables representing the load, serving vehicle, and serving time for each requests. A `circuit` constraint [33] ensures that the successor and predecessor variables form a circuit without subtours for each vehicle. The requests that are not serviced are assigned to a same dummy vehicle with infinite capacity. Finally, a maximum selection heuristic wrapped under LNS and a COS variable heuristic are also used for the search.

CP Optimizer implementation (CPO). The scheduling model has been implemented in IBM CP Optimizer in order to compare our search with the default search proposed by this solver. This search combines LNS with a failure directed search (FDS) strategy [34]. In order to accommodate the solver, the capacity constraints of vehicles are modeled using *cumul* functions in the same way as in the model of Liu et al. [18].

Scheduling Model with Simple Search (SCHED). It corresponds to the model presented in the previous section without the *maximal selection search* heuristic. Additional reified constraints assign the activity variables to a default value when a request is not served. It is used to avoid wasting time searching on activity variables when the corresponding request is not selected.

Datasets Used. The experiments are based on two datasets, a synthetic and a real one. The synthetic dataset has been randomly generated based on the characteristics of the problem. Synthetic instances are classified according to their size (number of patients, vehicles and health centers) and their difficulty which is related to the amount of constraints and the availability of vehicles. The real dataset has been provided by the non-profit organization. It corresponds to one month of exploitation with one instance per day. Each of them contains the requests received for the day, the vehicles available.

Experimental Protocol. Experiments have been carried out on an AMD Opteron 6176 processor (2300 MHz). Execution time for a run is limited to 1800 s and memory consumption to 6 GB. The greedy search has been implemented in Scala and the OscaR solver [35] is used for the other models except for the CPO model that has been modeled and solved with the academic version of IBM ILOG CPLEX CP Optimizer V12.8. For the reproducibility of results, the models, the synthetic dataset and the random generator are available online on CSPLib [36].

The backtrack limit and relaxation size of the LNS are adaptive parameters initially fixed to respectively 1000 failures and 10 requests. The backtrack limit is increased by 20% when 100 consecutive iterations have failed to find a new solution and to completely explore the search. The relaxation size is increased by 20% when the relaxed search space is completely explored for 50 consecutive iterations. Search parameters are set to their defaults for `CPO`. The greedy solution is considered as the first solution of the LNS for each method.

Given the random nature of approaches based on LNS (`SUCC`, `CPO`, `SCHED` and `SCHED+MSS`), 5 runs for each instance with a different seed have been performed and the best solution obtained is recorded. The greedy search (`GREEDY`) is ran only once due to its deterministic nature. The models are also compared using the improvement ratio (ρ_m) of a method (m) defined as the relative improvement of the solution obtained with the method (x_m) compared to the solution found using the greedy search ($x_{\texttt{GREEDY}}$): $\rho_m = \frac{x_m - x_{\texttt{GREEDY}}}{x_{\texttt{GREEDY}}}$.

Results. Results for both synthetic and real instances are reported in Table 2. Instances are ordered by their difficulty and the number of patients ($|R|$). The best solution obtained for each instance is also reported. The number of patients serviced is considered as the objective value. As the relaxation size is adaptive, it can eventually grow to 100%. In this case, if the search space is completely explored, the solution is proven optimal. Besides, if all the patients are serviced, the upper bound is reached and the solution is also proven optimal. The dominating model is highlighted for each instance.

Let us first focus on synthetic instances. As we can see, the scheduling model with the maximal selection search (`SCHED+MSS`) obtains the best solution for almost all the tests, even when the optimum is not reached. The improvement ratio is up to 130% compared to the greedy solution. Interestingly, performance of scheduling models is correlated with the difficulty of instances: the improvement gap increases when the instances are getting harder. The greedy search (`GREEDY`) gives poor solutions when the problem is strongly constrained. Results regarding the scheduling model with the simple search (`SCHED`) shows the interest of the custom search.

