



Optimization Models for Pickup-And-Delivery Problems with Reconfigurable Capacities

Arnoosh Golestanian ✉ 

Department of Mechanical and Industrial Engineering, University of Toronto, Canada

Giovanni Lo Bianco ✉ 

Department of Mechanical and Industrial Engineering, University of Toronto, Canada

Chengyu Tao ✉ 

Department of Mechanical and Industrial Engineering, University of Toronto, Canada

J. Christopher Beck ✉ 

Department of Mechanical and Industrial Engineering, University of Toronto, Canada

Abstract

When a transportation service accommodates both people and goods, operators sometimes opt for vehicles that can be dynamically reconfigured for different demands. Motivated by air service in remote communities in Canada's north, we define a pickup-and-delivery problem in which aircraft can add or remove seats during a multi-stop trip to accommodate varying demands. Given the demand for people and cargo as well as a seat inventory at each location, the problem consists in finding a tour that picks up and delivers all demand while potentially reconfiguring the vehicle capacity at each location by adding or removing seats. We develop a total of six models using three different approaches: constraint programming, mixed integer programming, and domain-independent dynamic programming. Our numerical experiments indicate that domain-independent dynamic programming is able to substantially outperform the other technologies on both solution quality and run-time on a set of randomly generated instances spanning the size of real problems in northern Canada.

2012 ACM Subject Classification Computing methodologies → Modeling methodologies

Keywords and phrases Pickup and delivery, Dial-a-ride problem, Optimization

Digital Object Identifier 10.4230/LIPIcs.CP.2023.17

Funding This research was supported by the Natural Sciences and Engineering Research Council of Canada.

1 Introduction

Pickup-and-delivery problems involve using vehicles to transport goods and/or passengers from a set of origins to a set of destinations on a given transportation network [1]. A typical pickup-and-delivery problem such as the Pickup and Delivery Traveling Salesperson Problem (PD-TSP) includes a one or more vehicles, requests with different pickup and delivery locations, and an objective to find a minimum-cost tour (or set of routes) that visit(s) each pickup location before its corresponding delivery location [4]. There has been substantial research literature on pickup and delivery problems over the past several years (e.g., [19, 21]) motivated, in part, by global efforts to reduce transportation-related carbon emissions [16]. Many variations of such problems have been proposed and studied in the operations research literature. For example, some problems include handling costs when an item is loaded or unloaded depending on the position of the item in the vehicle [24] and some include subsets of requests that cannot be in a vehicle at the same time [5].

In this paper, we propose and study a novel variation of PD-TSP: requests can include both goods (cargo) and passengers and the vehicle has a capacity that can be adjusted en-route depending on the request and equipment stored at locations in the network. The



© Arnoosh Golestanian, Giovanni Lo Bianco, Chengyu Tao, and J. Christopher Beck; licensed under Creative Commons License CC-BY 4.0

29th International Conference on Principles and Practice of Constraint Programming (CP 2023).

Editor: Roland H. C. Yap; Article No. 17; pp. 17:1–17:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

problem is motivated by a real transportation problem faced by air services in northern Canada. Since many communities in this region are reachable only by air during some parts of the year, their access to basic human needs such as fresh food and healthcare services is limited. The need for air transportation combined with the relatively small populations and lack of resources led northern air services to adopt the practice of transporting both cargo and passengers on the same flights. The vehicles are aircraft with removable seats, allowing staff to either remove passenger seats and store them at airports to transport more cargo or add additional seats, previously stored at airports, to carry more passengers. The problem, which we call the Pickup-and-Delivery with Seat Replacement Problem (PD-SRP), therefore requires finding the shortest tour delivering all goods and passengers from their origins to destinations without exceeding aircraft capacity but allowing seats to be removed from or added to aircraft at each location, subject to seat availability and total aircraft capacity.

To solve the PD-SRP, we developed three types of optimization models: one Constraint Programming (CP) model, three Mixed Integer Programming (MIP) models, and two Domain-Independent Dynamic Programming (DIDP) [14] models. We compare their performance on randomly generated instances based on the size of the problem in Canada's north, demonstrating that both of the DIDP models outperform the CP and MIP models in terms of the number of instances solved and proved optimal, solution quality, and solve time.

2 Related Works

Reconfigurable capacity is a general term in the transportation literature, typically indicating that vehicle capacity can be changed at some cost and/or limited by some constraints [22, 23]. Other terms such as multi-compartment vehicle or multi-purpose vehicle are used to convey a similar meaning [20, 8]. We review the vehicle routing and dial-a-ride problems literature for studies that considered adjustable vehicles.

Vehicle Routing Problems (VRP): The Vehicle Routing Problem and its many variations have been studied extensively over the past 50 years [18]. The idea of adjusting the vehicle to handle different types of demand has been studied in multi-compartment vehicle routing problems [20]. For example, Henke et al. [9] studied how to split the capacity of a truck into different compartments to maintain the separation of different colors of recycled glass. Similarly, for grocery distribution, different temperature-sensitive products can be transported on the same truck with multiple compartments [11]. In both of these problems, a vehicle's capacity configuration is fixed for its entire route and cannot be modified during the trip.

Dial-a-Ride Problems (DARP): In the Dial-a-Ride Problem a transportation request takes the form of pickup and delivery location pair and the service provider utilizes its fleet of vehicles to fulfill the requests while minimizing a cost function that typically includes some travel distance component [10]. Some variants include a reconfigurable vehicle capacity to serve the needs of different users: those who use seats or those who use wheelchairs [23]. Some of the vehicle seats can be folded and stored inside the vehicle to make room for passengers in wheelchairs. Unlike this problem, the seats of the vehicle in the PD-SRP cannot be stored on the aircraft without occupying cargo capacity and are instead detached and stored at the airports.

Hatzenbühler et al. [8] studied a multi-purpose pickup and delivery problem that can deliver passengers or cargo by exchanging the module of the vehicle at a depot or special service site. Each vehicle includes a removable module and a fixed platform such that changing modules modifies the ability of the vehicle from only carrying cargo to only carrying passengers and vice versa. Compared to problems with conventional solo-purpose vehicles,

requests can be served with a fewer vehicles but at the expense of adding extra service sites and visits. We can view the core multi-purpose pickup and delivery problem as a special case of PD-SRP where the seat exchange decisions must be all-or-none: either all seats are removed to maximize cargo space or all seats are installed to maximize passenger capacity.

3 Problem Definition

In PD-SRP, we are given n requests, each potentially requiring the transportation of cargo and passenger demands. Let $V = P \cup D$ where $P = \{v_1, \dots, v_n\}$ is the set of pickup locations and $D = \{v_{n+1}, \dots, v_{2n}\}$ is the set of delivery locations. We assumed that cargo is shipped in unit-sized boxes, each having the same weight and volume. Although, in reality cargo is shipped in various shapes and weights, incorporating four-dimensional packing (i.e., combining volume and weight) would substantially complicate the problem. Therefore, similar to approximations done in practice by airlines (e.g., standard weight per passenger), we opted for this simplification.

