The prisoner transportation problem

Jan Christiaens, Hatice Çalık, Tony Wauters, Reshma Chirayil Chandrasekharan, Greet Vanden Berghe

KU Leuven, Department of Computer Science, CODeS, Belgium

Abstract

Prisoners often require transportation to and from services such as hospital appointments, court proceedings and family visits during their imprisonment. Organising daily prisoners transportation consumes a huge amount of resources. A large fleet of highly protected vehicles, their drivers and security guards must be assigned to all prisoner transports such that all safety and time-related constraints are satisfied while inter-prisoner (inter-passenger) conflicts are avoided. It is beyond human planners' capabilities to minimize costs while attempting to feasibly schedule all prisoner transportation requests. Whereas the prisoner transportation problem (PTP) bears resemblance with vehicle routing, common software systems for vehicle routing fail to address the intricacies associated with the PTP. A dedicated decision support system is required to both support human planners as well as reduce operational costs. The considerable computational challenge due to problem-specific components (inter-passenger conflicts and simultaneous servicing) also makes the PTP interesting from an academic point of view. We formally introduce the problem by providing mixed integer programming models. We implement exact iterative procedures to solve these formulations and evaluate their performance on small instances. In order to solve instances of a realistic size, we present a heuristic. Academic PTP instances generated and employed for experimentation are made publicly available with a view towards encouraging further follow-up research. The heuristic presented in this paper provides all the necessary components to solve the PTP adequately and sets initial benchmarks for the new public instance set.

Keywords: Vehicle routing, prisoner transportation problem, simultaneous servicing, inter-passenger conflicts, multi-compartment

1. Introduction

This paper introduces the Prisoner Transportation Problem (PTP) to the academic community. The PTP, while being formally introduced by the present paper for the first time, has previously been alluded to within operational research literature on occasion. In most cases, however, the problem constitutes a mere academic footnote without any meaningful or rigorous elaboration, for example by Partyka and Hall (2000). In essence, the problem concerns the transportation

Email addresses: `jan.christiaens@cs.kuleuven.be` (Jan Christiaens), `hatice.calik@kuleuven.be` (Hatice Çalık)

of prisoners to and from various destinations - such as hospitals and courts - in special prisoner transportation vehicles which are subdivided into smaller individual compartments.

The PTP constitutes a variant of the commonplace vehicle routing problem (VRP), but more specifically the dial-a-ride problem (DARP). Two unique components make it a particularly interesting problem from a research point of view. The first one is simultaneous servicing, which enables multiple prisoners to embark or disembark simultaneously rather than sequentially. This is unlike the standard practice throughout most vehicle routing and delivery problems. The second component concerns inter-passenger conflicts, which may occur either at compartment or vehicle level. A compartment conflict indicates two prisoners who are not permitted to simultaneously occupy the same compartment, whereas a vehicle conflict denotes that they may not be transported in the same vehicle.

The PTP is a complicated problem and one which is currently solved manually by groups of planners who formulate schedules for the following day. These planners are expected to incorporate many safety measures while constructing such schedules. The real world case which inspired this research required up to three employees with certain skill combinations to transport a single prisoner. There are seven vehicle types, each of which has a different compartment layout and safety level. The vast majority of vehicles, located at one of thirteen depots, are of two types. The first is equipped with one compartment containing three seats and two compartments containing two seats, while the second type is equipped with only one three-seat and one four-seat compartment. After delivery of a prisoner within their time window, the vehicle may either travel onwards or wait on-site for its next service to begin. When certain prisoners are on board, it may be forbidden to wait on-site. Finally, maximum trip lengths should be respected for each individual prisoner. Given that many hundreds of prisoners require transportation on a daily basis, these planners employ technically unnecessary constraints, such as requiring vehicles to only carry prisoners of the same category for the entire day. These constraints make the problem more manageable for human planners, but are not necessary from a computational perspective.

While the PTP brings together a myriad of features and constraints, many of these are commonplace throughout other VRP or pickup-and-delivery problems. Therefore, in the interest of stimulating further research concerning the PTP, only those constraints which are the most characterizing for the PTP are maintained.

This paper first reviews related literature and indicates similarities and differences with respect to the PTP in Section 2. Section 3 formally describes the PTP with a mathematical programming formulation, while Section 4 adapts this formulation to several variants of the PTP. It also includes a computational analysis using these models and an iterative procedure to solve small size problem instances. Section 5 then provides a global overview of the contributing heuristic algorithm, while Sections 6 and 7 describe how inter-passenger conflicts and time-related restrictions are handled. Section 8 describes instances and discusses experimental results and, finally, Section 9 provides concluding remarks and outline some possible future research directions.

2. Related problems

Given that the PTP concerns the transportation of people, as opposed to goods or commodities, it is associated with some uncommon constraints. Despite the vast number of published VRP models and algorithms, the PTP is unsolvable with existing approaches. Indeed, due to the novel constraints outlined in the introduction, the PTP generalises both the VRP and the DARP. When considering *passenger transportation*, dial-a-ride (Cordeau and Laporte, 2007) and the handicapped persons transportation problem (Toth and Vigo, 1997) provide inspiring foundations, albeit inter-passenger conflicts are still absent in these studies.

The PTP differs from the DARP in that inter-passenger conflicts must be respected by preventing 'conflicting' prisoners from sharing a vehicle or compartment. Incorporating this constraint dramatically increases computational complexity compared to existing passenger transportation problems. The DARP variant studied by Beaudry et al. (2010) considers the transportation of patients who need to be transported in isolation as well as those who can be safely transported with others. Isolated patients can be considered as a special case of the conflicts present in the PTP where certain requests have vehicle conflicts with all other requests. This dedication of vehicles to a single patient can also be found in the study by Parragh et al. (2012). While Molenbruch et al. (2017) introduce a DARP with restrictions similar to the PTP's vehicle conflicts, they do not consider compartments, and hence no compartment conflicts, in their study. Product incompatibility restrictions within compartments are considered in the multi-compartment VRP (MCVRP). The three-index mixed integer programming formulation provided by Derigs et al. (2011) can be adapted to several variants of the MCVRP. However, it is not straightforward to modify this formulation to solve the PTP, primarily due to the pickup and delivery aspect of the PTP, which allows for the transportation of conflicting prisoners in the same compartment at different legs of a vehicle trip. This possibly necessitates decision variables with four or more indices. Such a formulation is provided by Lahyani et al. (2015) for a multi-product, multi-period MCVRP to restrict the number of different product types in a compartment to one at any period. From a modeling point of view, a vehicle trip at a period in this MCVRP variant can be considered as one leg of a vehicle trip in the PTP. Note that this type of compartment restriction remains a special case of the one in the PTP where every prisoner (or a group of prisoners) conflicts with every other prisoner (or every other group of prisoners) at the compartment level. Another VRP variant studied by both Oppen and Løkketangen (2008) and Oppen et al. (2010) requires certain animal types to be held in different compartments. However, since the animal types considered in these papers have dedicated compartments, no conflict can occur within compartments. The same characteristics appears in other MCVRP variants which dedicate separate compartments to conflicting goods.

A second significant difference between the PTP and other related problems concerns the way in which service times must be handled. Prisoner transportation enables the simultaneous execution of multiple services at a single location, for example (dis)embarking multiple prisoners at once. In such a case, the total service time at a location must be greater than or equal to the maximum of those individual service times. The combined service times encountered in batching machines scheduling (Potts and Kovalyov, 2000) are similar. Indeed, a batching machine's service time equals the maximum service time of all items in the batch. However, Potts and Kovalyov (2000) do not consider any time windows in their schedules.

The DARP, meanwhile, assumes no more than one service at a time. On-site service times may be categorized as deterministic, stochastic, unknown or dependent according to Eksioglu et al. (2009). Their taxonomic review and recent classification of VRPs never consider multiple services per location. Nevertheless, many situations of this nature do occur in practice. For example, services may be conducted sequentially, and thus their execution order becomes important. As such, one on-site execution sequence may prove infeasible due to time window violations whereas another may introduce unnecessary waiting times. Parragh et al. (2015) introduce the *DARP with split requests and profits*, where group requests may be split across vehicles while service times are assumed independent of the number of people being picked-up or delivered. Although the problem is somewhat similar to the PTP, the model proposed by Parragh et al. (2015) does not accommodate the simultaneous servicing of passengers whose time windows are possibly different. Desaulniers (2010) assumes service times to be quantity-independent in the *split delivery VRP with time windows (SDVRPTW)*. Salani and Vacca (2011) introduce the *discrete* SDVRPTW which assumes delivery-dependent service times. Although uncommon in academic papers, this model is very appropriate in real-world logistics. Service times may, for example, equal the sum of a delivery setup time and a duration proportional to the number of items delivered. In the context of prisoner transportation, the critical period during which prisoners (dis)embark must be minimized. The transportation organization consequently assigns sufficient staff and resources to accommodate the handling of each prisoner's service independently. Multiple on-site services are therefore executed at once, provided the services' time windows are respected.

From a fleet perspective, the PTP also generalizes the multi-compartment vehicle routing problem (MCVRP) (Fallahi et al., 2008). The MCVRP deals with specific product types, each having a corresponding dedicated compartment, possibly of flexible sizes (Henke et al., 2015). Reverse logistics, which exhibits conflicts at the product type level, represents an important application domain for MCVRP. The PTP, by contrast, expresses conflicts at an individual level. Additionally, prisoners may occupy any compartment provided there are no conflicts with other prisoners occupying that compartment at the same time. Paraskevopoulos et al. (2017) indicate the existence of multiple-compartment vehicles for transporting non-mixable products. They explicitly mention that research regarding heterogeneous vehicle types and multiple products is largely absent.

When comparing the PTP to modeling knowledge available in the literature, both inter-passenger conflicts and simultaneous servicing are new and provide interesting challenges to academia.

3. Notation and problem formulation

Let $G = (N, A)$ be a network such as that depicted in Figure 1, with arc set A and node set $N = U \cup I$ where U represents the set of depot nodes and $I = I^P \cup I^D = \{1, \ldots, 2n\}$ represents the set of service nodes with $I^P = \{1, \ldots, n\}$ corresponding to pickup nodes and $I^D = \{n+1, \ldots, 2n\}$ denoting delivery nodes. The set of depots U is a union of two partitions of identical size: $U^1 = \{u_1, \ldots, u_k\}$ and $U^2 = \{u_1^e, \ldots, u_k^e\}$ where U^1 denotes the set of original depots as the starting depots of vehicles and where U^2 contains their duplicates as the ending depots. Thus, for any node $u \in U^1$, U^2 contains one and only one duplicate node u^e. Moreover, $A = A^1 \cup A^2 \cup A^3$ where $A^1 = \{(i,j) : i \in U^1, j \in I\}$, $A^2 = \{(j,i) : i \in U^2, j \in I\}$ and $A^3 = \{(i,j) : i,j \in I : i \neq j, i \neq j+n\}$.

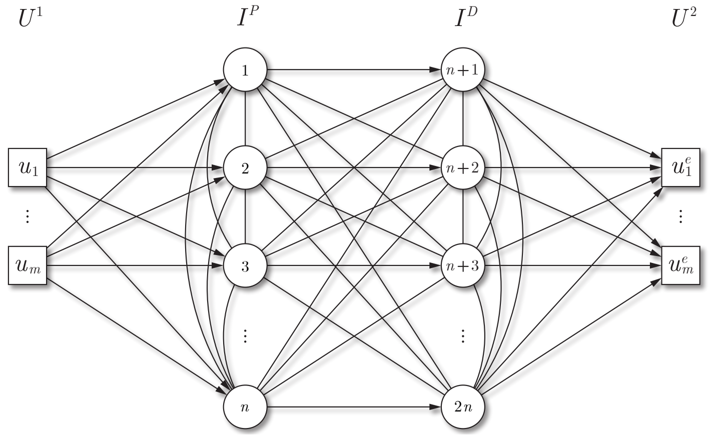

Figure 1: Underlying network representation for the PTP formulations.