The successor model (`SUCC`) is outperformed by the scheduling approaches. This is expected as the successor model has a larger search space due to the additional decisions variables compared to the scheduling model. Furthermore, the successor approach makes the insertion of new stops in routes more difficult as it requires to change the value of the successor variables forming the routes in addition to the vehicle variable. This limits the effectiveness of the LNS.

Concerning the CP Optimizer model (`CPO`), it is also outperformed by the two other scheduling approaches. Such results are mainly due because of the default search used in `CPO` model: it is generic and not designed for this specific problem. However, it is important to point out that on harder instances, it tends to perform better than the successor model. This could indicate that the model used contributes more to the effectiveness of the approach than the search method. Note that as the `CPO` approach is based on another solver, other factors could also influence the performances.

Table 2. Experimental results ($|R|$, $|V|$ and $|H|$ are the number of requests, vehicles and hospitals ; ρ is the improvement ratio in percent, * indicates that the solution has been proven optimal).

Instances						GREEDY	SUCC		CPO		SCHED		SCHED+MSS	
Difficulty	Name	$\|H\|$	$\|V\|$	$\|R\|$	BestSol	Sol	Sol	ρ	Sol	ρ	Sol	ρ	Sol	ρ
	RAND-E-1	4	2	16	*15	14	15	7.1	*15	7.1	*15	7.1	15	7.1
	RAND-E-2	8	4	32	*32	32	32	0.0	*32	0.0	*32	0.0	*32	0.0
	RAND-E-3	12	5	48	*28	26	26	0.0	*28	7.7	28	7.7	*28	7.7
	RAND-E-4	16	6	64	62	58	61	5.2	59	1.7	62	6.9	62	6.9
Easy	RAND-E-5	20	8	80	74	72	73	1.4	72	0.0	73	1.4	74	2.8
	RAND-E-6	24	9	96	95	91	93	2.2	92	1.1	92	1.1	95	4.4
	RAND-E-7	28	10	112	106	100	101	1.0	100	0.0	103	3.0	106	6.0
	RAND-E-8	32	12	128	*128	127	*128	0.8	127	0.0	*128	0.8	*128	0.8
	RAND-E-9	36	14	144	142	141	142	0.7	141	0.0	142	0.7	142	0.7
	RAND-E-10	40	16	160	157	154	154	0.0	157	1.9	157	1.9	157	1.9
	RAND-M-1	8	2	16	*12	8	9	12.5	11	37.5	*12	50.0	11	37.5
	RAND-M-2	16	3	32	19	16	18	12.5	17	6.3	19	18.8	19	18.8
	RAND-M-3	24	4	48	32	25	25	0.0	26	4.0	30	20.0	32	28.0
	RAND-M-4	32	4	64	37	25	25	0.0	33	32.0	35	40.0	37	48.0
Medium	RAND-M-5	40	5	80	55	45	45	0.0	48	6.7	51	13.3	55	22.2
	RAND-M-6	48	5	96	52	36	40	11.1	40	11.1	50	38.9	52	44.4
	RAND-M-7	56	6	112	63	46	47	2.2	48	4.3	63	37.0	63	37.0
	RAND-M-8	64	8	128	83	65	70	7.7	65	0.0	81	24.6	83	27.7
	RAND-M-9	72	8	144	81	62	62	0.0	64	3.2	72	16.1	81	30.6
	RAND-M-10	80	9	160	99	73	75	2.7	75	2.7	88	20.5	99	35.6
	RAND-H-1	16	2	16	*8	7	7	0.0	*8	14.3	*8	14.3	*8	14.3