Each request i includes picking up \hat{q}_i boxes of cargo and $\hat{\pi}_i$ passengers from location v_i and delivering them to location v_{n+i} . Thus, the demand of the corresponding delivery location has an equal magnitude negative value (i.e., $-\hat{q}_i = \hat{q}_{i+n}$, $-\hat{\pi}_i = \hat{\pi}_{i+n}$, $\forall i \in P$). Note that this representation can model more complex patterns (e.g., requests that share pickup or delivery locations but not both) by copying locations for each unique pickup-delivery pair.

When an aircraft is at its maximum seat capacity, it has \hat{S} seats and can carry \hat{C} boxes of cargo. By removing a seat, L boxes of cargo capacity are added to the aircraft. Therefore, the maximum cargo capacity when removing all the seats is $K = \hat{S}L + \hat{C}$. Each location i starts with S_i^0 stored seats and therefore the aircraft can add at most $\min(\hat{S}, S_i^0)$ seats or remove at most \hat{S} seats when visiting location i . There is no maximum number of seats that can be stored at a given location.

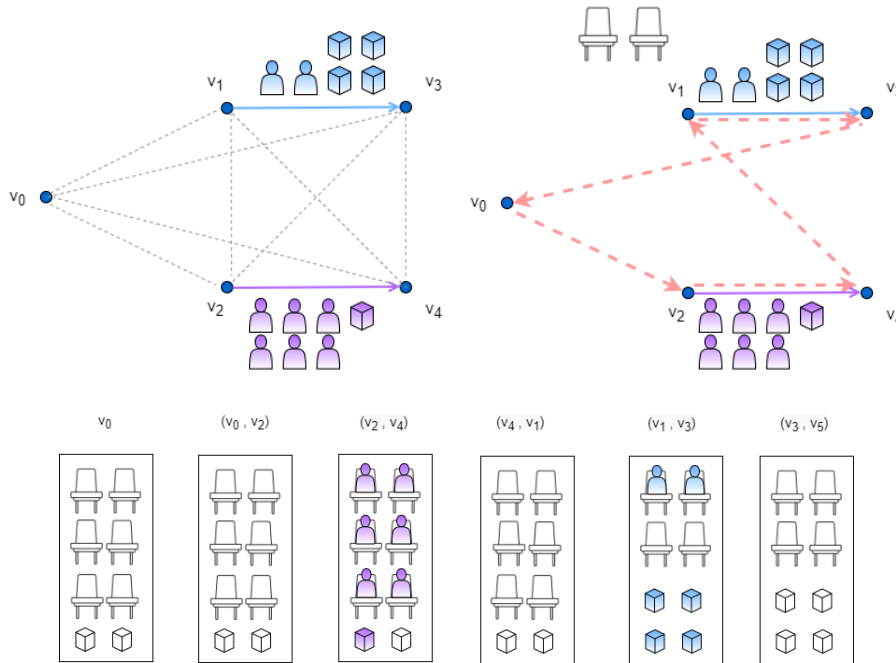
In order to represent the problem as a path, two nodes are assigned to the depot: v_0 is the start node and v_{2n+1} is the end node. For modeling purposes we define sets $V_{N+1} = V \cup \{v_{2n+1}\}$, $V_0 = V \cup \{v_0\}$ and $V_{0,N+1} = V \cup \{v_0, v_{2n+1}\}$. Therefore, the problem is defined on graph $G = (V_{0,N+1}, A)$ where $A = \{(i, j) | i, j \in V_{0,N+1}, i \neq j\}$ with each arc having an associated distance, d_{ij} . The vehicle is initially at the depot v_0 with a cargo and passenger capacity of \tilde{C}_0 and \tilde{S}_0 where $\tilde{C}_0 = K - L\tilde{S}_0$ and $\tilde{S}_0 \leq \hat{S}$, respectively, and must finish the trip at depot v_{2n+1} . We assume that the start and end nodes are not the pickup or delivery location of any requests. Again, this assumption is not limiting as such requests can be represented by adding extra nodes at the same location as the start and end nodes.

In PD-SRP we aim to minimize the travel distance while deciding how many seats to add or remove at each location to fulfill all the requests while respecting capacities. The PD-SRP is NP-hard because if we fix the seat decisions and set all the demands to zero, the problem can be reduced to TSP which is known to be NP-hard [13].

An instance of this problem can be seen in Figure 1. The optimal tour is shown in pink, and the seat icon near each vertex represents the number of seats stored at the corresponding base. The optimal tour for this instance is $(v_0, v_2, v_4, v_1, v_3, v_5)$ with two seats left at v_1 .

4 Methods

We develop six models for the PD-SRP using constraint programming (CP), mixed integer programming (MIP), and domain independent dynamic programming (DIDP). One of the MIP models solves a restricted version of the PD-SRP and is used to warm-start the CP model and the two other MIP models. In this section, we describe each of the models in detail.



■ **Figure 1** Example of an PD-SRP instance with 2 requests. The optimal tour is shown by dotted pink edges. In the aircraft configuration, white seats and boxes show the current passenger and cargo capacity, respectively. The colored seats and boxes show the corresponding cargo and passenger requests that are picked up.

4.1 A Constraint Programming Model (CP)

Our CP model equates distance and time and, thus, uses a one-machine scheduling approach where jobs correspond to the visits and the setup times between two consecutive jobs correspond to the distance between two locations. The model uses $|V_{0,2n+1}|$ interval variables x_i that represent visits to each location, and a sequence variable, π , that constrains interval variables to form a sequence with an extra end node representing the return to the depot. The size of the interval variable is 0 because there is no service time associated with the visits. For every location $i \in \{0, \dots, 2n\}$, variable s_i is introduced to represent the number of seats that are added or removed. The formulation of the CP model is presented in Figure 2. Note that CP model is written in CP Optimizer language.

The objective function is the minimization of the total distance traveled by the aircraft. $\text{EndOf}(x_{2n+1})$ corresponds to the end-point of the last interval variable in the sequence variable π : the time (i.e., total distance travelled) when the aircraft returns to the depot. Constraint (1a) ensures that each pair of consecutive interval variables is scheduled with a transition time equal to at least the required travel distance between the two corresponding locations. Constraint (1b) enforces that the pickup location of each request is visited before the delivery location. Constraint (1c) specifies that the aircraft begins and ends at the start and end depot locations.