Define P as the set of prisoners and R as the set of all transportation requests. Each request corresponds to a node tuple $(i, n+i)$ such that $1 \leq i \leq n$. Each node $i \in I$ associated with request $r(i) = r(n+i) \in R$ of prisoner $p(i) = p(n+i) \in P$ has a service time s_i and a time window $[t_i^-, t_i^+]$. Each service must be completed within its time window. Note that it is possible for an individual prisoner to have multiple requests and multiple services (nodes) may take place at the same physical location. Let L be the set of all distinct physical locations of the nodes in N and $l(i) \in L$ denote the location of $i \in I$, then the length e_{ij} of arc $(i,j) \in A$ is equal to the travel time from $l(i) \in L$ to $l(j) \in L$, with $e_{ij} = 0$ if $l(i) = l(j)$. We further define $t_{max}^+ = \max_{i \in I^D} t_i^+$ and $t_{max} = t_{max}^+ + \max_{(i,j) \in A^2} e_{ij}$ to be used in the formulations.

Multiple services occurring at the same location may be executed simultaneously. However, partial, or even the complete, simultaneous execution of services depends on the degree to which their respective time windows overlap. In a general setting, we denote $M = \{(i,j) : i,j \in R\}$ as the set of request pairs which can be served simultaneously.

Let $V = \{1, \ldots, |V|\}$ denote the set of vehicles and $u_v, u_v^e \in U$ denote the starting and ending depot nodes of $v \in V$. All vehicles are of the same type and each vehicle v is defined by a subset C_v of compartments C. To be more explicit, $C = \{c_1, c_2, \ldots, c_k, c_{k+1}, \ldots, c_{2k}, \ldots, c_{(|V|-1)k+1}, \ldots, c_{|V|k}\}$

and $C_v = \{c_{(v-1)k+1}, \dots, c_{(v-1)k+k}\}$. Each compartment $c \in C$ has a capacity of q_c and belongs to vehicle $v(c) \in V$. Figure 2 illustrates the notation for vehicle compartments. Let H^c and H^v be the sets of all pairs of prisoners who may not simultaneously occupy the same compartment and who may not travel in the same vehicle, respectively. Analogous to H^c and H^v, we further introduce $R^c = \{(r(i), r(j)) : (p(i), p(j)) \in H^c\}$ and $R^v = \{(r(i), r(j)) : (p(i), p(j)) \in H^v\}$ as the conflict sets at the request level.

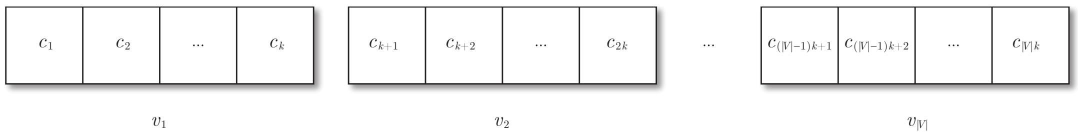

Figure 2: The notation for vehicle compartments.

The trip length for any request r, which begins when the prisoner is embarked and ends immediately before they disembark, should not exceed a certain time threshold L_r. Finally, the duration of a vehicle's trip may be no longer than D_{max}. The objective of the problem is to serve all prisoners while minimizing the total time spent by the vehicles.

We introduce three formulations which are slightly different from each other. The computational performances of these models are quite similar and none of them is clearly dominant over the others. In this section we present only the one that gives the best average solution values and keep the other formulations in Appendix A and Appendix B since the best solution values are obtained from the others for some of the instances.

3.1. A mixed integer programming formulation (FPTP)

We define the following decision variables: $z_{ij}^v = 1$ if vehicle v traverses arc $(i,j) \in A$, 0 otherwise; $x_{ij}^{rc} = 1$ if the prisoner associated with request r is carried in compartment c on arc $(i,j) \in A^3$, 0 otherwise; $y_{ij}^v = 1$ if node i precedes node j (not necessarily immediately) on the trip of vehicle v, 0 otherwise; τ_i^v is the departure time at node $i \in N$ by vehicle $v \in V$; and δ_i^v is the merged service time at node $i \in I$ by vehicle $v \in V$. The following is a mixed integer programming formulation for the PTP, which we will refer to as FPTP.

$$\min \sum_{v \in V} (\tau_{u_v^e}^v - \tau_{u_v}^v) \tag{1}$$

$$\text{s.t.} \sum_{j \in I^P} z_{u_v j}^v \leq 1 \qquad \forall v \in V \tag{2}$$

$$\sum_{j \in I^P} z_{u_v j}^v = \sum_{j \in I^D} z_{j u_v^e}^v \qquad \forall v \in V \tag{3}$$

$$\sum_{j \in N} z_{ij}^v = \sum_{j \in N} z_{ji}^v \qquad \forall v \in V, i \in I \tag{4}$$

$$\sum_{v \in V} \sum_{j \in I} z_{ij}^v = 1 \qquad \forall i \in I^P \tag{5}$$

$$\sum_{c \in C_v} x_{ij}^{rc} \leq z_{ij}^{v} \qquad \forall v \in V, (i,j) \in A^3, r \in R \tag{6}$$

$$\sum_{j \in I} \sum_{c \in C} x_{ij}^{r(i)c} = 1 \qquad \forall i \in I^P \tag{7}$$

$$\sum_{j \in I} x_{ij}^{r(i)c} = \sum_{j \in I} x_{j(i+n)}^{r(i)c} \qquad \forall i \in I^P, c \in C \tag{8}$$

$$\sum_{j \in I} x_{ij}^{rc} = \sum_{j \in I} x_{ji}^{rc} \qquad \forall r \in R, i \in I : r \neq r(i), c \in C \tag{9}$$

$$x_{ij}^{rc} + x_{ij}^{r(i)c} \leq 1 \qquad \forall (r, r(i)) \in R^c, (i,j) \in A^3, c \in C \tag{10}$$

$$\sum_{c \in C_v} x_{ij}^{rc} + \sum_{c \in C_v} x_{ij}^{r(i)c} \leq 1 \qquad \forall (r, r(i)) \in R^v, (i,j) \in A^3, v \in V \tag{11}$$

$$\sum_{r \in R} x_{ij}^{rc} \leq q_c z_{ij}^{v(c)} \qquad \forall (i,j) \in A^3, c \in C \tag{12}$$

$$\tau_i^v + e_{ij} + \delta_j^v \leq \tau_j^v + t_{max}(1 - z_{ij}^v) \qquad \forall v \in V, (i,j) \in A : l(i) \neq l(j) \tag{13}$$

$$\tau_i^v \leq \tau_j^v + t_i^+(1 - z_{ij}^v) \qquad \forall v \in V, (i,j) \in A^3 : l(i) = l(j) \tag{14}$$

$$\delta_i^v \geq \sum_{j \in I : l(i) = l(j)} s_j z_{ij}^v \qquad \forall v \in V, i \in I \tag{15}$$

$$\delta_j^v \geq \sum_{i \in I : l(i) = l(j)} s_i z_{ij}^v \qquad \forall v \in V, j \in I \tag{16}$$

$$\tau_{n+i}^v - \tau_i^v - s_{i+n} \leq L_{r(i)} \qquad \forall v \in V, i \in I^P \tag{17}$$

$$\tau_{u_v^e}^v - \tau_{u_v}^v \leq D_{max} \qquad \forall v \in V \tag{18}$$

$$\begin{aligned}\tau_{u_v^e}^v - \tau_{u_v}^v \geq &\sum_{i \in I^P} (e_{u_v i} + s_i) z_{u_v i}^v \\ &+ \sum_{i \in I^D} (e_{i u_v^e} + s_i) z_{i u_v^e}^v + \sum_{(i,j) \in A^3} e_{ij} z_{ij}^v \qquad \forall v \in V\end{aligned} \tag{19}$$

$$t_i^- + s_i \leq \tau_i^v \leq t_i^+ \qquad \forall v \in V, i \in I \tag{20}$$

$$s_i \leq \delta_i^v \leq L_{r(i)} \qquad \forall i \in I, v \in V \tag{21}$$

$$z_{ij}^v \leq y_{ij}^v, \qquad \forall (i,j) \in A^3, v \in V \tag{22}$$

$$y_{ij}^v + y_{ji}^v = 1, \qquad \forall (i,j) \in A^3, v \in V \tag{23}$$

$$y_{ij}^v + y_{jl}^v + y_{li}^v \leq 2 \qquad \forall (i,j), (j,l), (l,i) \in A^3, v \in V \tag{24}$$

$$y_{ij}^v \in \{0,1\} \qquad \forall v \in V, (i,j) \in A \tag{25}$$

$$z_{ij}^v \in \{0,1\} \qquad \forall v \in V, (i,j) \in A \tag{26}$$

$$x_{ij}^{rc} \in \{0,1\} \qquad \forall r \in R, c \in C, (i,j) \in A^3 \tag{27}$$

The objective function (1) minimizes the total time spent by the vehicles. Constraints (2) limit the number of outgoing arcs from the depot of any vehicle to one whereas Constraints (3) equal this number to the number of incoming arcs to its duplicate depot node. For any intermediate node, Constraints (4) ensure that every vehicle which enters also leaves that node. Constraints (5) ensure that each pickup is served by a vehicle. Constraints (6) prevent requests to traverse an arc in a compartment if the vehicle associated with this compartment is not traversing that arc. While Constraints (7) assign a compartment to each request when leaving its pickup node, Constraints (8) and (9) make sure that the request is in the same compartment when entering its delivery node as well as any intermediate nodes, respectively. Constraints (10) and (11) prevent request assignments causing compartment and vehicle conflicts, respectively. The capacity restrictions of compartments are enforced by Constraints (12). Since simultaneously servicing distinct requests at the same location is possible, Constraints (13) update the time labels of consecutively visited nodes of distinct locations. When the locations of two consecutively visited nodes are identical, Constraints (14) ensure that the departure time at the preceding node is less than or equal to the following node's departure time. Constraints (15) and (16) ensure that the merged service time of the nodes which are being simultaneously serviced is no less than the maximum of their individual service times. Constraints (17) and (18) impose the maximum travel time for requests and for vehicles, respectively. Constraints (19) require the total time spent by a vehicle to be greater than or equal to the total traveling time plus the total service time of the nodes visited by it. By Constraints (20), the departure time at each node is limited to be no earlier than the earliest departure time plus the service time and no later than the latest departure time. Constraints (21) restrict merged service time at a node to be greater than or equal to its service time and less than or equal to the maximum traveling time of the associated passenger. We eliminate sub-tours via Constraints (22)-(24), which are known to provide tight bounds (Öncan et al., 2009). Finally, Constraints (25)-(27) are binary restrictions.

3.2. Symmetry breaking constraints and additional valid inequalities

When vehicles are identical, Constraints (28) break the symmetry among the vehicles of any depot. Let v^{1*} denote the smallest index vehicle of the closest depot to the smallest index pickup node, Constraint (29) ensures that this pickup node is served by v^{1*}. For any identical pair of compartments in the same vehicle, Constraints (30) assign compartments to new requests in lexicographic order of compartment indices if both are empty.

$$\sum_{j \in I^P} z_{uj}^{v_2} \leq \sum_{j \in I^P} z_{uj}^{v_1} \qquad \forall v_1 < v_2 \in V : u = u_{v_1} = u_{v_2} \tag{28}$$

$$\sum_{(1,j) \in A^3} z_{1j}^{v^{1*}} \geq 1 \tag{29}$$

$$\sum_{(i,j) \in A^3} x_{ij}^{r(i)c_2} \leq \sum_{(i,j) \in A^3} \sum_{r \neq r(i)} (x_{ij}^{rc_1} + x_{ij}^{rc_2}) \qquad \forall i \in I^P,\ c_1, c_2 \in C : c_1 < c_2,\ v(c_1) = v(c_2),\ q_{c_1} = q_{c_2} \tag{30}$$

Although Constraints (31)-(33) are implied by Constraints (2)-(5), their inclusion improves the performance of the formulation when solving with CPLEX.

$$\sum_{v\in V}\sum_{j\in U^1\cup I} z_{ji}^v = 1 \qquad \forall i \in I^P \tag{31}$$

$$\sum_{v\in V}\sum_{j\in I} z_{ji}^v = 1 \qquad \forall i \in I^D \tag{32}$$

$$\sum_{v\in V}\sum_{j\in I\cup U^2} z_{ij}^v = 1 \qquad \forall i \in I^D \tag{33}$$

Constraints (34) and (35) are tighter than Constraints (13) for restricting the departure times at the depot nodes when the triangular inequality holds for $e_{ij}, (i,j) \in A$.

$$\tau_{u_v}^v + e_{u_v i} + \delta_i \leq \tau_i^v + t_{max}^+(1 - \sum_{j\in I} z_{ij}^v) \qquad \forall v \in V, i \in I \tag{34}$$

$$\tau_i^v + e_{iu_v^e} \leq \tau_{u_v^e}^v + t_{max}(1 - \sum_{j\in I} z_{ji}^v) \qquad \forall v \in V, i \in I \tag{35}$$

4. Modeling and testing the variants and special cases of the PTP

Heterogeneous fleets and passenger-vehicle compatibility restrictions: The formulations presented in Section 3, Appendix A and Appendix B solve the generalization of the PTP with a heterogeneous fleet of vehicles. When certain passengers can only be transported by a subset of vehicles, one can easily adapt these formulations by defining $V_i \subset V$ as the set of vehicles which can transport the passenger associated with node $i \in I$.