	RAND-H-2	32	3	32	19	15	15	0.0	18	20.0	19	26.7	17	13.3
	RAND-H-3	48	4	48	32	18	19	5.6	23	27.8	32	77.8	29	61.1
	RAND-H-4	64	4	64	23	10	12	20.0	22	120.0	20	100.0	23	130.0
Hard	RAND-H-5	80	5	80	42	29	31	6.9	29	0.0	38	31.0	42	44.8
	RAND-H-6	96	5	96	38	22	22	0.0	27	22.7	38	72.7	38	72.7
	RAND-H-7	112	6	112	39	25	27	8.0	32	28.0	37	48.0	39	56.0
	RAND-H-8	128	8	128	75	57	63	10.5	61	7.0	71	24.6	75	31.6
	RAND-H-9	144	8	144	72	50	54	8.0	53	6.0	67	34.0	72	44.0
	RAND-H-10	160	8	160	72	46	48	4.3	50	8.7	63	37.0	72	56.5
	REAL-1	1	9	2	*2	2	*2	0.0	*2	0.0	*2	0.0	*2	0.0
	REAL-2	1	9	2	*2	2	*2	0.0	*2	0.0	*2	0.0	*2	0.0
	REAL-3	3	9	3	*1	1	*1	0.0	*1	0.0	*1	0.0	*1	0.0
	REAL-4	2	9	4	*4	4	*4	0.0	*4	0.0	*4	0.0	*4	0.0
	REAL-5	5	9	21	*21	21	*21	0.0	*21	0.0	*21	0.0	*21	0.0
	REAL-6	5	9	22	*22	22	*22	0.0	*22	0.0	*22	0.0	*22	0.0
	REAL-7	5	9	23	*23	23	*23	0.0	*23	0.0	*23	0.0	*23	0.0
	REAL-8	7	9	24	*24	24	*24	0.0	*24	0.0	*24	0.0	*24	0.0
	REAL-9	15	9	45	*44	44	44	0.0	*44	0.0	*44	0.0	*44	0.0
	REAL-10	26	9	99	*98	98	98	0.0	*98	0.0	*98	0.0	*98	0.0
	REAL-11	22	9	100	91	87	89	2.3	87	0.0	90	3.4	91	4.6
	REAL-12	32	9	101	*100	97	98	1.0	97	0.0	*100	3.1	99	2.1
	REAL-13	37	9	110	103	97	98	1.0	97	0.0	100	3.1	103	6.2
	REAL-14	28	9	111	*102	99	99	0.0	100	1.0	100	1.0	*102	3.0
Real	REAL-15	35	9	122	110	94	97	3.2	94	0.0	102	8.5	110	17.0
	REAL-16	36	9	123	108	107	107	0.0	108	0.9	108	0.9	108	0.9
	REAL-17	42	9	128	114	103	103	0.0	105	1.9	105	1.9	114	10.7
	REAL-18	31	9	130	121	112	115	2.7	113	0.9	115	2.7	121	8.0
	REAL-19	34	9	131	114	103	107	3.9	103	0.0	108	4.9	114	10.7
	REAL-20	34	9	134	118	106	107	0.9	106	0.0	108	1.9	118	11.3
	REAL-21	39	9	136	119	108	112	3.7	108	0.0	114	5.6	119	10.2
	REAL-22	31	9	138	121	113	117	3.5	113	0.0	117	3.5	121	7.1
	REAL-23	31	9	139	121	113	113	0.0	113	0.0	115	1.8	121	7.1
	REAL-24	37	9	139	110	103	103	0.0	104	1.0	106	2.9	110	6.8
	REAL-25	39	9	139	125	118	118	0.0	121	2.5	121	2.5	125	5.9
	REAL-26	38	9	140	119	107	107	0.0	109	1.9	115	7.5	119	11.2
	REAL-27	35	9	147	129	120	121	0.8	120	0.0	126	5.0	129	7.5
	REAL-28	34	9	151	131	115	116	0.9	115	0.0	121	5.2	131	13.9
	REAL-29	39	9	155	127	117	119	1.7	117	0.0	123	5.1	127	8.5
	REAL-30	41	9	159	131	115	115	0.0	119	3.5	121	5.2	131	13.9

Similar results are observed for the real instances. The scheduling model with the maximal selection search is dominating again. However, the improvement ratio is now up to 17% only. It happens because such real instances are easier to solve compared to the medium and difficult synthetic instances. It shows both the pertinence of the scheduling model and the search framework we introduced.