We used three cumulative functions to represent the following values that are potentially changed by each interval variable (aircraft visits): available cargo space, number of empty seats, and the total number of seats. In particular, cumulative functions (1d) and (1f) are used to represent the available passenger and cargo space as the trip proceeds. H represents the number of empty seats in the aircraft (i.e., the available passenger space) and C represents

$$\begin{aligned}
\min \text{EndOf}(x_{2n+1}) & \tag{CP} \\
\text{s.t. NoOverlap}(\pi, \{d_{i,j} : (i,j) \in A\}) & \tag{1a} \\
\text{EndBeforeStart}(x_i, x_{n+i}) & \forall i \in \{1, \dots, n\} \tag{1b} \\
\text{First}(\pi, x_0), \text{Last}(\pi, x_{2n+1}) & \tag{1c} \\
C = \text{StepAt}(x_0, \tilde{C}_0) + \sum_{i=0}^{2n} \text{StepAtStart}(x_i, -\hat{q}_i - L \cdot s_i) & \tag{1d} \\
C \geq 0 & \tag{1e} \\
H = \text{StepAt}(x_0, \tilde{S}_0) + \sum_{i=0}^{2n} \text{StepAtStart}(x_i, -\hat{\pi}_i + s_i) & \tag{1f} \\
H \geq 0 & \tag{1g} \\
S = \text{StepAt}(x_0, \tilde{S}_0) + \sum_{i=0}^{2n} \text{StepAtStart}(x_i, s_i) & \tag{1h} \\
S \leq \hat{S} & \tag{1i} \\
x_i : \text{intervalVar}(0) & \forall i \in V_{0,2n+1} \tag{1j} \\
s_i : \text{integerVar}(-\hat{S}, \min(\hat{S}, S_i^0)) & \forall i \in V_0 \tag{1k} \\
\pi : \text{sequenceVar}(x_0, \dots, x_{2n+1}) & \tag{1l}
\end{aligned}$$

■ **Figure 2** The CP Model for the PD-SRP.

the available cargo space. Before the trip starts, $K = H + C$ and, if there are \hat{S}_0 seats in the aircraft at the start, $H = L \cdot \hat{S}_0$. The expression $\text{StepAtStart}(var, impact)$ specifies the change (increment or decrement) to the cumulative function at the start of an interval variable. The available cargo space C will decrease as cargo and seats are picked up, therefore we use $\text{StepAtStart}(x_i, -\hat{q}_i - L \cdot s_i)$ to represent the changes to available cargo space at each location $i \in \{1, \dots, 2n\}$. The available passenger space will decrease when cargo is picked up, while increasing when adding seats as represented by $\text{StepAtStart}(x_i, -\hat{\pi}_i + s_i)$ at each location $i \in \{0, \dots, 2n\}$. The cumulative function S is introduced in constraint (1h) to describe the change of the total number of seats in the aircraft. S will change with the number of seats being added or removed as represented by s_i . Constraint (1i) restricts the total number of seats by the maximum seat capacity \hat{S} . In constraint (1k), the domain of s_i is $[-\hat{S}, S_i^0]$ reflecting the range of the number of seats that the aircraft can remove or add at location i .

It should be noted that every interval variable contributes to the cumulative constraint, which means that these bounds are maintained throughout the sequence. Therefore, we do not need to have a separate cumulative function for every location of the tour.

4.2 Mixed Integer Programming Models

In this section, we describe three MIP models motivated by existing models for pickup and delivery problems. The first two models exactly represent the PD-SRP and therefore admit optimal solutions. The final model is a restriction of the PD-SRP problem that can be used to quickly find a feasible solution and, therefore, an upper bound for the PD-SRP. In our experiments, we investigate the use of this restricted model to warm-start the CP model and two other MIP models.

$$\begin{aligned}
 \min \quad & \sum_{i \in V_0} \sum_{j \in V_{N+1}, i \neq j} d_{ij} x_{ij} && (\text{MIIIP}_{loc}) \\
 & \sum_{j \in V_{N+1}} x_{ij} = 1 && i \in V_0 \quad (2a) \\
 & \sum_{j \in V_0, j \neq i} x_{ji} - \sum_{j \in V_{N+1}, i \neq j} x_{ij} = 0 && i \in V \quad (2b) \\
 & \tau_i + x_{ij} - |V|(1 - x_{ij}) \leq \tau_j && i \in V_0, j \in V_{N+1}, i \neq j \quad (2c) \\
 & 1 \leq \tau_i \leq |V| && i \in V \quad (2d) \\
 & \tau_i + 1 \leq \tau_{n+i} && i \in P \quad (2e) \\
 & y_i + \pi_i \leq \hat{S} && i \in V_0 \quad (2f) \\
 & y_i + \pi_i + s_i \leq \hat{S} && i \in V_0 \quad (2g) \\
 & \pi_i + \hat{\pi}_i \leq \hat{S} && i \in V_0 \quad (2h) \\
 & u_i + q_i + Ly_i + L\pi_i \leq K && i \in V_0 \quad (2i) \\
 & u_j \leq u_i - Ls_i - \hat{q}_i x_{ij} + (2K)(1 - x_{ij}) && i \in V, j \in V_{N+1}, i \neq j \quad (2j) \\
 & u_j \geq u_i - Ls_i - \hat{q}_i x_{ij} - (2K)(1 - x_{ij}) && i \in V, j \in V_{N+1}, i \neq j \quad (2k) \\
 & y_j \leq y_i + s_i - \hat{\pi}_i x_{ij} + 2\hat{S}(1 - x_{ij}) && i \in V, j \in V_{N+1}, i \neq j \quad (2l) \\
 & y_j \geq y_i + s_i - \hat{\pi}_i x_{ij} - 2\hat{S}(1 - x_{ij}) && i \in V, j \in V_{N+1}, i \neq j \quad (2m) \\
 & \pi_j \geq \pi_i + \hat{\pi}_i x_{ij} - \hat{S}(1 - x_{ij}) && i \in V, j \in V_{N+1}, i \neq j \quad (2n) \\
 & q_j \geq q_i + \hat{q}_i x_{ij} - K(1 - x_{ij}) && i \in V, j \in V_{N+1}, i \neq j \quad (2o) \\
 & -y_i \leq s_i \leq \min(\hat{S}, \mathcal{S}_i^0) && i \in V \quad (2p) \\
 & \tau_0 = \pi_0 = q_0 = 0, u_0 = \tilde{C}_0, y_0 = \tilde{S}_0 && (2q) \\
 & x_{ij} \in \{0, 1\} && i \in V_0, j \in V_{N+1}, i \neq j \quad (2r) \\
 & u_i, y_i, \pi_i, q_i, \tau_i \in \mathbb{R}^{0+}, s_i \in \mathbb{R} && i \in V_{0, N+1} \quad (2s)
 \end{aligned}$$

■ **Figure 3** The MIIIP_{loc} Model for the PD-SRP.

4.2.1 Two-indexed Location-Based MIP (MIIIP_{loc})

We propose a two-indexed location-based MIP model for PD-SRP (MIIIP_{loc}) based on a model for an existing pickup and delivery variant [7]. In MIIIP_{loc} , x_{ij} is a binary variable that is 1 if arc $(i, j) \in A$ is traveled and is 0 otherwise. Non-negative continuous variables τ_i , u_i , and y_i represent the distance, available cargo space, and empty seats, respectively, on arrival at vertex $i \in V_{0, N+1}$. As above, let variable s_i be the number of seats that are added ($s_i > 0$) or removed ($s_i < 0$) at location i . Finally, let π_i and q_i be the number of passengers and boxes of cargo on the aircraft on arrival at vertex $i \in V_{0, N+1}$, respectively. The MIIIP_{loc} model is shown in Figure 3.