Minimizing the total routing cost: A common objective function used in the vehicle routing literature is to minimize the total routing cost, which is usually proportional to the total distance traveled. Let f_{ij}^v be the cost of traversing arc $(i,j) \in A$ with vehicle $v \in V$. The PTP to minimize the total routing cost can be solved by replacing the objective function with (36):

$$\min \sum_{v\in V}\sum_{(i,j)\in A} f_{ij}^v z_{ij}^v. \tag{36}$$

No simultaneous servicing (NSS): When $M = \emptyset$ (no passenger pair may be served simultaneously), the problem becomes a generalization of the DARP with passenger conflicts. To model this general case, we exclude $\boldsymbol{\delta}$ variables, the constraints including those variables as well as Constraints (14) from FPTP. We then add (37) to update the departure times of consecutively visited nodes.

$$\tau_i^v + e_{ij} + s_j \leq \tau_j^v + t_{max}(1 - z_{ij}^v) \qquad \forall v \in V, (i,j) \in A \tag{37}$$

No passenger conflicts (NC): In order to formulate the special case of the PTP where there are no passenger conflicts ($R^C = \emptyset = H$), the compartment index of the $\boldsymbol{x}$ variables can be replaced with a single vehicle index. In this case, $x_{ij}^{rv} = 1$ if the passenger associated with request r is carried in vehicle v on arc $(i, j) \in A^3$, and 0 otherwise. While Constraints (10) and (11) are no more relevant and can therefore be omitted, Constraints (6)-(12) must be replaced with (38)-(42).

$$x_{ij}^{rv} \le z_{ij}^{v} \qquad \forall v \in V, (i,j) \in A^3, r \in R \tag{38}$$

$$\sum_{j \in I} \sum_{v \in V} x_{ij}^{r(i)v} = 1 \qquad \forall i \in I^P \tag{39}$$

$$\sum_{j \in I} x_{ij}^{r(i)v} = \sum_{j \in I} x_{j(i+n)}^{r(i)v} \qquad \forall i \in I^P, v \in V \tag{40}$$

$$\sum_{j \in I} x_{ij}^{rv} = \sum_{j \in I} x_{ji}^{rv} \qquad \forall r \in R, i \in I : r \neq r(i), v \in V \tag{41}$$

$$\sum_{r \in R} x_{ij}^{rv} \le \sum_{c \in C : v(c) = v} q_c z_{ij}^{v} \qquad \forall (i,j) \in A^3, v \in V \tag{42}$$

The dial-a-ride problem: When there are no passenger conflicts ($R^C = \emptyset = H$) and simultaneous servicing is not possible for any pair of prisoners ($M = \emptyset$), the PTP reduces to the DARP with the objective of minimizing the total travel time of the vehicles. For this case, FPTP can be utilized to solve the DARP by employing the aforementioned modifications for the NSS and the NC cases.

The PTP with transfers: An interesting variant of pickup-and-delivery problems involves transferring the load from one vehicle to another at certain transshipment points. This variant is referred to as PDPT in the literature. Although transfers are not considered in this particular PTP application that we study, such a variant might be worthwhile investigating. For modeling the PTP with transfers, we refer the reader to a very elegant study by Rais et al. (2014) that formulates several variants of the PDPT which can easily be combined with the formulations we provide for the PTP.

4.1. Testing the formulations

Although the PTP considers there to be an infinite number of vehicles available, the number of vehicles utilized in any feasible solution cannot be greater than the number of requests. Preliminary experiments indicate that models perform relatively better when $|V| \le 3$ but suffer from slow convergence when $|V| \ge 4$. In order to overcome this difficulty, we implemented an iterative algorithm which utilizes any of these formulations to solve the problem for a fixed number $1 \le k \le |V|$ of vehicles at each iteration k.

Let $F(UB, k)$ be any of the PTP models where $|V| = k$, $\sum_{j \in I^P} z_{u_v j}^{v} = 1, \forall v \in V$ and $UB - \epsilon$ is an upper bound on the objective function value with $\epsilon > 0$ being a sufficiently small number. Let

$\mathbf{\Theta}$ be an optimal solution obtained from $F(UB, k)$ and $f(\mathbf{\Theta})$ be its value, Algorithm 1 details an iterative procedure for solving the PTP to proven optimality.

Algorithm 1: Iterative algorithm (IT)

```
   Input: $n$, $UB \leftarrow \infty$, $k \leftarrow 1$, $\epsilon > 0$
   Output $bestSolution$
1  while $k \leq n$ do
2      Solve $F(UB, k)$
3      if feasible with the optimal solution $\mathbf{\Theta}$ then
4          $UB \leftarrow f(\mathbf{\Theta}) - \epsilon$
5          $bestSolution \leftarrow \mathbf{\Theta}$
6          $k \leftarrow k + 1$
7      else if infeasible then
8          if $UB = \infty$ then
9              $k \leftarrow k + 1$
10         else
11             $k \leftarrow n + 1$
12 return $bestSolution$
```

In order to investigate the impact of simultaneous servicing and conflicts on the performance of the models and the solutions they provide, we conduct tests removing these components one at a time. Let PF be one of the aforementioned formulations for some variant of the PTP, we denote the iterative procedure for solving PF as IT-PF in this section.

Table 1 compares the performance of the PTP and NSS formulations and their iterative procedures on the instances with up to 10 requests (20 service nodes). The 'id' of an instance consists of three numbers in a format $n\_g\_n^C$ where n is the number of requests to serve, g is the group code and n^C is the number of compartments in each vehicle. Among these instances, the first group ($g = 0$) contains a large number of services that can be served simultaneously, none of which conflict with one another. The second group of instances ($g = 1$) has either no or only few services that can be served simultaneously and it contains many conflicting requests. The third group ($g = 2$) has more balanced instances with both mergeable services and conflicting requests. For each instance, the compartment configuration is provided in the second column: 1-1-1-1 indicates four compartments where each has a capacity of a single unit, 2-2 denotes two compartments where each has a capacity of 2, while the final configuration is simply 4 and denotes a single compartment whose capacity is 4.

Table 1 indicates the optimal values in boldface and the best solution values obtained if none of the methods reach proven optimality. Some of these optimal values are obtained by the iterative procedures of other formulations (see Appendix C). Columns '$g\%$' report the CPLEX gap for the compact formulations when they reach the time limit of one hour. If the time limit is reached at any iteration of the algorithms, this is indicated with 'TL' under the solving time column '$t(s)$' and the objective value of the best solution obtained is reported under column 'Obj'. We do not report

Table 1: Comparison of PTP and NSS methods.

			PTP		FPTP			IT-PTP		NSS		FNSS			IT-NSS		PTP (36)	
id	Comp.	n	Best	k	Obj	$t(s)$	$g\%$	Obj	$t(s)$	Best	k	Obj	$t(s)$	$g\%$	Obj	$t(s)$	Best	k
4_0_4	1-1-1-1	4	**54**	1	**54**	9.73	0.00	**54**	3.21	**129**	1	**129**	17.22		**129**	5.97	**9**	1
4_1_4	1-1-1-1		**271**	2	**271**	0.22	0.00	**271**	0.65	**271**	2	**271**	2.32		**271**	0.82	**42**	1
4_1_2	2-2		**271**	2	**271**	0.24	0.00	**271**	17.12	**271**	2	**271**	0.24		**271**	11.38	**42**	1
4_1_1	4		**271**	2	**271**	0.19	0.00	**271**	15.43	**271**	2	**271**	0.16		**271**	8.31	**46**	1
4_2_4	1-1-1-1		**181**	2	**181**	0.62	0.00	**181**	2.04	**226**	2	**226**	3.52		**226**	3.09	**102**	1
4_2_2	2-2		**181**	2	**181**	0.98	0.00	**181**	2.10	**226**	2	**226**	4.07		**226**	1.69	**102**	1
4_2_1	4		**181**	2	**181**	1.25	0.00	**181**	12.54	**226**	2	**226**	1.21		**226**	3.78	**102**	1
6_0_4	1-1-1-1	6	**90**	2	90	TL	93.89	**90**	405.92	**193**	1	**193**	TL	94.82	**193**	3198.74	**13**	1
6_1_4	1-1-1-1		**399**	1	**399**	93.68	0.00	**399**	10.29	**399**	1	**399**	78.55		**399**	8.25	**69**	1
6_1_2	2-2		**399**	1	**399**	5.96	0.00	**399**	11.78	**399**	1	**399**	8.18		**399**	3.49	**69**	1
6_1_1	4		**403**	1	**403**	2.35	0.00	**403**	7.88	**403**	1	**403**	1.82		**403**	1.20	**73**	1
6_2_4	1-1-1-1		**282**	3	**282**	353.12	0.00	**282**	34.49	**342**	3	**342**	1137.91		**342**	135.57	**106**	1
6_2_2	2-2		**282**	3	**282**	130.94	0.00	**282**	13.61	**342**	3	**342**	380.57		**342**	39.69	**106**	1
6_2_1	4		**282**	3	**282**	6.27	0.00	**282**	114.64	**342**	3	**342**	47.34		**342**	38.67	**106**	1
8_0_4	1-1-1-1	8	106	2	126	TL	97.88	106	TL	256	1	256	TL	98.44	256	TL	**15**	1
8_1_4	1-1-1-1		**516**	2	519	TL	60.67	516	TL	**529**	2	536	TL	74.82	529	TL	**83**	2
8_1_2	2-2		**516**	2	516	TL	37.44	**516**	1659.33	**529**	2	529	TL	61.06	**529**	564.73	**83**	2
8_1_1	4		**521**	2	**521**	169.04	0.00	**521**	23.22	**536**	2	**536**	273.47		**536**	17.12	**83**	2
8_2_4	1-1-1-1		**356**	3	393	TL	59.45	356	TL	423	2	423	TL	69.98	423	TL	**177**	1
8_2_2	2-2		**356**	3	356	TL	16.57	**356**	2175.07	**423**	2	423	TL	39.95	**423**	3672.67	**177**	1
8_2_1	4		**356**	3	**356**	2757.98	0.00	**356**	823.36	**423**	2	423	TL	27.23	**423**	1251.62	**177**	1
10_0_4	1-1-1-1	10	106	2	273	TL	100.00	106	TL	331	3	366	TL	100.00	331	TL	20	2
10_1_4	1-1-1-1		599	2	607	TL	75.86	599	TL	629	2	642	TL	75.62	629	TL	**134**	2
10_1_2	2-2		599	2	682	TL	84.90	599	TL	629	2	674	TL	84.72	629	TL	**134**	2
10_1_1	4		**601**	2	603	TL	31.82	**601**	488.14	**631**	2	633	TL	37.60	**631**	436.35	**134**	2
10_2_4	1-1-1-1		451	2	706	TL	76.00	451	TL	541	2	631	TL	73.22	541	TL	227	2
10_2_2	2-2		450	2	531	TL	67.93	450	TL	540	2	581	TL	70.42	540	TL	**227**	2
10_2_1	4		450	2	467	TL	63.09	450	TL	540	2	540	TL	67.12	540	TL	**227**	2

Obj: best solution value; k: number of vehicles used; $t(s)$: solving time in seconds; $g\%$: CPLEX gap; TL: time limit (3600 seconds) is reached.

CPLEX gaps for the iterative algorithms as they are only relevant for the restrictive individual problems requiring exactly k vehicles, but not for the original problem with at most n vehicles. The last two columns of this table provide the results obtained from the PTP where the objective function is replaced with Equation (36) to minimize the total routing cost.