Finally, we also considered the waiting time minimization (Eq. 11) as a secondary objective using a lexicographical search. However, it yielded only minor improvements regarding the solution obtained using the main objective. It mainly occurs because the value heuristic used already ensures that solutions minimizing the waiting time are tried first.

7 Conclusion and Perspective

In many countries, there is an increasing demand for disabled people requiring health care. Providing a door-to-door transportation to patients minimizing the operational costs while maintaining a sufficient quality of service is still a challenge nowadays. In this context, we introduced the Patient Transportation Problem, which is a specific case of the well-known Dial-a-Ride Problem. This paper proposes a CP approach based on scheduling for solving Patient Transportation Problems. The focus was to design a flexible approach that can easily handle different variants of the problem while being efficient enough to solve real instances. Experimental results have shown that the scheduling models outperforms greedy strategies and successor models often used in classical Vehicles Routing Problems. A generic search strategy maximizing the number of selected requests is also proposed and improves the results.

In practice, Patient Transportation Problems also have a dynamic aspect: new requests, or modification/cancellation of old ones can occur online and a new solution must be found in real time. As future work, we plan to extend our approach in order to deal with such aspects. To do so, we plan to use the CP solution as an initial solution and local search for quickly adapting the solution as modifications are received.

Having discovered recently the approach of Liu et al. [18] developed in parallel with our work, we also wish to investigate experimentally the differences of performances with both models. We also plan to design more advanced LNS relaxations, for instance based on partial order schedules [37]. Lazy clause generation approaches relying on explaining the cumulative constraint [24] may also be worth trying on this problem.

Acknowledgments. This research is financed by the Walloon Region (Belgium) as part of PRESupply Project. The problem has been proposed by the CSD, a Belgian non-profit organization operating at Liège.

References

1. Cordeau, J.F., Laporte, G.: The dial-a-ride problem: models and algorithms. Ann. Oper. Res. **153**, 29–46 (2007)
2. Melachrinoudis, E., Min, H.: A tabu search heuristic for solving the multi-depot, multi-vehicle, double request dial-a-ride problem faced by a healthcare organisation. Int. J. Oper. Res. **10**, 214–239 (2011)
3. Liu, R., Xie, X., Augusto, V., Rodriguez, C.: Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. Eur. J. Oper. Res. **230**, 475–486 (2013)
4. Detti, P., Papalini, F., de Lara, G.Z.M.: A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. Omega **70**, 1–14 (2017)
5. Cordeau, J.F., Laporte, G.: A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp. Res. Part B Methodol. **37**, 579–594 (2003)
6. Parragh, S.N.: Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. Transp. Res. Part C Emerg. Technol. **19**, 912–930 (2011)
7. Parragh, S.N., Cordeau, J.F., Doerner, K.F., Hartl, R.F.: Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. OR Spectr. **34**, 593–633 (2012)
8. Psaraftis, H.N.: An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. Transp. Sci. **17**, 351–357 (1983)
9. Melachrinoudis, E., Ilhan, A.B., Min, H.: A dial-a-ride problem for client transportation in a health-care organization. Comput. Oper. Res. **34**, 742–759 (2007)
10. Cordeau, J.F., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks **30**, 105–119 (1997)
11. Parragh, S.N., Doerner, K.F., Hartl, R.F., Gandibleux, X.: A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. Networks **54**, 227–242 (2009)
12. Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G.: Static pickup and delivery problems: a classification scheme and survey. Top **15**, 1–31 (2007)
13. Attanasio, A., Cordeau, J.F., Ghiani, G., Laporte, G.: Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Comput. **30**, 377–387 (2004)
14. Berbeglia, G., Pesant, G., Rousseau, L.M.: Checking the feasibility of dial-a-ride instances using constraint programming. Transp. Sci. **45**, 399–412 (2011)
15. Berbeglia, G., Cordeau, J.F., Laporte, G.: A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. INFORMS J. Comput. **24**, 343–355 (2012)
16. Parragh, S.N., Schmid, V.: Hybrid column generation and large neighborhood search for the dial-a-ride problem. Comput. Oper. Res. **40**, 490–497 (2013)
17. Jain, S., Van Hentenryck, P.: Large neighborhood search for dial-a-ride problems. In: Lee, J. (ed.) CP 2011. LNCS, vol. 6876, pp. 400–413. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23786-7_31
18. Liu, C., Aleman, D.M., Beck, J.C.: Modelling and solving the senior transportation problem. In: van Hoeve, W.-J. (ed.) CPAIOR 2018. LNCS, vol. 10848, pp. 412–428. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93031-2_30
19. Beldiceanu, N., Carlsson, M., Rampon, J.X.: Global constraint catalog, (revision a) (2012)