The objective function minimizes the total distance traveled. Constraint (2a) ensures that each customer is visited exactly once while constraint (2b) forces an arrival and departure at each non-depot vertex. Constraints (2c) and (2d) prevent the formation of the subtours, using Miller-Tucker-Zemlin (MTZ) constraints [17]. Constraint (2e) forces the aircraft to visit the pickup location of each commodity before the delivery location. Constraints (2f) and (2g) respectively ensure that the total number of seats before and after adding or removing

new seats does not exceed the passenger capacity. Similarly, constraint (2h) ensures that the total number of passengers on the aircraft after fulfilling the demand of vertex i does not exceed the passenger capacity. Constraint (2i) enforces the relationship between u_i and y_i . Note that the left hand side of the constraint restricts the picked-up cargo and passengers to not exceed the aircraft capacity. Constraints (2j) and (2k) define the upper bound and lower bound on the available cargo space, respectively. Similarly, constraints (2l) and (2m) set the upper and lower bounds on the number of empty seats. Constraint (2n) ensures that the passenger demand is met at each location, while constraint (2o) does the same thing for the cargo demand. Constraint (2p) restricts the number of the seats that can be added based on their availability. The lower bound on the number of removed seats, when $s_i < 0$, is always the number of seats on the aircraft at the arrival of location i . Lastly, constraints (2q) - (2s) specify binary and continuous variable domains.

4.2.2 Three-indexed Rank-Based MIP (MIIP_{rank})

The three-indexed ranked-based MIP model for PD-SRP (MIIP_{rank}) is adapted from a model for the multi-commodity pickup and delivery traveling salesperson problem [3]. In MIIP_{rank} , $z_{i,j}^t$ is a binary variable indicating that aircraft goes directly from location i to location j and location i is at position t of the tour, for $i, j \in V_{0,N+1}, i \neq j, t \in \{0, \dots, 2n+1\}$. Binary variable $y_{i,t}$ is 1 if location i is visited at position t of the tour, $i \in V_{0,N+1}, t \in \{0, \dots, 2n+1\}$ and 0 otherwise. Variable s_t is the number of seats added or removed at the t 'th position of the tour, for $t \in \{0, \dots, 2n+1\}$, with a negative value corresponding to the number of seats removed. Variables w_t and u_t represent the empty seats and available cargo space on arrival at t 'th position of the tour, for $t \in \{0, \dots, 2n+1\}$. Finally, let π_t and q_t be the number of passengers and boxes of cargo on arrival at t 'th position of the tour. The MIIP_{rank} is presented in Figure 4.

The objective function minimizes the total travel distance. Constraints (3a) and (3b) ensure that tour positions are assigned to exactly one location and that each location is visited exactly once, respectively. Constraint (3c) calculates the number of empty seats just before visit t , where $\sum_{i=1}^n y_{i,t-1} \hat{\pi}_i$ is the number of passengers picked up at position $t-1$ of the tour. Similarly, constraint (3d) calculates the available cargo space just before visit t , where $\sum_{i=1}^n y_{i,t} \hat{q}_i$ is the amount of cargo picked up at position t of the tour. Constraint (3e) states that each commodity is picked up before it is delivered. Constraint (3f) enforces the relationship between w_t and u_t . The left hand side of the constraint enforces that the picked-up cargo and passengers do not exceed the available aircraft capacity. Constraint (3g) ensures that there is always \hat{C} space available for cargo on the aircraft. From (3f) and (3g) we can conclude that $\pi_t + w_t \leq \hat{S}$: the total number of seats does not exceed the passenger capacity. Constraint (3h) ensures the feasibility of the number of seats to be added or removed. Constraints (3i) and (3j) calculate the number of passengers and boxes of cargo at each position of the tour, respectively. Constraints (3k) and (3l) enforce the relationship between y and z variables and, together with (3e) and (3m), prevent the formation of subtours in a MTZ fashion. Lastly, constraints (3n) - (3q) specify the domains of the variables.

4.2.3 Upper bound MIP Model (MIIP_{UB})

Our preliminary experiments suggested that the CP and MIP models presented above struggled to find good feasible solutions. We, therefore, investigate the use of a third MIP model, designed to quickly find an upper bound on the PD-SRP by solving a restriction of the full problem. Such a model provides a heuristic solution as well as a potential warm-start solution for the complete models.

$$\begin{aligned}
 \min \quad & \sum_{t=0}^{2n} \sum_{i \in V_{0,N+1}} \sum_{j \in V_{0,N+1}, j \neq i} d_{i,j} z_{i,j}^t & (\text{MIP}_{rank}) \\
 \sum_{i \in V_0} y_{i,t} = 1 & & t \in \{0, \dots, 2n\} & (3a) \\
 \sum_{t=1}^{2n} y_{i,t} = 1 & & i \in V & (3b) \\
 w_t = w_{t-1} + s_{t-1} - \sum_{i=1}^n y_{i,t-1} \hat{\pi}_i & & t \in \{1, \dots, 2n+1\} & (3c) \\
 u_t = u_{t-1} - Ls_{t-1} - \sum_{i=1}^n y_{i,t-1} \hat{q}_i & & t \in \{1, \dots, 2n+1\} & (3d) \\
 \sum_{t=1}^n ty_{i,t} - \sum_{t=1}^n ty_{n+i,t} \leq -1 & & i \in P & (3e) \\
 q_t + u_t + Lw_t + L\pi_t \leq K & & t \in \{0, \dots, 2n+1\} & (3f) \\
 q_t + u_t \geq \hat{C} & & t \in \{0, \dots, 2n+1\} & (3g) \\
 -w_t \leq s_t \leq \min(\hat{S}, \sum_{i=1}^n S_i^0 y_{i,t}) & & t \in \{0, \dots, 2n+1\} & (3h) \\
 \pi_t = \pi_{t-1} + \sum_{i=1}^n y_{i,t-1} \hat{\pi}_i & & t \in \{1, \dots, 2n+1\} & (3i) \\
 q_t = q_{t-1} + \sum_{i=1}^n y_{i,t-1} \hat{q}_i & & t \in \{1, \dots, 2n+1\} & (3j) \\
 y_{i,t} - \sum_{j=0}^n z_{i,j}^t = 0 & & i \in V_0, t \in \{0, \dots, 2n\} & (3k) \\
 y_{j,t} - \sum_{i=0}^n z_{i,j}^{t-1} = 0 & & j \in V_{0,N+1}, t \in \{1, \dots, 2n+1\} & (3l) \\
 y_{0,0} = y_{2n+1,2n+1} = 1, y_{0,t} = 0 & & t \in \{1, \dots, 2n\} & (3m) \\
 s_t \leq \hat{S}, w_t \leq \hat{S}, u_t \leq K & & t \in \{0, \dots, 2n+1\} & (3n) \\
 u_0 = \tilde{C}_0, w_0 = \tilde{S}_0, \pi_0 = q_0 = 0 & & & (3o) \\
 y_{i,t} \in \{0, 1\}, z_{i,j}^t \in \{0, 1\} & & i, j \in V_{0,N+1}, t \in \{0, \dots, 2n+1\} & (3p) \\
 u_t, w_t, \pi_t, q_t \in \mathbb{R}^{0+}, s_t \in \mathbb{R} & & t \in \{0, \dots, 2n+1\} & (3q)
 \end{aligned}$$