When there are many conflicting passengers, we observe that the compartment configuration matters: the best solution value increases for those instances where vehicles have a single compartment rather than multiple. The optimal values mostly increase as the number of requests gets larger, but the magnitude of the increase is less for $g = 0$ instances. Note that these instances are high in number of requests that can be served simultaneously, with none of them conflicting. This enables serving more requests without incurring any additional travel or service time if the new requests can be combined with the existing ones, as in the case for instances 8_0_4 and 10_0_4.

Comparing the performance of individual PTP methods, we observe that although FPTP exhibits a satisfactory performance on small instances, particularly the ones with 4 requests, it performs poorly on larger instances. On the other hand, using the iterative algorithms we are able to obtain the optimal solutions for all the instances with up to 8 requests, except for 8_0_4. The iterative algorithms only terminate with proven optimality for 10_1_1 among the instances where $n = 10$.

The performance of the NSS methods follows a pattern similar to that of the PTP methods in the sense that the PTP instances which reach the time limit also remain unsolved. In general, the $g = 0$ instances appear the most difficult for all methods. The compact formulations are in particular unable to obtain high quality dual bounds quickly and they need to explore a large number of branch-and-bound nodes. We suspect this is due to a large number of symmetric solutions given that the time windows of most requests in this group largely overlap and there are no conflicts to, at least partially, break the symmetry.

The most interesting observation when comparing the best solution values of the PTP and the NSS is that one can save a considerable amount of time by simultaneously serving requests whenever possible. The gain is particularly significant for those instances which contain a large number of requests convenient for simultaneous servicing.

Table 2 provides the results for the NC and the DARP. Testing different compartment configurations is irrelevant for these problems since no passenger conflicts are present. Therefore, all the instances in this table have vehicles with a single compartment whose capacity is 4. The last two columns of this table report the results obtained from the DARP, where the objective function is replaced with (36) to minimize the total routing cost. When comparing the best solution values of the NC and the PTP, we observe that it is higher for the PTP in all instances that contain conflicting customers, except for 6_2_1 which has a relatively sparse conflict matrix. The best solution values of the PTP are generally lower than the best solution values of the DARP. However, we observe that these values might be higher for the PTP for those instances where there is little to gain by simultaneous servicing and for which passenger conflicts are in effect.

Table 2: Results for NC and DARP methods.

		PTP	NC		FNC			IT-NC		DARP		FDARP			IT-DARP		DARP (36)	
id	n	Best	Best	k	Obj	$t(s)$	$g\%$	Obj	$t(s)$	Best	k	Obj	$t(s)$	$g\%$	Obj	$t(s)$	Best	k
4_0_1	4	54	**54**	1	**54**	2.35		**54**	11.42	**129**	1	**129**	10.28		**129**	2.85	**9**	1
4_1_1		271	**271**	2	**271**	0.27		**271**	4.43	**271**	2	**271**	0.49		**271**	1.26	**42**	1
4_2_1		181	**181**	2	**181**	1.09		**181**	8.50	**226**	2	**226**	0.9		**226**	1.34	**102**	**1**
6_0_1	6	90	**90**	2	**90**	279.78		**90**	119.19	**193**	1	**193**	1947.75		**193**	151.21	**13**	1
6_1_1		403	**399**	1	**399**	14.14		**399**	9.35	**399**	1	**399**	20.12		**399**	2.04	**69**	1
6_2_1		282	**282**	3	**282**	14.77		**282**	66.89	**342**	3	**342**	155.15		**342**	12.54	**106**	1
8_0_1	8	106	106	2	106	TL	96.23	106	TL	256	1	256	TL	98.34	256	TL	**15**	2
8_1_1		521	**514**	**1**	516	TL	31.36	**514**	378.64	**529**	2	536	TL	35.06	**529**	748.25	**83**	2
8_2_1		356	**350**	**3**	352	TL	44.55	**350**	5385.99	414	2	414	TL	71.56	414	TL	**137**	2
10_0_1	10	106	106	1	110	TL	96.36	106	TL	320	3	326	TL	99.39	320	TL	16	2
10_1_1		601	576	3	609	TL	91.20	601	TL	610	2	639	TL	92.04	610	TL	104	2
10_2_1		450	444	2	507	TL	80.19	444	TL	535	2	576	TL	79.35	535	TL	216	2

When comparing the best solution values of the PTP and its variants, we observe that among all the problems included in these tables, the best solution values are always the highest for the NSS. For those which consider inter-passenger conflicts, the least number of instances are solved to optimality for the NSS. The results indicate that it was computationally easier to solve the PTP with Objective (36) than the original PTP on these instances. This is reflected in both the number of instances solved to optimality and the average solving time. A similar pattern is observed when comparing the DARP and the DARP with Objective (36) as well.

The mathematical models introduced in this paper provide formal descriptions for the PTP variants and insights for solving them to optimality. However, the challenging combination of constraints present in these formulations make it difficult to solve real-world instances of the PTP in reasonable amounts of time. As the practitioners need to update their trips on a daily basis, we also introduce a heuristic to solve realistically generated large-scale instances and provide an additional computational analysis on these instances.

5. PTP Heuristic

The heuristic approach introduced for addressing the PTP in this paper utilizes the ruin & recreate approach proposed by Schrimpf et al. (2000), guided by a stochastic Local Search-based metaheuristic (LS). LS is employed given that it is both computationally quick and effective for the capacitated vehicle routing problem and similar other problems (Laporte, 2009). The PTP is accordingly decomposed into a master and a subproblem. The master problem is modeled as a multi depot dial-a-ride problem with simultaneous servicing (Section 7), while the prevention of the inter-passenger conflicts (Section 6) corresponds to the subproblem.

5.1. Constructive heuristic

The initial solution is created by a greedy constructive method. All prisoners are first stored in a list while the solution is completely empty. The resulting unscheduled prisoner list is shuffled before employing *best insertion*. The procedure iterates over all feasible combinations for both pickup and delivery insertions inside the current partial solution and eventually inserts the prisoner's request into the solution at the minimal additional cost. If no such feasible combination exists, such as when none of the vehicles inside the incumbent solution are capable of transporting the prisoner within D_{max}, a new trip is initiated at the depot closest to the relevant prisoner. This best insertion procedure is repeated until all prisoner transports are scheduled.

5.2. Local search improvement

Simulated Annealing (Kirkpatrick et al., 1983) is utilized to guide the local search towards better solutions. The initial temperature is set to 100 and is cooled down to 1 (the final temperature). Both temperatures were empirically determined. The cooling rate α is computed such that the final temperature is reached after one million iterations. Classical local search neighbourhoods such as two-opt* would prove impractical given the PTP's constraints since, for example, timing and allocation feasibilities are both affected when adjusting the order of visits in a trip. Therefore, a more flexible local search heuristic is applied during the improvement iterations. A current solution's neighbour is generated iteratively by a ruin & recreate procedure. This procedure begins from a copy of the current solution and removes its first trip. The removed prisoners are subsequently appended to the unscheduled prisoner list. The ruined solution is afterwards recreated by the same best insertion procedure employed for the initial solution's construction. The resulting neighbour solution is passed to the metaheuristic's acceptance procedure for either rejecting or accepting it as the incumbent solution. Furthermore, a neighbour may only be accepted if it contains either an equal or greater number of prisoners than the current solution. If the neighbour is rejected, the current solution's first trip is moved to the end of the solution's list of trips.

6. Prisoner allocation heuristic

Given a vehicle's trip and a set of prisoners, one should check whether or not all prisoners can be allocated to truck compartments without violating inter-passenger conflicts. Formally, this feasibility problem can be defined as follows: Let $T = \{T_0, \ldots, T_\eta\}$ be the set of transit points of a vehicle's trip. Define P_t as the set of prisoners to be allocated to the compartments of the vehicle at transit point $t \in T$, $P^T = \bigcup_{t \in T} P_t$, and F as the set of conflicting prisoner pairs. The prisoner allocation problem (PAP) checks if a set P^T of prisoners can be feasibly allocated to the compartments of the vehicle such that no conflicting prisoners are in the same compartment at the same time.

The PAP is a generalization of the *Bounded Vertex Coloring Problem* (BVCP). Consider the special case of the PAP where all compartments have identical capacities, say q, and all prisoners of a vehicle trip T are picked up at the transit point T_1 and delivered to transit point T_2. The latter implies that all prisoners will be traveling in the vehicle between points T_1 and T_2. This special case of the PAP reduces to the BVCP which is known to be NP-Complete on general graphs with $q \geq 3$ (Hansen et al., 1993).

In order to formulate the PAP, define a binary decision variable W_{pc} which equals 1 if prisoner $p \in P^T$ is assigned to compartment $c \in C$, and 0 otherwise. Equations (43)-(46) provide an integer programming model for the PAP. Constraints (43) guarantee that each prisoner in P^T is assigned to exactly one compartment in C. Constraints (44) ensure that the number of assigned prisoners to a compartment c never exceeds its capacity q_c, while Constraints (45) ensure no conflicting prisoners are assigned to the same compartment. Note that set F only contains those conflicting pairs of prisoners whose presence in the vehicle overlaps in time.

$$(FPAP) \qquad \sum_{c \in C} W_{p,c} = 1 \qquad \forall\, p \in P^T \tag{43}$$

$$\sum_{p \in P_t} W_{p,c} \leq q_c \qquad \forall\, c \in C, t \in T \tag{44}$$

$$W_{p1,c} + W_{p2,c} \leq 1 \qquad \forall\, (p1, p2) \in F, c \in C \tag{45}$$

$$W_{p,c} \in \{0, 1\} \qquad \forall\, p \in P^T, c \in C \tag{46}$$

This feasibility check must be conducted very frequently within the PTP heuristic. The fact that FPAP requires a relatively lengthy amount of time encourages us to employ a fast greedy approach for solving the PAP. Consider an iterative approach where prisoners are added to a trip one by one. The feasibility check implies that, for each prisoner insertion evaluation, a feasible prisoner-truck allocation must be achieved during the complete trip.

Figure 3 illustrates an example trip which stops at six transit points while transporting four prisoners. Figure 3(a) shows the sequence of the trip's transit points and prisoner transports a, b, c and d via dashed arrows, whereas Figure 3(b) indicates the vehicle's occupation throughout the trip.

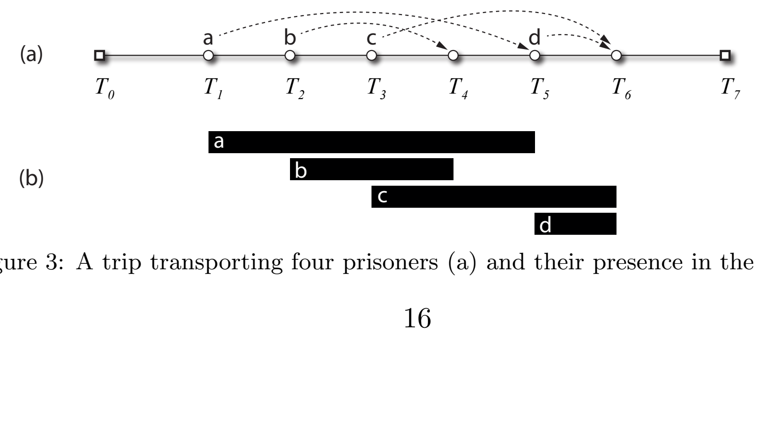

Figure 3: A trip transporting four prisoners (a) and their presence in the truck (b).

Figure 4(a) illustrates a vehicle with two compartments with (b) corresponding to a graph H^c wherein an edge indicates an inter-passenger conflict. It is noteworthy that while prisoners b and d may not share the same compartment, this edge is irrelevant given that the presence of these two prisoners in the vehicle does not overlap in time. Therefore this edge may be removed from the graph, resulting in (c). In order to find a feasible prisoner allocation, one may consider a graph coloring problem wherein the number of an individual color's assignments is bound to a compartment's capacity at each transit point.

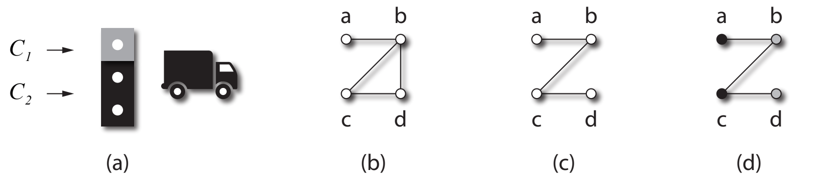

Figure 4: Prisoner allocation and conflict graphs.