20. Gay, S., Hartert, R., Schaus, P.: Simple and scalable time-table filtering for the cumulative constraint. In: Pesant, G. (ed.) CP 2015. LNCS, vol. 9255, pp. 149–157. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23219-5_11
21. Vilím, P.: Timetable edge finding filtering algorithm for discrete cumulative resources. In: Achterberg, T., Beck, J.C. (eds.) CPAIOR 2011. LNCS, vol. 6697, pp. 230–245. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21311-3_22
22. Gay, S., Hartert, R., Schaus, P.: Time-table disjunctive reasoning for the cumulative constraint. In: Michel, L. (ed.) CPAIOR 2015. LNCS, vol. 9075, pp. 157–172. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18008-3_11
23. Ouellet, P., Quimper, C.-G.: Time-table extended-edge-finding for the cumulative constraint. In: Schulte, C. (ed.) CP 2013. LNCS, vol. 8124, pp. 562–577. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40627-0_42
24. Schutt, A., Feydy, T., Stuckey, P.J., Wallace, M.G.: Explaining the cumulative propagator. Constraints **16**, 250–282 (2011)
25. Simonis, H., Cornelissens, T.: Modelling producer/consumer constraints. In: Montanari, U., Rossi, F. (eds.) CP 1995. LNCS, vol. 976, pp. 449–462. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60299-2_27
26. Laborie, P., Rogerie, J.: Reasoning with conditional time-intervals. In: FLAIRS Conference, pp. 555–560 (2008)
27. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: Reasoning with conditional time-intervals. Part II: an algebraical model for resources. In: FLAIRS Conference, pp. 201–206 (2009)
28. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: IBM ILOG CP optimizer for scheduling. Constraints, 1–41 (2018)
29. Dejemeppe, C., Van Cauwelaert, S., Schaus, P.: The unary resource with transition times. In: Pesant, G. (ed.) CP 2015. LNCS, vol. 9255, pp. 89–104. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23219-5_7
30. Ngatchou, P., Zarei, A., El-Sharkawi, A.: Pareto multi objective optimization. In: 2005 Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems, pp. 84–91. IEEE (2005)
31. Gay, S., Hartert, R., Lecoutre, C., Schaus, P.: Conflict ordering search for scheduling problems. In: Pesant, G. (ed.) CP 2015. LNCS, vol. 9255, pp. 140–148. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23219-5_10
32. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: Maher, M., Puget, J.-F. (eds.) CP 1998. LNCS, vol. 1520, pp. 417–431. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49481-2_30
33. Lauriere, J.L.: A language and a program for stating and solving combinatorial problems. Artif. Intell. **10**, 29–127 (1978)
34. Vilím, P., Laborie, P., Shaw, P.: Failure-directed search for constraint-based scheduling. In: Michel, L. (ed.) CPAIOR 2015. LNCS, vol. 9075, pp. 437–453. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18008-3_30
35. OscaR Team: OscaR: Scala in OR (2012). https://bitbucket.org/oscarlib/oscar
36. Thomas, C., Cappart, Q., Schaus, P., Rousseau, L.M.: CSPLib problem 082: Patient transportation problem. http://www.csplib.org/Problems/prob082
37. Godard, D., Laborie, P., Nuijten, W.: Randomized large neighborhood search for cumulative scheduling. ICAPS **5**, 81–89 (2005)