■ **Figure 4** A Three-Indexed MIP Model for the PD-SRP.

$$\begin{aligned}
\min \quad & \sum_{i \in P_0} \sum_{j \in P_0, i \neq j} c_{i,j} x_{i,j} && (\text{MIP}_{UB}) \\
\text{s.t.} \quad & \sum_{j \in P_0, i \neq j} x_{i,j} = 1 && \forall i \in P \quad (4a) \\
& \sum_{i \in P_{N+1}, i \neq j} x_{j,i} - \sum_{i \in P_0, i \neq j} x_{i,j} = 0 && \forall j \in P \quad (4b) \\
& t_i + x_{i,j} - |P|(1 - x_{i,j}) \leq t_j && \forall i \in P_{N+1}, j \in P_{N+1}, i \neq j \quad (4c) \\
& s_j \leq s_i + (S_{i+n}^0 + S_j^0)x_{i,j} + |\hat{S}|(1 - x_{i,j}) && \forall i \in P_0, j \in P_{N+1}, i \neq j \quad (4d) \\
& Ls_i + \hat{q}_i \leq K && \forall i \in P_{N+1} \quad (4e) \\
& 1 \leq t_i \leq |P| && \forall i \in P \quad (4f) \\
& s_i \leq \hat{S} && \forall i \in P \quad (4g) \\
& s_i \geq \hat{\pi}_i && \forall i \in P \quad (4h) \\
& t_0 = 0 && (4i) \\
& \tilde{S}_0 \leq s_0 \leq \tilde{S}_0 + S_0^0 && (4j) \\
& x_{i,j} \in \{0, 1\} && \forall i \in P \quad (4k) \\
& t_i \in \mathbb{N}, s_i \in \mathbb{N} && \forall i \in P_{0,N+1} \quad (4l)
\end{aligned}$$

■ **Figure 5** The Upper Bound MIP model for a restriction of PD-SRP.

The upper bound model is obtained by over-constraining the original problem to require that a request must be delivered immediately after being picked up. The nodes in this problem include the start depot v_0 , the end depot v_{N+1} , and all the pickup nodes $P = \{v_1, \dots, v_n\}$. For modeling purposes we define sets $P_{N+1} = P \cup \{v_{N+1}\}$, $P_0 = P \cup \{v_0\}$ and $P_{0,N+1} = P \cup \{v_0, v_{N+1}\}$. The delivery nodes are not explicitly included because each origin-to-destination trip takes place immediately after the visit to the pickup node with the total distance increased by both the travel to the pickup node and the travel between the pickup node and the delivery node.

The MIP_{UB} model is presented in Figure 5. Let $x_{i,j}$ be a binary variable indicating that the aircraft goes from the delivery location of the request i to the pickup location of request j . Let s_i be the number of seats in the aircraft right after visiting location i . Finally, let t_i be the position of location i on the tour. The solution returned by this model is likely to be sub-optimal for the PD-SRP.

The objective function minimizes the total distance traveled. The coefficient $c_{i,j}$ represents the total distance starting from the delivery location of request i , visiting the pickup location of request j , and then travelling to the delivery locations of request j . Request 0 is to travel from the depot to the pickup location of the first request. The delivery and pickup locations of request 0 are nodes v_0 and v_{2n+1} , respectively.

Constraints (4a) and (4b) ensure that each node is visited exactly once. Constraints (4c), (4f), and (4i) prevent the formation of subtours. Constraint (4d) describes seat changes when the aircraft visits a node and constraint (4e) requires that the space taken up by the seats in the aircraft must be less than or equal to the remaining space after picking up the cargo of the current request. Constraint (4g) restricts the number of seats to never surpasses the maximum number of seats allowed in the aircraft and constraint (4h) ensures that the number of seats in the aircraft never drops below the number of passengers to be picked up. Constraints (4j) - (4l) specify the domains of the variables.

4.3 Domain-Independent Dynamic Programming Models

Domain-Independent Dynamic Programming (DIDP) is a recently proposed methodology for solving combinatorial optimization problems by formulating the problem as state-based dynamic program (DP) and using a generic solver to solve it [14]. DP models are declaratively formulated in Dynamic Programming Description Language (DyPDL), a solver-independent modeling formalism for DP that is inspired by AI planning. In DyPDL, a model consists of the following:

- *state variables*: variables that take on numeric, set, or set-element values that define the states in the search space of the problem
- *target state*: the problem state for which the optimal value is to be computed by the recursive formulation
- *constants*: state-independent values
- *transitions*: decisions in the DP that move between states
- *base cases*: a set of conditions that define states that terminate the recursion
- *state constraints*: conditions that must be satisfied by all states
- *dual bound*: an optional lower (upper) bound on the objective function for minimization (maximization) problems.

We developed two DIDP models for the PD-SRP.

4.3.1 A Two-transition DIDP Model (DIDP_{2T})

Our first DIDP model has two types of transition: one to represent adding or removing seats and picking up or delivering cargo and passengers and a second to model moving the aircraft to a different location. In the model, a state is a tuple $\langle U, i, q, \pi, s, \alpha \rangle$, which represents the set of unvisited vertices, U , the current location, i , the cargo load, q , the number of passengers, π , the number of seats, s , and a flag representing which type of transition to apply, α . We set $\alpha = 1$ if we have finished pickup/delivery at a location to indicate that the next transition should be to move the aircraft. Otherwise, $\alpha = 0$.

The DIDP_{2T} model is defined in Figure 6. We focus first on Eqs. (5c) and (5d), which respectively define the possible seat changes and possible next locations at a location i .

Suppose that the number of seats at the current location i is increased by δ . Since there are S_i^0 seats stored at each location initially, when the aircraft has s seats, at i we can add at most $\min\{S_i^0, \hat{S} - s\}$ seats and remove at most s seats. For simplicity we will denote $\hat{S}_i = \min\{S_i^0, \hat{S} - s\}$. Therefore, $\delta \in [-s, \hat{S}_i]$. Let numeric constants w_i and u_i be the net change of cargo and passengers at location i , respectively. The cargo will be increased by w_i , so the current cargo will become $q + w_i \leq K - (s + \delta)L$, the current space for cargo. Similarly, the number of passengers will be $\pi + u_i \leq s + \delta$. Lastly, δ must only take integer values. With these conditions, Eq. (5c) specifies the values of δ .