Figure 4(d) provides a feasible allocation: Prisoners a and c are assigned to compartment 2 and prisoners b and d to compartment 1. Notice that while the color for compartment 1 is employed more than its corresponding capacity, it never exceeds this capacity at any single transit point as illustrated in Figure 5.

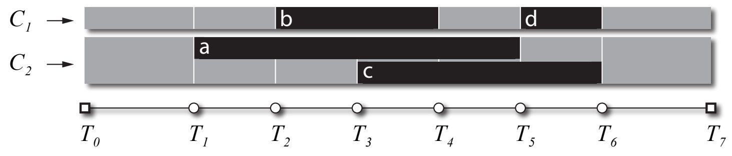

Figure 5: Solution for allocating prisoners.

6.1. Greedy Prisoner Allocation Heuristic

This section introduces a greedy prisoner allocation heuristic (PAH) to assign prisoners to compartments. The algorithm is based on a graph coloring heuristic proposed by Al-Omari and Sabri (2006). The pseudocode is provided by Algorithm 2. All prisoners which are transported by trip T are included in set P^T. These prisoners are sequentially assigned to one of the vehicle's compartments. To begin, the prisoner p with the fewest feasible compartment allocations is selected from P^T (lines 3-11). If this number of compartments is the same for two prisoners, then the prisoner with the greatest number of conflicts (the number of incident edges in Figure 4 (c)) is selected. Next, the compartment with the greatest number of free seats is selected from the set of all compartments with no conflicting prisoners for p (lines 12-21). If prisoner p cannot be assigned to any compartment, the heuristic fails to find a feasible assignment for the trip.

Although Algorithm 2 does not guarantee finding a feasible allocation (should one exist), the extensive experimental evaluations conducted indicate its effectiveness. In these experiments, the PAH is tested on a set of randomly generated prisoner allocation instances, which are proven to

Algorithm 2: Prisoner Allocation Heuristic

```
Input: Trip $T$ and vehicle $V$
AllocatePrisoners($T$, $V$)
1   $P^T \leftarrow$ prisoners transported by $T$
2   while $P^T \neq \emptyset$ do
        // select the next prisoner to allocate
3       $p \leftarrow null$
4       foreach $p' \in P^T$ do
5           if $p = null$ then
6               $p \leftarrow p'$
7           else if $\#nonConflictingComps(p') < \#nonConflictingComps(p)$ then
8               $p \leftarrow p'$
9           else if $\#nonConflictingComps(p') = \#nonConflictingComps(p)$ then
10              if $\#conflicts(p') > \#conflicts(p)$ then
11                  $p \leftarrow p'$
        // select a compartment for $p$
12      $c \leftarrow null$
13      foreach $c' \in nonConflictingComps(p)$ do
14          if $c = null$ then
15              $c \leftarrow c'$
16          else if $\#freeSeats(c') < \#freeSeats(c)$ then
17              $c \leftarrow c'$
18      if $c \neq null$ then
19          assign $p$ to $c$
20      else
21          return $false$
22  return $true$
```

be feasible using the FPAP. Vehicles contain up to 10 compartments, with the number of seats within each individual compartment between 1 and 10. The number of transported prisoners is 50-100% of the total number of seats in the vehicle. The fraction of conflicting pairs is selected from between 30% and 100% of all pairs. Finally, the number of transit points is fixed to 10 while the number of transit points at which a prisoner can be present in the vehicle is at least 1 and at most 7.

Among 19581 PAP instances mathematically proven to be feasible, the PAH was able to generate a feasible solution for 18949, corresponding to a success rate of 96.7%. Given its efficiency, the greedy PAH was taken to be sufficiently effective for employment within the PTP heuristic.

7. Accommodating time constraints

When composing or evaluating a trip, intelligent time constraint handling is indispensable. This section details how the PTP's time constraints are employed for constructing feasible trips. The discussion is presented for a single vehicle trip and, for simplicity reasons, vehicle indices are dropped from the notations and equations.

7.1. Time windows

Let k be a prisoner's pickup or delivery service. The service time s_k to embark or disembark the prisoner from the vehicle must be completed within its time window $t_k = [t_k^-, t_k^+]$. Figure 6 represents the time window by way of a gray rectangle, with the earliest possible execution of the service illustrated by a black rectangle above this time window while the latest possible execution is illustrated with a black box below this same time window. Figure 6(a) illustrates a lengthy time window for servicing prisoner a. A service which must take place at a specific time is represented by way of the time window's length equaling the service time ($s_k = |t_k|$), as indicated in Figure 6(b). The third possible case occurs when a truck departs or arrives at the depot with no prisoners on board, here the service is only restricted by a time window without any service time for (dis)embarking a prisoner ($s_k = 0$), as illustrated in Figure 6(c).

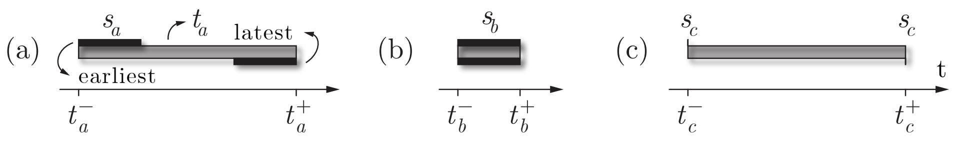

Figure 6: Time window and service time notations.

Inserting a single service into a trip's sequence of η transit points requires evaluating the feasibility of completing all services within their time windows. Although it is possible to evaluate this in $O(\eta)$ time by iterating over the complete trip, an alternative approach based upon auxiliary variables enables $O(1)$ evaluation. First, earliest arrival times a_k^- and earliest departure times d_k^- are precalculated at every location by iterating from the earliest transit over the entire trip (Eq. 47). Second, latest departure times d_k^+ and latest arrival times a_k^+ are precalculated by iterating over the trip from the latest to earliest transit (Eq. 48).

$$a_k^- = d_{k-1}^- + e_{k-1,k} \qquad d_k^- = \max\{a_k^-, t_k^-\} + s_k \tag{47}$$

$$d_k^+ = a_{k+1}^+ - e_{k,k+1} \qquad a_k^+ = \min\{d_k^+, t_k^+\} - s_k \tag{48}$$

Arrival and departure times may therefore only occur within their associated *arrival time window* $a_k = [a_k^-, a_k^+]$ and *departure time window* $d_k = [d_k^-, d_k^+]$. Such *feasibility time windows* are illustrated via two examples in Figure 7. A trip's transit points are denoted by the set $\{T_0, T_1, T_2, T_3\}$, where the first and last points correspond to the depot. The travel time from T_i to T_j (e_{ij}) is represented by an arrow. The arrival time window a and departure time d are illustrated by gray rectangles above and below the service time window t, denoted by $(a, t, d)_i$ for transit point T_i.

Consider the insertion of a service k between services i and j of a trip's service sequence. It is sufficient to simply ensure feasibility of earliest arrival and earliest departure at k: $d_k^- \leq \min\{t_k^+, d_k^+\}$. Equivalently, one could check for feasible latest departure and latest arrival times at k: $a_k^+ \geq \max\{t_k^-, a_k^-\}$. In essence, the length of each arrival window $|a_k|$ is equal to the

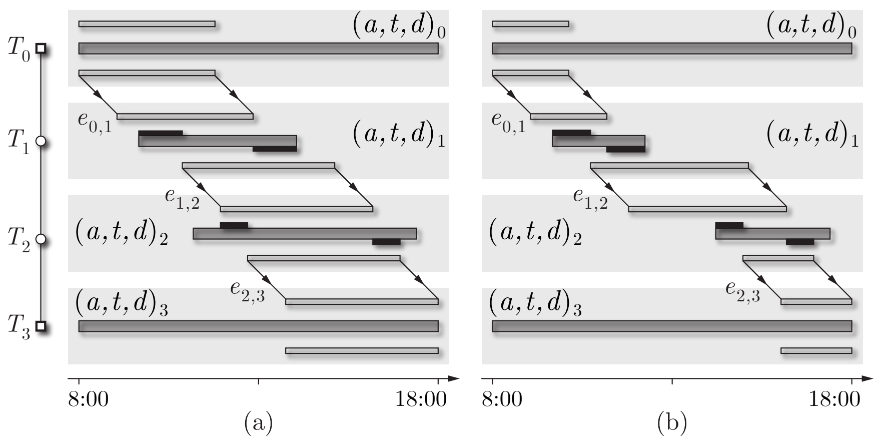

Figure 7: Feasibility time windows, schedules without (a) and with wait times (b).

precomputed maximum delays, as in the *push forward* method proposed by Savelsbergh (1992, 1990) and Kindervater and Savelsbergh (1997).

7.2. Maximum trip duration

In addition to time window constraints, a trip is only feasible if its minimum duration D is no longer than D_{max}. Given feasibility time windows a_k and d_k for all trip transit points, one may determine a fixed time schedule by either earliest or latest trip execution of minimum duration. Figure 8 provides sample fixed schedules for the time windows provided by Figure 7.

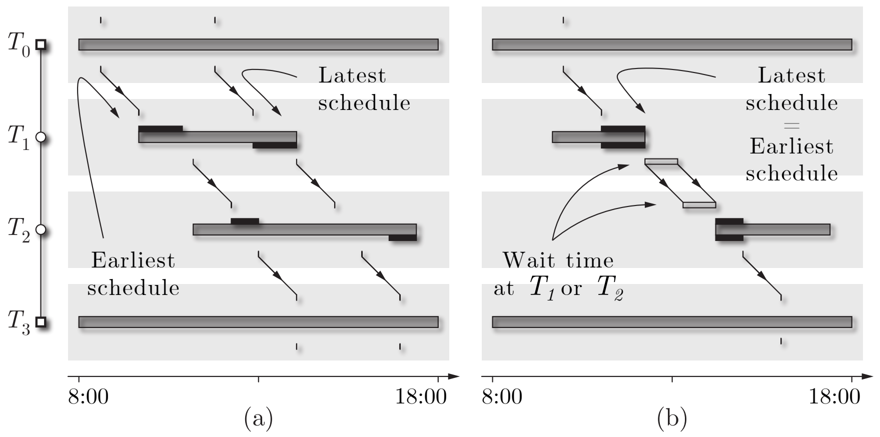

Figure 8: Fixing the schedules, without (a) and with (b) wait times.

The earliest schedule is obtained when all a_k^+ values are updated by pulling back the latest arrival time at the final transit point as much as possible ($a_\eta^+ = a_\eta^-$) and enabling the trip's schedule to begin at a_0^+. Similarly, if the earliest departure time at the first transit point were pushed forward as much as possible ($d_0^- = d_0^+$), before updating all following d_k^+, the latest schedule may be obtained, starting at d_0^-. The duration of a trip may now simply be obtained by $D = a_\eta^- - d_0^+$ of either the earliest or latest schedule. While this approach is inefficient as regards determining D due to its $O(\eta)$ runtime, it is employed given that it helps us introduce the following $O(1)$ approach.

First, notice that if no inevitable wait times are imposed, see Figure 8(a), a trip's duration D is equal to the sum of all travel and service times. This simply corresponds to the amount of time that the trip's vehicle is active, which we refer to as A, given as in Eq. 49.

$$A = \sum_{k=0}^{\eta-1} e_{k,k+1} + \sum_{k=0}^{\eta} s_k \tag{49}$$

When inevitable wait times are imposed, as illustrated in Figure 8(b), the earliest and latest fixed schedules are equal. Let d_0^+ be the latest start time of the trip, a_η^- be the earliest end time of the trip, and $S = a_\eta^- - d_0^+$ denote the time elapsed in between, a trip's duration can then be obtained using S. It is therefore possible to determine D without first fixing the schedule.

When no wait times occur, S may be smaller than D, and even negative. When wait times are imposed, active duration A is shorter than D. Therefore, knowledge of any wait times is not required to calculate a trip's duration via Eq. 50.

$$D = \max\{A, S\} = \max\left\{\sum_{k=0}^{\eta-1} e_{k,k+1} + \sum_{k=0}^{\eta} s_k,\ a_\eta^- - d_0^+\right\} \tag{50}$$

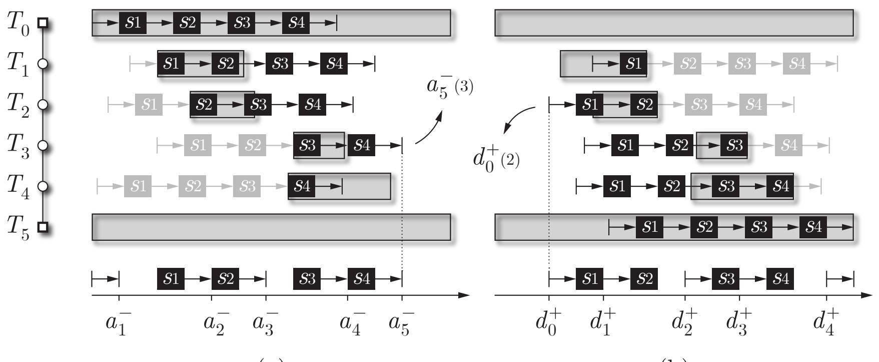

Figure 9: Visual representation of Eq. 52 (a) and Eq. 54 (b).

It is possible to determine the effect of inserting a service i upon a trip's duration as follows. First, as an alternative to the recurrence formulations for earliest arrival time a_k^- (Eq. 47) and latest departure time d_k^+ (Eq. 48), explicit formulations can be derived intuitively. Consider the example in Figure 9, which consists of six transit points, with each point's time window t_i denoted by a gray rectangle and its associated service time by a black square. At every T_i in Figure 9 (a), service i is executed as soon as possible, with its initiation dependent on the start of its time window t_i^-. As such, the earliest arrival time at T_k depends exclusively on the start of the time window at T_i. Let $a_k^-(i)$ be defined as in Eq. 51 for $0 \le i < k$. Then, Eq. 52 gives a_k^-.

$$a_k^-(i) = t_i^- + \sum_{j=i}^{k-1} \Big(s_j + e_{j,j+1}\Big) \tag{51}$$

$$a_k^- = \max_{0 \leq i < k} a_k^-(i) = \max_{0 \leq i < k} \left[t_i^- + \sum_{j=i}^{k-1} \Big(s_j + e_{j,j+1}\Big)\right] \tag{52}$$

For the example in Figure 9 (a), earliest arrival times $a_5^-(0...4)$ depend on $T_{2...4}$ whereas the time window of T_3 delays the arrival time at T_5 the most and therefore $a_5^- = a_5^-(3)$. The earliest arrival times at all other transit points may be obtained via Eq. 52, with the results indicated on the timeline of Figure 9 (a).

A similar approach is employed to derive all latest departure times d_k^+. At every T_i in Figure 9 (b), service i is executed as late as possible, depending exclusively upon the end of its time window t_i^+ and preceded by all prior travel and service times. Define $d_k^+(i)$ as in Eq. 53 for $k < i \leq \eta$, Eq. 54 then gives the value of d_k^+. Among all latest departure times $d_0^+(1...5)$ in Figure 9 (b), $d_0^+(2)$ is the earliest and thus $d_0^+ = d_0^+(2)$.

$$d_k^+(i) = t_i^+ - \sum_{j=k+1}^{i} \Big(s_j + e_{j-1,j}\Big) \tag{53}$$

$$d_k^+ = \min_{k < i \leq \eta} d_k^+(i) = \min_{k < i \leq \eta} \left[t_i^+ - \sum_{j=k+1}^{i} \Big(s_j + e_{j-1,j}\Big)\right] \tag{54}$$

When a service j is inserted between transit points T_i and T_k, the resulting trip duration D^* is affected as follows. First, A^* is obtained by simply adjusting the traversed edges and adding the necessary service time (Eq. 55). The earliest arrival time a_η^- at the last transit point may consequently be delayed due to the additional travel and service times. Let a_k^{-*} be the new earliest arrival time at T_k and A_k^+ be the sum of all service and travel times from T_k to T_η. The earliest arrival time at T_η is only delayed if $a_k^{-*} + A_k^+ > a_\eta^-$ (Eq. 56). Similarly, a_i^{+*} is defined as the new latest departure time at T_i and A_i^- as the sum of all service and travel times from T_0 to T_i. The new latest departure time d_0^+ can be obtained via Eq. 57.

$$A^* = A - e_{i,k} + e_{i,j} + e_{j,k} + s_j \tag{55}$$

$$a_\eta^{-*} = \max\{a_\eta^-, a_k^{-*} + A_k^+\} \tag{56}$$

$$d_0^{+*} = \min\{d_\eta^+, d_i^{+*} - A_i^-\} \tag{57}$$

Assuming feasibility time windows a_i and d_i are precalculated together with the sum of preceding A_i^- and succeeding A_i^+ activity at each transit point T_i, it is possible to validate time window feasibility and maximum trip duration in $O(1)$.

7.3. Simultaneous servicing (merged service times)

Multiple services conducted by a vehicle at a single transit point may be executed simultaneously provided that each individual service is executed within its time window. Therefore, the total time spent at the transit point may be longer than the longest service time of those services. Execution of services $a(t_a, s_a)$ and $b(t_b, s_b)$ at a single transit point may therefore be merged into the service $m(t_m, s_m)$ which represents the shortest feasible execution of both. Earliest departure d_m^- and latest arrival a_m^+ times are derived simply by Eqs. 58 and 59.

$$d_m^- = \min\{d_a^-, d_b^-\} \tag{58}$$

$$a_m^+ = \max\{a_a^+, a_b^+\} \tag{59}$$

Merged service time s_m and its associated time window t_m are derived from d_m^- and a_m^+ by Eqs. 60, 61 and 62.

$$s_m = \max\{s_a, s_b, d_m^- - a_m^+\} \tag{60}$$

$$t_m^- = d_m^- - s_m \tag{61}$$

$$t_m^+ = a_m^+ + s_m \tag{62}$$

Figure 10 illustrates four examples of two services $(t, s)_a$ and $(t, s)_b$ merged into single service $(t, s)_m$. This difference between the earliest departure d_m^- and latest arrival time a_m^+ is progressively increased in each example from (a) to (d). The difference in example (a) is negative, thus much shorter than the longest service time, which ultimately results in a service with a lengthy time window. Meanwhile, for (d) the difference is greater than the longest service time and thus equals merged service time s_m with time window $t_m = s_m$, thereby representing a fixed service in time.

7.4. Maximum ride times

In order to guarantee maximum ride time feasibility, we apply the following preprocessing step. Let t_r^{p-} and s_r^p be the beginning of pickup time window and pickup service time for request r and let t_r^{d+} and s_r^d be the end of delivery time window and delivery service time. Feasibility for the request's maximum ride time L_r is ensured by restricting the earliest pickup time $\hat{t}_r^{p-}$, as in Eq. 63.

$$\hat{t}_r^{p-} = \max\left\{t_r^{p-}, t_r^{d+} - \left(s_r^p + L_r + s_r^d\right)\right\} \tag{63}$$

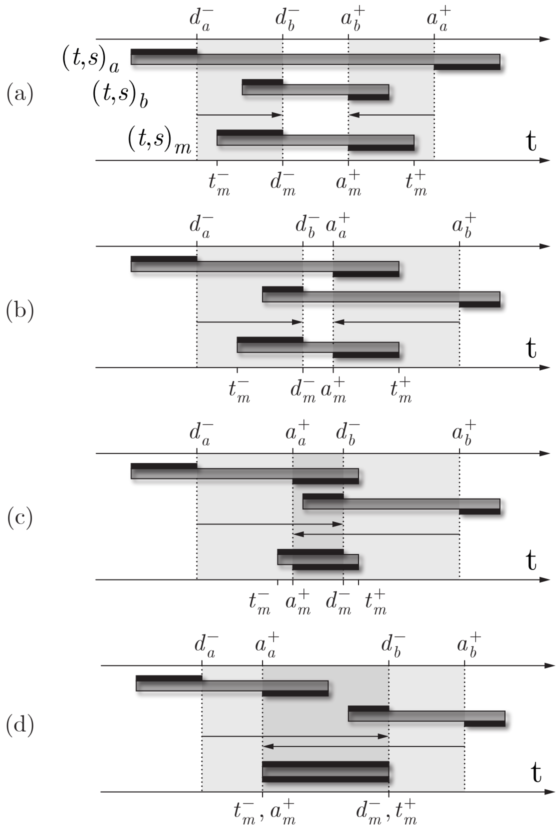

Figure 10: Four examples of merging services a and b into m.

8. PTP instances and solutions

The proposed instances have been made publicly available at the PTP benchmark site[1] in order to stimulate further research regarding the problem. Real world PTP data and the accompanying schedules for a single day of prisoner transports were obtained, which enabled a fair comparison between the heuristically generated and the manually constructed solutions. The location coordinates of PTP instances that were generated using real world data have been altered in the interest of preserving confidentiality. As such, it is impossible to retrieve the original coordinates from any of the instances that have been made publicly available. This ensures that private user information cannot be unambiguously extrapolated from these instances.

Table 3 summarizes instance properties as a function of the number of prisoner transportation requests. The smallest number of requests r begins at 50, immediately challenging exact approaches. Instances of sizes 100, 200, 400 and, in order to reflect the real world problem size, 700 were also generated. The number of prisoners per instance p varies from 38 to 476, with the number of unique locations associated with their trips ranging from 28 to 169, of which between 1 and 13 are depot locations. Cartesian coordinates for each location are obtained by way of clas-

[1]https://benchmark.gent.cs.kuleuven.be/ptp/

sical multidimensional scaling of real travel times. Therefore, the cartesian distance between the obtained locations' coordinates provide realistic travel times. A single depot is assumed for the smallest instances, all the way up to a total of 13 for the largest instances. This number of depots is based on the ratio of trips per depot present in the real world data. The maximum trip duration is fixed to 9 hours (540 minutes) for all instances. For each transportation request, separate time windows and service times for pickup and delivery services are defined. Time windows occur in a timespan of 24 hours, and although service times are typically 15 minutes, they may be as long as 225 minutes.

Table 3: Common properties of the PTP instances.

Name	r	p	l	u	D_{max}
(x)-r050-(v)	50	38	28	1	540
(x)-r100-(v)	100	71	45	2	540
(x)-r200-(v)	200	140	70	4	540
(x)-r400-(v)	400	264	101	8	540
(x)-r700-(v)	700	476	169	13	540

Ten different vehicle types are considered for each of these instances (Table 4), resulting in a total of 50 PTP instances. Each vehicle type's capacity Q totals four seats for $v1$ - $v5$ and seven seats for $v6$ - $v10$, which are grouped in various compartments. The vehicle types contain compartments of various sizes, a few of which are worth detailing. Vehicle types $v1$ and $v6$ have compartments of one single seat. Such vehicles always avoid inter-passenger conflicts at compartment level, however they are less common in practice due to their high investment costs. Vehicle types $v2$ and $v7$ represent vehicle configurations with two (almost) equal sized compartments. Vehicles of $v5$ and $v10$ type contain one single compartment and are, therefore, incapable of combining conflicting prisoners.

Table 4: PTP vehicle types.

Name	Q	C	Name	Q	C
v1	4	$\{1,1,1,1\}$	v6	7	$\{1,1,1,1,1,1,1\}$
v2	4	$\{2,2\}$	v7	7	$\{3,4\}$
v3	4	$\{1,1,2\}$	v8	7	$\{2,2,3\}$
v4	4	$\{1,3\}$	v9	7	$\{1,6\}$
v5	4	$\{4\}$	v10	7	$\{7\}$

In order to assess the impact of inter-passenger conflicts and maximum ride time restrictions, this set of instances - referred to as Set A - is duplicated to solve the PTP by relaxing these constraints one at a time. This results in three additional instance sets: Set B with $H \neq \emptyset$ and

$L_r = D_{max}$ $\forall r \in R$, Set C with $H = \emptyset$ and $L_r = 150$ $\forall r \in R$, and Set D with $H = \emptyset$ and $L_r = D_{max}$ $\forall r \in R$.

Relaxation of these conflicts essentially constitutes instances for the multi depot DARP with simultaneous servicing. Additionally, when relaxing both conflicts and maximum ride durations, the problem becomes a multi-depot pickup-and-delivery problem with time windows and simultaneous servicing. Notice that only one vehicle type is employed for the problem sets where conflicts are relaxed, given that the vehicle's total capacity then becomes the only relevant constraint.