Consider visiting the next location, j , from current location i . To be a valid location to visit next, j must be unvisited ($j \in U$), it must be connected by an edge in the graph to i ($(i, j) \in A$), and it must be either a pickup location ($j \notin D$) or its corresponding pickup location must have already been visited. If we let p_j be the pickup location for the request whose delivery location is j , then this final condition is: $p_j \notin U$. Eq. (5d) represents the candidate locations to visit next after current location i .

The objective function specifies the state for which the optimal cost needs to be computed: the state where all pickup and delivery nodes are unvisited, the current location is the start depot (v_0), the cargo and passenger loads are 0, the aircraft has \tilde{S}_0 seats, and the next

$$\begin{aligned}
& \text{compute } Z(V, 0, 0, 0, \tilde{S}_0, 0) && (\mathbb{DIDP}_{2T}) \\
Z(U, i, q, \pi, s, \alpha) &= \begin{cases} d_{i,2n+1} & \text{if } U = \emptyset, \alpha = 1 \\ \min_{\delta \in T(q, \pi, s, i)} Z(U, i, q + w_i, \pi + u_i, s + \delta, 1) & \text{if } U \neq \emptyset, \alpha = 0 \\ \min_{j \in R(U, i)} d_{i,j} Z(U \setminus \{j\}, j, q, \pi, s, 0) & \text{if } U \neq \emptyset, \alpha = 1 \end{cases} && (5a) \\
Z(U, i, q, \pi, s, \alpha) &\geq 0 && (5b) \\
T(i, q, \pi, s) &= \left\{ \delta \in [-s, \hat{S}_i] \mid q + w_i \leq K - (s + \delta)L \wedge \pi + u_i \leq s + \delta, \delta \in \mathbb{Z} \right\} && (5c) \\
R(U, i) &= \{j \in U \mid (i, j) \in A \wedge (j \notin D \vee p_j \notin U)\}. && (5d)
\end{aligned}$$

■ **Figure 6** The Two-transition DIDP Model (\mathbb{DIDP}_{2T}) for PD-SRP.

$$\begin{aligned}
& \text{compute } Z(V, 0, 0, 0, \tilde{S}_0) && (\mathbb{DIDP}_{1T}) \\
Z(U, i, q, \pi, s) &= \begin{cases} d_{i,2n+1} & \text{if } U = \emptyset \wedge \exists \delta \in T(i, q, \pi, s) \\ \min_{(\delta, j) \in T(i, q, \pi, s) \times R(U, i)} d_{i,j} + Z(U \setminus \{j\}, j, q + w_i, \pi + u_i, s + \delta) & \text{if } U \neq \emptyset \end{cases} && (6a) \\
Z(U, i, q, \pi, s) &\geq 0 && (6b) \\
& \text{Eq. (5c), Eq. (5d).}
\end{aligned}$$

■ **Figure 7** The One-transition DIDP Model (\mathbb{DIDP}_{1T}) for PD-SRP.

transition should be to move the aircraft ($\alpha = 0$). In Eq. (5a), the first line computes the cost to return to the depot from node i , the second line describes the cost of adding or removing δ seats at node i , and the third line describes the cost of visiting node j from i . Note that when the aircraft is moved, the state variable α is set to 0 and if the decision regarding seats is made in this transition, α is set to 1. Constraint (5b) is a dual bound for the DIDP model which is optional but may be exploited by the solver.

4.3.2 A One-transition DIDP Model (\mathbb{DIDP}_{1T})

We present the \mathbb{DIDP}_{1T} model in Figure 7. In this model, instead of two types of transitions, we define one type that performs the pickup/delivery and seat exchange at a location and then moves the aircraft to a new location. A state is the same as in \mathbb{DIDP}_{1T} with the exception of the α flag which is no longer necessary: $\langle U, i, q, \pi, s \rangle$. As a transition first picks up or delivers cargo, passengers, and seats at the current location and then moves the aircraft to the next location, each transition corresponds to selecting (δ, j) : δ is the number of picked up seats and j is the next location to visit. The set of possible decisions at each state is therefore $T(i, q, \pi, s) \times R(U, i)$ as defined in the second line of Eq. (6a).

The objective function of \mathbb{DIDP}_{1T} defines the state for which the optimal cost is to be calculated. It is identical to the target state in \mathbb{DIDP}_{2T} with the removal of α . In Eq. (6a), the first line describes the cost of returning to the depot from node i , and the second line describes the cost of visiting node j from i . Note that the first line checks if there exists some δ such that the capacity constraints on the cargo and the passengers are satisfied. If there is no such δ , we assume $Z(\emptyset, i, q, \pi, s) = \infty$.

4.3.3 Model Sizes and Solver

In a DIDP model, we need to define all transitions that are applicable in a state. In \mathbb{DIDP}_{2T} , δ can take an integer in $[-\hat{S}, \hat{S}]$ depending on a state, so there are $2\hat{S} + 1$ candidates. We have $|V_{N+1}|$ locations to visit. Thus, \mathbb{DIDP}_{2T} requires $2\hat{S} + 1 + |V_{N+1}|$ transitions to be defined in total. In contrast, \mathbb{DIDP}_{1T} needs to define $(2\hat{S} + 1)|V_{N+1}|$ transitions but does not have state variable α . An alternative perspective is that the two DIDP models make different trade-offs between the maximum branching factor and solution length. \mathbb{DIDP}_{1T} has a branching factor of at most $(2\hat{S} + 1)|V_{N+1}|$ at each state and a solution path length of $|V_{N+1}|$. \mathbb{DIDP}_{2T} has a maximum branching factor that alternates between $2\hat{S} + 1$ and $|V_{N+1}|$ and a solution length of $2|V_{N+1}|$. The performance of a solver is affected by the number of state variables, the branching factor, and the solution length.

We solve the DIDP models with a complete anytime beam search (CABS) solver [25, 15]. CABS is an anytime algorithm meaning that seeks to quickly find a feasible solution and then to improve it in the remaining run-time. CABS employs beam search: a heuristic search algorithm that maintains a fixed number, b (beam width), of best states when exploring the search space. In CABS, beam search is performed iteratively with increasing the beam width until a stopping condition is met. Due to the iteratively increasing beam width, CABS is a complete algorithm [25].

5 Numerical Evaluation

5.1 Experimental Setup

We have developed six different models, i.e., CP, MIP_{loc} , MIP_{rank} , MIP_{UB} , \mathbb{DIDP}_{1T} , \mathbb{DIDP}_{2T} . For the experiment, we use MIP_{UB} to warm-start the MIP and CP models, producing three additional approaches: MIP_{loc_W} , MIP_{rank_W} and CP_W .

To implement and solve the models we used Python v3.8.0 and the corresponding Python interfaces to the solvers: Gurobi Optimizer 10.0.1 and gurobipy for MIP, CP Optimizer 22.1.0.0 and DOCplex for CP, and didppy 0.3.3 for DIDP.¹ Each run has a time limit of 600s. The machine used to run the experiment has Intel(R) Core(TM) i7-9700 8 core CPU @ 3.00GHz, 12MB cache, and memory of 31G.