In order to provide initial solutions and verify *error-free* instance files for these four benchmark sets, ten runs are conducted for which the best (*Best*) and average (*Avg.*) results are accompanied by average run times T_a in minutes. These results are detailed in Tables 5 and 6. Considering only the real world scale of the problem instances, which contain 700 requests, we observe the following. The presence of inter-passenger conflicts implies an overall cost increase of up to 4.4% when employing small vehicles ($v1$ - $v5$) and up to 5.3% when employing large vehicles ($v6$ - $v10$). Meanwhile, the effect of inter-passenger conflicts at compartment level can be determined by comparing results for $v1$, which is always capable of combining conflicting prisoners, and $v5$, which is always incapable of combing them. Employing $v1$ rather than $v5$ implies a cost reduction of 1.8%. Similarly, employing $v6$ rather than $v10$ implies a cost reduction of 1.6%. Increasing the vehicle's capacity from 4 ($v1$ - $v5$) to 7 seats ($v6$ - $v10$) may reduce the total cost by up to 2.1%. Finally, we observe that the presence of the maximum ride time constraint may triple the cost. These results are not without practical significance given that the maximum ride time constraints were not initially considered as critical constraints by the prisoner transportation company involved in this research. However, after modeling the problem and evaluating the solution approach through these experiments, it became apparent that maximum ride times are indeed crucial for prisoner transportation. On the other hand, it is very likely that this observation is not only related to the constraint, but also to how it is accommodated by preprocessing the time windows. Therefore, the computational results of maximum ride time must not be regarded as an independent observation.

9. Conclusion

This paper introduced the Prisoner Transportation Problem (PTP), which constitutes a generalization of common combinatorial optimization problems: vehicle routing with time windows, dial-a-ride and multi-compartment vehicle routing. The PTP is interesting from a practical viewpoint because its daily operations inevitably represent a huge cost for society. Its modeling difficulty is primarily related to the presence of inter-passenger conflicts and simultaneous servicing, two intricate constraints for which models were previously absent. The PTP therefore posed a significant academic challenge.

Together with several mathematical formulations this paper contributed a set of instances and provides initial benchmarks obtained by applying a local search based approach in order

Table 5: PTP instance and results for vehicle types v1 to v5.

Set A(v1-v5) ($H \neq \emptyset, L_r = 150$)				Set B(v1-v5) ($H \neq \emptyset, L_r = D_{max}$)			
Instance	Best	Avg.	T_a	Instance	Best	Avg.	T_a
a-r050-v1	(18) 7385	7385.7	1.1	b-r050-v1	(6) 2208	2216.0	3.0
a-r050-v2	(18) 7382	7384.4	1.0	b-r050-v2	(6) 2211	2226.4	2.6
a-r050-v3	(18) 7382	7387.2	1.1	b-r050-v3	(6) 2211	2217.6	2.9
a-r050-v4	(18) 7413	7413.2	1.0	b-r050-v4	(6) 2220	2229.9	2.8
a-r050-v5	(18) 7434	7441.1	1.0	b-r050-v5	(6) 2392	2403.3	2.3
a-r100-v1	(37) 16514	16524.5	1.9	b-r100-v1	(13) 5317	5348.9	4.4
a-r100-v2	(38) 16511	16524.9	1.7	b-r100-v2	(13) 5275	5322.1	3.9
a-r100-v3	(37) 16509	16530.2	1.8	b-r100-v3	(13) 5294	5338.6	4.2
a-r100-v4	(38) 16546	16559.9	1.8	b-r100-v4	(13) 5315	5364.0	4.0
a-r100-v5	(38) 16604	16615.2	1.7	b-r100-v5	(14) 5458	5489.6	3.3
a-r200-v1	(67) 30308	30376.3	3.6	b-r200-v1	(21) 9778	9922.6	9.5
a-r200-v2	(67) 30337	30370.4	3.3	b-r200-v2	(22) 9878	9990.4	8.0
a-r200-v3	(67) 30311	30359.8	3.5	b-r200-v3	(22) 9814	9967.3	9.0
a-r200-v4	(67) 30371	30427.1	3.4	b-r200-v4	(22) 9872	9990.6	8.3
a-r200-v5	(68) 30671	30751.5	3.2	b-r200-v5	(22) 9896	10075.8	6.9
a-r400-v1	(128) 59030	59164.4	7.3	b-r400-v1	(41) 19562	19809.8	18.3
a-r400-v2	(128) 59134	59212.2	6.8	b-r400-v2	(42) 19715	19924.5	15.7
a-r400-v3	(127) 59030	59152.6	7.2	b-r400-v3	(42) 19765	19971.0	17.3
a-r400-v4	(127) 59123	59207.2	7.0	b-r400-v4	(41) 19576	19896.5	16.3
a-r400-v5	(131) 60128	60307.6	6.5	b-r400-v5	(42) 20014	20305.2	13.4
a-r700-v1	(274) 120836	121019.7	12.0	b-r700-v1	(84) 41270	41570.8	27.7
a-r700-v2	(272) 121292	121503.2	11.5	b-r700-v2	(84) 41511	41930.5	24.5
a-r700-v3	(272) 120820	120990.8	12.0	b-r700-v3	(85) 41116	41466.1	26.7
a-r700-v4	(275) 120865	121143.1	11.6	b-r700-v4	(83) 41218	41593.0	25.5
a-r700-v5	(276) 123044	123185.8	10.9	b-r700-v5	(87) 42527	42804.5	20.8

Set C(v1) ($H = \emptyset, L_r = 150$)				Set D(v1) ($H = \emptyset, L_r = D_{max}$)			
Instance	Best	Avg.	T_a	Instance	Best	Avg.	T_a
c-r050-v1	(18) 7343	7352.5	1.1	d-r050-v1	(6) 2118	2133.3	3.0
c-r100-v1	(37) 16346	16374.9	2.0	d-r100-v1	(12) 4991	5070.6	6.8
c-r200-v1	(64) 29510	29572.6	3.7	d-r200-v1	(21) 9316	9453.4	14.5
c-r400-v1	(123) 57411	57583.1	7.4	d-r400-v1	(40) 18821	19094.2	24.9
c-r700-v1	(261) 117399	117947.9	12.1	d-r700-v1	(78) 38125	38604.9	36.0

Table 6: PTP instance and results for vehicle types v6 to v10.

Set A(v6-v10) ($H \neq \emptyset, L_r = 150$)				Set B(v6-v10) ($H \neq \emptyset, L_r = D_{max}$)			
Instance	Best	Avg.	T_a	Instance	Best	Avg.	T_a
a-r050-v6	(18) 7343	7343.0	1.1	b-r050-v6	(6) 2102	2126.1	3.0
a-r050-v7	(18) 7343	7343.0	1.0	b-r050-v7	(6) 2113	2127.7	3.0
a-r050-v8	(18) 7343	7343.0	1.1	b-r050-v8	(6) 2102	2124.9	3.0
a-r050-v9	(18) 7371	7373.4	1.0	b-r050-v9	(6) 2131	2141.4	3.0
a-r050-v10	(18) 7395	7399.5	1.0	b-r050-v10	(7) 2318	2327.7	2.5
a-r100-v6	(36) 16112	16135.3	2.0	b-r100-v6	(13) 4970	5014.5	5.8
a-r100-v7	(37) 16175	16195.3	1.7	b-r100-v7	(14) 4982	5022.9	4.4
a-r100-v8	(37) 16185	16201.4	1.8	b-r100-v8	(13) 4987	5015.0	4.8
a-r100-v9	(36) 16113	16124.6	1.8	b-r100-v9	(14) 4945	5013.9	4.6
a-r100-v10	(37) 16206	16224.5	1.7	b-r100-v10	(13) 5107	5137.6	3.8
a-r200-v6	(65) 29377	29431.2	3.8	b-r200-v6	(20) 8970	9067.7	12.9
a-r200-v7	(65) 29375	29431.7	3.3	b-r200-v7	(20) 9010	9096.7	9.2
a-r200-v8	(65) 29404	29448.8	3.5	b-r200-v8	(20) 8939	9067.4	10.6
a-r200-v9	(66) 29421	29452.8	3.4	b-r200-v9	(21) 9043	9131.8	9.6
a-r200-v10	(66) 29694	29727.6	3.2	b-r200-v10	(20) 9160	9298.7	7.9
a-r400-v6	(124) 57248	57330.0	7.6	b-r400-v6	(37) 17625	17802.8	25.3
a-r400-v7	(125) 57236	57380.2	6.8	b-r400-v7	(39) 17812	17927.8	18.1
a-r400-v8	(123) 57272	57404.6	7.0	b-r400-v8	(38) 17692	17872.6	20.0
a-r400-v9	(124) 57274	57406.5	7.0	b-r400-v9	(40) 17837	17961.8	18.8
a-r400-v10	(128) 58636	58734.6	6.5	b-r400-v10	(40) 18204	18434.4	14.9
a-r700-v6	(264) 118372	118748.7	12.6	b-r700-v6	(78) 37566	37937.0	35.3
a-r700-v7	(267) 118722	119041.6	11.3	b-r700-v7	(77) 37756	38060.7	26.2
a-r700-v8	(263) 118282	118720.6	11.8	b-r700-v8	(79) 37734	38101.8	29.1
a-r700-v9	(263) 118301	118642.3	11.6	b-r700-v9	(77) 37947	38218.2	27.9
a-r700-v10	(273) 120523	120690.7	10.8	b-r700-v10	(81) 39263	39652.5	22.0

Set C(v6) ($H = \emptyset, L_r = 150$)				Set D(v6) ($H = \emptyset, L_r = D_{max}$)			
Instance	Best	Avg.	T_a	Instance	Best	Avg.	T_a
c-r050-v6	(17) 7303	7303.0	1.2	d-r050-v6	(5) 1964	1973.5	3.0
c-r100-v6	(36) 15948	15969.6	2.1	d-r100-v6	(12) 4620	4660.9	10.2
c-r200-v6	(61) 28427	28502.2	3.9	d-r200-v6	(19) 8305	8460.8	24.2
c-r400-v6	(116) 55359	55606.1	7.7	d-r400-v6	(35) 16379	16552.8	41.0
c-r700-v6	(250) 114166	114657.3	12.5	d-r700-v6	(70) 33554	33767.2	51.2

to stimulate further research on the PTP. While constructing and optimizing the PTP's trips, feasible prisoner-to-compartment allocations are essential. An efficient and effective heuristic has been introduced for solving this subproblem.

The mathematical formulations introduced in this paper set in place the strong foundations necessary for subsequent research on the PTP. Possible future research directions include establishing further theoretical insights as well as practical algorithms for improving solutions. Several variants being already discussed in this paper, it is also worthwhile to study the impact of additional real world constraints which have, for the sake of clarity, been omitted in this study.

Acknowledgement

Work funded by IWT 130855 grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) in cooperation with Conundra (www.conundra.eu), and supported by the Belgian Science Policy Office (BELSPO) in the Interuniversity Attraction Pole COMEX (http://comex.ulb.ac.be). This research is also partially supported by Data-driven logistics (FWO-S007318N). Editorial consultation provided by Luke Connolly (KU Leuven).

References

Al-Omari H, Sabri KE. New graph coloring algorithms. American Journal of Mathematics and Statistics 2006;2 (4):739–41.

Beaudry A, Laporte G, Melo T, Nickel S. Dynamic transportation of patients in hospitals. OR spectrum 2010;32(1):77–107.

Cordeau JF, Laporte G. The dial-a-ride problem: models and algorithms. Annals of Operations Research 2007;153(1):29–46.

Derigs U, Gottlieb J, Kalkoff J, Piesche M, Rothlauf F, Vogel U. Vehicle routing with compartments: applications, modelling and heuristics. OR spectrum 2011;33(4):885–914.

Desaulniers G. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. Operations Research 2010;58(1):179–92.

Eksioglu B, Vural AV, Reisman A. The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering 2009;57(4):1472–83.

Fallahi AE, Prins C, Calvo RW. A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. Computers & Operations Research 2008;35(5):1725–41.

Hansen P, Hertz A, Kuplinsky J. Bounded vertex colorings of graphs. Discrete Mathematics 1993;111(1-3):305–12.

Henke T, Speranza MG, Wäscher G. The multi-compartment vehicle routing problem with flexible compartment sizes. European Journal of Operational Research 2015;246(3):730–43.