The models are tested on randomly generated instances with sizes 4, 6, 8, 10, 12, 15, and 20 with 10 instances per size. The size of each instance is the number of requests, which is half of the number of locations. We generate problem instances randomly, approximately reflecting real-world problem size, aircraft capacity and configurations, and stored seats at each location. We fix the maximum number of seats in the aircraft $\hat{S} = 6$, the cargo-to-seat ratio $L = 100$, and the cargo capacity on a full-seat aircraft $\hat{C} = 200$. The number of seats in the aircraft start configuration, \tilde{S}_0 , is selected uniformly from $\{0, \dots, 6\}$ and the cargo capacity in the start configuration is $\tilde{C}_0 = 800 - 100\tilde{S}_0$. Similarly, the number of seats available at location i , S_i^0 , is set uniformly from $\{0, \dots, 6\}$, independently for each location. The (x, y) coordinates of every location are uniformly generated from $\{0, \dots, 100\}^2$.

We generate the passenger and cargo demand to ensure the existence of capacity-feasible solutions. For each request $i \in \{1, \dots, n\}$, there is a demand of $\hat{\pi}_i$ passengers and demand of \hat{q}_i kg cargo (i.e., \hat{q}_i/L units of cargo). We first define the total number of passengers and units of cargo as $\hat{K} = \hat{q}_i/L + \hat{\pi}_i$, and \hat{K} is uniformly generated from $\{1, \dots, \hat{S} + \hat{C}/L = 8\}$. The passenger request $\hat{\pi}_i$ is then selected uniformly from $\{0, \dots, \min(\hat{S}, \hat{K})\}$. Consequently, the cargo request is $\hat{q}_i = L(\hat{K} - \hat{\pi}_i)$.

¹ <https://didp.ai>

To compare the models, we used the number of instances solved and proved optimal, the PAR10 score time [12] (i.e., mean run-time with 10 times the time limit used if no optimal solution was proved), and mean relative error (MRE).

MRE compares the solution quality returned by each model. For an optimization problem let $obj_{t,m,i}$ be the objective value of the best solution achieved by time t of model m for instance i and let obj_i^* be the best-known objective value for that instance considering all the models. For the set of instances, \mathcal{I} , the relative error and mean relative error are computed in Eqs. (7) and (8). If a model did not find a feasible solution by a given time, the MIP_{UB} value is used to calculate a non-infinite measure.

$$RE(t, m, i) = \frac{obj_{t,m,i} - obj_i^*}{obj_i^*} \quad (7)$$

$$MRE(t, m) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} RE(t, m, i) \quad (8)$$

5.2 Results

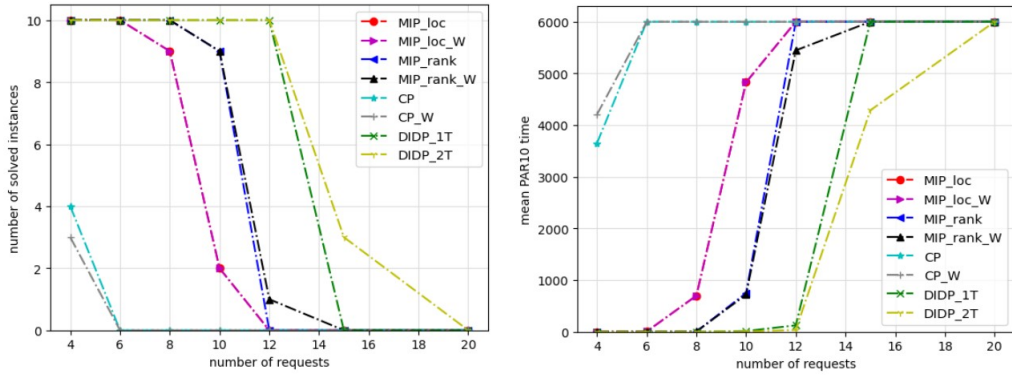
Figures 8a and 8b show the number of solved instances (i.e., proved infeasible or optimal) and mean PAR10 times for all the models. We do not include MIP_{UB} as it is incomplete, however for each model, its run-time is less than 0.02s. The run times for MIP_{loc_W} , MIP_{rank_W} , and CP_W models, include the warm-start time.

The DIDP models solved all of the instances with 12 or fewer requests, with DIDP_{2T} performing slightly better than DIDP_{1T} for instances of size 15 as it could solve three instances compared to none for DIDP_{1T} . Neither CP nor CP_W were able to solve any instances of size larger than 6 while the MIP models scaled up to size 10 or 12. There was one instance of size 4 that CP_W could not prove optimality, but CP could.

In terms of solution time, the DIDP models were the fastest and CP models were the slowest. For the MIP models, MIP_{rank_W} performed slightly better than MIP_{rank} in terms of both the number of solved instances and mean solution time. For one instance of size 12, MIP_{rank_W} proved optimality where MIP_{rank} could not.

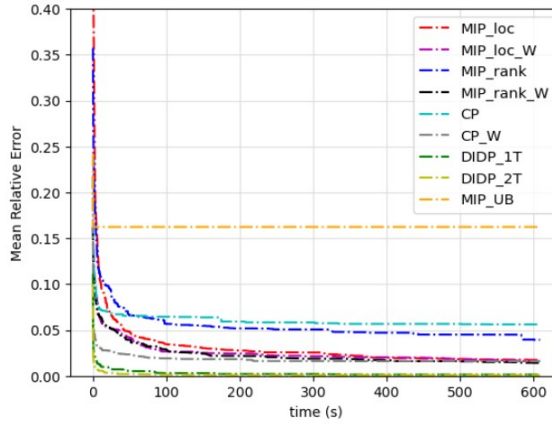
The MRE graph is shown in Figure 8c. DIDP_{2T} returns the best solutions and finds those best solutions within a few 10s of seconds. Up to 300s, CP_W outperformed MIP_{rank_W} , MIP_{loc_W} , MIP_{loc} but after that point, their solution qualities are very similar. The solution qualities returned by CP are the worst after 100 seconds. However, the use of MIP_{UB} as a warm start substantially improves CP quality especially for short run times. The performance of the MIP_{loc} and MIP_{loc_W} models was very similar, however, the MIP_{loc_W} model returned slightly better solution qualities than MIP_{loc} , especially before 200s. As we expected, the solutions found by the incomplete MIP_{UB} are substantially worse than other models.

Overall, DIDP models performed better than MIP and CP, and in particular DIDP_{2T} performed best in terms of the number of solved instances and average time to solve the instances. We hypothesize that DIDP outperforms other models due to the combination of tight capacity constraints and the precedence constraints induced by the pickup-and-delivery structure. DIDP uses these constraints to prune many transitions and, thus, reduce the search space. This result is consistent with previously observed behavior of DIDP on constrained routing problems [15] and suggests an opportunity for research to understand model characteristics that correlate with strong DIDP performance compared to other optimization approaches.