Kindervater GA, Savelsbergh M. Vehicle routing: handling edge exchanges. In: Aarts E, Lenstra J, editors. Local search in combinatorial optimization. Wiley Chichester, UK; 1997. p. 337–60.

Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. SCIENCE 1983;220(4598):671–80.

Lahyani R, Coelho LC, Khemakhem M, Laporte G, Semet F. A multi-compartment vehicle routing problem arising in the collection of olive oil in tunisia. Omega 2015;51:1–10.

Laporte G. Fifty years of vehicle routing. Transportation Science 2009;43(4):408–16.

Molenbruch Y, Braekers K, Caris A, Vanden Berghe G. Multi-directional local search for a bi-objective dial-a-ride problem in patient transportation. Computers & Operations Research 2017;77:58–71.

Öncan T, Altınel İK, Laporte G. A comparative analysis of several asymmetric traveling salesman problem formulations. Computers & Operations Research 2009;36(3):637–54.

Oppen J, Løkketangen A. A tabu search approach for the livestock collection problem. Computers & Operations Research 2008;35(10):3213–29.

Oppen J, Løkketangen A, Desrosiers J. Solving a rich vehicle routing and inventory problem using column generation. Computers & Operations Research 2010;37(7):1308–17.

Paraskevopoulos DC, Laporte G, Repoussis PP, Tarantilis CD. Resource constrained routing and scheduling: Review and research prospects. European Journal of Operational Research 2017;263(3):737–54.

Parragh SN, Cordeau JF, Doerner KF, Hartl RF. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. OR spectrum 2012;34(3):593–633.

Parragh SN, Pinho de Sousa J, Almada-Lobo B. The dial-a-ride problem with split requests and profits. Transportation Science 2015;49(2):311–34.

Partyka J, Hall R. On the road to service. ORMS Today 2000;27(4):26–30.

Potts CN, Kovalyov MY. Scheduling with batching: A review. European Journal of Operational Research 2000;120(2):228–49.

Rais A, Alvelos F, Carvalho MS. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. European Journal of Operational Research 2014;235(3):530–9.

Salani M, Vacca I. Branch and price for the vehicle routing problem with discrete split deliveries and time windows. European Journal of Operational Research 2011;213(3):470–7.

Savelsbergh M. An efficient implementation of local search algorithms for constrained routing problems. European Journal of Operational Research 1990;47(1):75–85.

Savelsbergh M. The vehicle routing problem with time windows: Minimizing route duration. ORSA Journal on Computing 1992;4(2):146.

Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G. Record breaking optimization results using the ruin and recreate principle. Journal of Computational Physics 2000;159(2):139–71.

Toth P, Vigo D. Heuristic algorithms for the handicapped persons transportation problem. Transportation Science 1997;31(1):60–71.

Appendices

Appendix A. Formulation with single-index time variables (FPTP-2)

In this formulation, $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ variables are defined as before. Variable τ_i denotes the departure time from node $i \in I$ and δ_i is the merged service time of node $i \in I$. Moreover, ϕ^v is the departure time from the starting depot and Φ^v is the arrival time to the ending depot of vehicle $v \in V$. The following is a mixed integer programming formulation for the PTP with single-index time variables.

$$\min \sum_{v \in V} (\Phi^v - \phi^v) \tag{A.1}$$

$$\text{s.t. } (2) - (12), (22) - (27)$$

$$\tau_i + e_{ij} + \delta_j \le \tau_j + t^+_{max}(1 - \sum_{v \in V} z^v_{ij}) \quad \forall (i,j) \in A^3 : l(i) \neq l(j) \tag{A.2}$$

$$\tau_i \le \tau_j + t^+_i(1 - \sum_{v \in V} z^v_{ij}) \quad \forall (i,j) \in A^3 : l(i) = l(j) \tag{A.3}$$

$$\delta_i \geq \sum_{v \in V} \sum_{j \in I} s_j z_{ij}^v \qquad \forall i \in I \qquad \text{(A.4)}$$

$$\delta_j \geq \sum_{v \in V} \sum_{i \in I} s_i z_{ij}^v \qquad \forall j \in I \qquad \text{(A.5)}$$

$$\tau_{n+i} - \tau_i - s_{i+n} \leq L_{r(i)} \qquad \forall i \in I^P \qquad \text{(A.6)}$$

$$\Phi^v - \phi^v \leq D_{max} \qquad \forall v \in V \qquad \text{(A.7)}$$

$$\Phi^v - \phi^v \geq \sum_{i \in I^P} (e_{u_v i} + s_i) z_{u_v i}^v + \sum_{i \in I^D} (e_{i u_v^e} + s_i) z_{i u_v^e}^v + \sum_{(i,j) \in A^3} e_{ij} z_{ij}^v \qquad \forall v \in V \qquad \text{(A.8)}$$

$$\Phi^v \geq \tau_i + e_{i u_v^e} + t_{max}(z_{i u_v^e}^v - 1) \qquad \forall v \in V, i \in I \qquad \text{(A.9)}$$

$$\phi^v \leq \tau_i - e_{u_v i} - \delta_i + t_{max}^+(1 - z_{u_v i}^v) \qquad \forall v \in V, i \in I \qquad \text{(A.10)}$$

$$t_i^- + s_i \leq \tau_i \leq t_i^+ \qquad \forall i \in I \qquad \text{(A.11)}$$

$$s_i \leq \delta_i \leq L_{r(i)} \qquad \forall i \in I \qquad \text{(A.12)}$$

The objective function (A.1) of FPTP-2 is expressed as a linear function of single-index Φ^v and ϕ^v variables for $v \in V$. Constraints (A.2), (A.10) and (A.9) replace (13) of the original model. The remaining constraints are analogous to those omitted from FPTP.

Appendix B. Formulation without $\boldsymbol{\delta}$ variables (FPTP-3)

In addition to FPTP and FPTP-2, we provide a third formulation which does not contain $\boldsymbol{\delta}$ variables. Model FPTP-3 then becomes:

$$\begin{aligned} \min\ & (1) \\ \text{s.t. } & (2) - (12), (22) - (27), (14), (17) - (20) \\ & \tau_i^v + e_{ij} + s_j \leq \tau_j^v + t_{max}(1 - z_{ij}^v) \qquad \forall v \in V, (i,j) \in A : l(i) \neq l(j) \qquad \text{(B.1)} \end{aligned}$$

As in the case of FPTP, we utilize all the valid inequalities introduced in Section 3 for PTP-2 and PTP-3 whenever applicable. Additionally for PTP-2, we replace (A.9) and (A.10) with stronger constraints (B.2) and (B.3).

$$\Phi^v \geq \tau_i + e_{i u_v^e} + t_{max}(\sum_{j \in I} z_{ij}^v - 1) \qquad \forall v \in V, i \in I \qquad \text{(B.2)}$$

$$\phi^v \leq \tau_i - e_{u_v i} - \delta_i + t_{max}^+(1 - \sum_{j \in I} z_{ji}^v) \qquad \forall v \in V, i \in I \qquad \text{(B.3)}$$

Appendix C. Comparison of the formulations and iterative algorithms for PTP

Table C.7: Comparison of PTP methods.

					FPTP			FPTP-2			FPTP-3			IT-FPTP		IT-PTP-2		IT-PTP-3	
id	Comp.	n	Best	k	Obj	$t(s)$	$g\%$	Obj	$t(s)$	$g\%$	Obj	$t(s)$	$g\%$	Obj	$t(s)$	Obj	$t(s)$	Obj	$t(s)$
4_0_4	1-1-1-1	4	**54**	1	**54**	9.73	0.00	**54**	48.04	0.00	**54**	5.2	0.00	**54**	3.21	**54**	9.44	**54**	4.30
4_1_4	1-1-1-1		**271**	2	**271**	0.22	0.00	**271**	9.62	0.00	**271**	0.38	0.00	**271**	0.65	**271**	7.55	**271**	0.92
4_1_2	2-2		**271**	2	**271**	0.24	0.00	**271**	3.50	0.00	**271**	0.18	0.00	**271**	17.12	**271**	6.22	**271**	0.64
4_1_1	4		**271**	2	**271**	0.19	0.00	**271**	3.72	0.00	**271**	0.22	0.00	**271**	15.43	**271**	5.78	**271**	0.36
4_2_4	1-1-1-1		**181**	2	**181**	0.62	0.00	**181**	10.47	0.00	**181**	1.43	0.00	**181**	2.04	**181**	8.08	**181**	2.57
4_2_2	2-2		**181**	2	**181**	0.98	0.00	**181**	5.53	0.00	**181**	1.16	0.00	**181**	2.10	**181**	11.91	**181**	1.52
4_2_1	4		**181**	2	**181**	1.25	0.00	**181**	4.76	0.00	**181**	0.37	0.00	**181**	12.54	**181**	6.86	**181**	1.49
6_0_4	1-1-1-1	6	**90**	2	90	TL	93.89	90	TL	93.33	**90**	TL	92.33	**90**	405.92	**90**	399.15	**90**	506.80
6_1_4	1-1-1-1		**399**	1	**399**	93.68	0.00	**399**	72.13	0.00	**399**	35.17	0.00	**399**	10.29	**399**	8.74	**399**	6.69
6_1_2	2-2		**399**	1	**399**	5.96	0.00	**399**	12.93	0.00	**399**	7.47	0.00	**399**	11.78	**399**	7.77	**399**	3.85
6_1_1	4		**403**	1	**403**	2.35	0.00	**403**	9.03	0.00	**403**	1.83	0.00	**403**	7.88	**403**	21.27	**403**	0.98
6_2_4	1-1-1-1		**282**	3	**282**	353.12	0.00	**282**	331.20	0.00	**282**	89.94	0.00	**282**	34.49	**282**	38.02	**282**	75.83
6_2_2	2-2		**282**	3	**282**	130.94	0.00	**282**	192.35	0.00	**282**	110.53	0.00	**282**	13.61	**282**	15.19	**282**	16.63
6_2_1	4		**282**	3	**282**	6.27	0.00	**282**	36.82	0.00	**282**	11.82	0.00	**282**	114.64	**282**	74.58	**282**	6.44
8_0_4	1-1-1-1	8	106	2	126	TL	97.88	294	TL	99.46	128	TL	100.00	106	TL	106	TL	106	TL
8_1_4	1-1-1-1		**516**	2	519	TL	60.67	516	TL	66.38	516	TL	56.77	516	TL	**516**	4281.62	**516**	3165.77
8_1_2	2-2		**516**	2	516	TL	37.44	516	TL	38.87	516	TL	39.15	**516**	1659.33	**516**	849.50	**516**	1114.13
8_1_1	4		**521**	2	**521**	169.04	0.00	**521**	215.38	0.00	**521**	223.97	0.00	**521**	23.22	**521**	56.44	**521**	24.25
8_2_4	1-1-1-1		**356**	3	393	TL	59.45	356	TL	49.89	356	TL	54.19	356	TL	**356**	5061.98	356	TL
8_2_2	2-2		**356**	3	356	TL	16.57	356	TL	18.03	356	TL	18.57	**356**	2175.07	**356**	911.33	**356**	2160.09
8_2_1	4		**356**	3	**356**	2757.98	0.00	356	TL	16.29	356	2543.25	0.00	**356**	823.36	**356**	1037.22	**356**	486.27
10_0_4	1-1-1-1	10	106	2	273	TL	100.00	423	TL	100.00	366	TL	100.00	106	TL	106	TL	131	TL
10_1_4	1-1-1-1		599	2	607	TL	75.86	601	TL	75.39	607	TL	73.63	599	TL	599	TL	599	TL
10_1_2	2-2		599	2	682	TL	84.90	620	TL	70.91	656	TL	84.36	599	TL	599	TL	599	TL
10_1_1	4		**601**	2	603	TL	31.82	603	TL	34.30	603	TL	32.43	**601**	488.14	**601**	410.27	**601**	334.30
10_2_4	1-1-1-1		451	2	706	TL	76.00	841	TL	79.84	770	TL	78.01	451	TL	453	TL	453	TL
10_2_2	2-2		450	2	531	TL	67.93	579	TL	69.01	519	TL	66.90	450	TL	450	TL	450	TL
10_2_1	4		450	2	467	TL	63.09	453	TL	57.24	451	TL	60.53	450	TL	450	TL	450	TL