(a) Number of instances solved to optimality.

(b) Mean PAR10 time to solve instances.



(c) Comparison of MRE of all the models.

■ Figure 8 Performance of MIP, CP and DIDP models.

6 Discussion

The contributions of this work are the introduction of a novel pickup-and-delivery problem inspired by air services in northern Canada, the creation and evaluation of six optimization models in three different frameworks, and the further demonstration that the recently proposed domain-independent dynamic programming approach can out-perform incumbent techniques in a model-and-solve paradigm.

While DP models are inherently state-based, the DIDP formalism provides a novel avenue for constraint-based problem solving with connections to early ideas in CP (e.g., [6]). The DIDP models for PD-SRP are unusual as DP models due to the extensive, constraint-based, limitations on transitions (i.e., Eqs. (5c) and (5d)). While such limitations are key to strong DP performance, they are typically procedurally implemented in a problem-specific DP search algorithm. In DIDP, in contrast, constraint reasoning is used to prune transitions based on the values of state variables rather than pruning variable domains based on partial assignments. We believe that understanding this difference and developing constraint-based reasoning for this context is a fruitful research direction for CP.

Our study has a number of limitations and opportunities for further research:

- In the definition of PD-SRP, we discretized cargo into identical boxes with one size dimension (i.e., weight). In reality, cargo can take many forms from boxes of different

sizes and weights to baggage in various forms. Minimally, the volume of cargo needs to be represented. More generally, the problem should address the four-dimensional (i.e., volume plus weight) packing of heterogeneous cargo.

- We made the assumption that passengers do not have travel time restrictions. However, as a potential avenue for future research it would be interesting to incorporate additional constraints regarding how long a single passenger can be stowed in the aircraft or how long they can wait to be picked up.
- As is common in OR literature on transportation problems, our objective function is the minimization of the travel distance. A more realistic objective would represent aspects such as time and fuel consumption as well as handling and storage costs for seats.
- Most airlines run regular services with defined timetables and routings. Preliminary work indicates that determining seat exchanges is an easy problem when routes are decided. If this result bears out, there are two implications. First, we may have tools to deal with harder aspects of the real world problem including multiple aircraft, uncertain and dynamically changing demand (e.g., due to extreme weather in Canada's north), and strategic decisions about timetable creation, seat inventory, and aircraft capacities. Second, even with the version of PD-SRP presented here, we may be able to scale by exploiting the “easy” seat exchange part of the problem through Benders decomposition [2].
- Although, in this study, our focus was to design simple models that can be used “off the shelf”, it is interesting to investigate sophisticated custom-constraint CP models in the future development of this work to see if they outperform the currently developed MIP and CP models.

7 Conclusion

This paper studied a novel pickup and delivery transportation problem with reconfigurable capacities, a problem inspired by air service in northern Canada. We defined the problem formally and developed six models in three different modeling formalisms: constraint programming, mixed integer programming, and domain-independent dynamic programming. We compared the performance of the models on a set of randomly generated instances. MIP and CP models were solved with commercial solvers, the DIDP model was solved using the recently developed domain-independent dynamic programming solver [15].

Our results show that domain-independent dynamic programming models are the fastest in both finding high-quality feasible solutions to problem instances and in solving them to optimality. For large instances, when the number of requests is greater than 15, even DIDP models were not able to solve the instances to the optimality. Although in general, MIP models were faster to find feasible solutions than CP, for short run times, CP found better solutions than both of the MIP models.

Our future work will study generalizations of the problem by considering multiple aircraft and more realistic representation of cargo size and aircraft capacity. We have also embarked on a study of the decomposition of the problem both to better fit the real-world use case where routes are often predefined and to exploit the computational advances of the mathematical structure of the decomposition.

References

- 1 Maria Battarra, Jean-François Cordeau, and Manuel Iori. Chapter 6: pickup-and-delivery problems for goods transportation. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 161–191. SIAM, 2014.

17:16 Optimization Models for PDPs with Reconfigurable Capacities

- 2 J. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.
- 3 Sanjeeb Dash, Oktay Günlük, Andrea Lodi, and Andrea Tramontani. A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(1):132–147, 2012.
- 4 Irina Dumitrescu, Stefan Ropke, Jean-François Cordeau, and Gilbert Laporte. The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. *Mathematical Programming*, 121:269–305, 2010.
- 5 Pablo Factorovich, Isabel Méndez-Díaz, and Paula Zabala. Pickup and delivery problem with incompatibility constraints. *Computers & Operations Research*, 113:104805, 2020.
- 6 M. S. Fox, N. Sadeh, and C. Baykan. Constrained heuristic search. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 309–316, 1989.
- 7 Maria Gabriela S Furtado, Pedro Munari, and Reinaldo Morabito. Pickup and delivery problem with time windows: a new compact two-index formulation. *Operations Research Letters*, 45(4):334–341, 2017.
- 8 Jonas Hatzenbühler, Erik Jenelius, Gyöző Gidófalvi, and Oded Cats. Multi-purpose pickup and delivery problem for combined passenger and freight transport. *arXiv preprint arXiv:2210.05700*, 2022.
- 9 Tino Henke, M Grazia Speranza, and Gerhard Wäscher. The multi-compartment vehicle routing problem with flexible compartment sizes. *European Journal of Operational Research*, 246(3):730–743, 2015.
- 10 Sin C Ho, Wai Yuen Szeto, Yong-Hong Kuo, Janny MY Leung, Matthew Petering, and Terence WH Tou. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111:395–421, 2018.
- 11 Alexander Hübner and Manuel Ostermeier. A multi-compartment vehicle routing problem with loading and unloading costs. *Transportation Science*, 53(1):282–300, 2019.
- 12 Serdar Kadioglu, Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm selection and scheduling. In *International conference on principles and practice of constraint programming*, pages 454–469. Springer, 2011.
- 13 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.
- 14 Ryo Kuroiwa and J. Christopher Beck. Domain-independent dynamic programming: Generic state space search for combinatorial optimization. *Proceedings of the International Conference on Automated Planning and Scheduling*, 33(1):236–244, July 2023.
- 15 Ryo Kuroiwa and J. Christopher Beck. Solving domain-independent dynamic programming problems with anytime heuristic search. *Proceedings of the International Conference on Automated Planning and Scheduling*, 33:245–253, July 2023.
- 16 Jin Li, Qihui Lu, and Peihua Fu. Carbon footprint management of road freight transport under the carbon emission trading mechanism. *Mathematical Problems in Engineering*, 2015, 2015.
- 17 Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.
- 18 Andrea Mor and Maria Grazia Speranza. Vehicle routing problems over time: a survey. *Annals of Operations Research*, 314(1):255–275, 2022.
- 19 Salma Naccache, Jean-François Côté, and Leandro C Coelho. The multi-pickup and delivery problem with time windows. *European Journal of Operational Research*, 269(1):353–362, 2018.

- 20 Manuel Ostermeier, Tino Henke, Alexander Hübner, and Gerhard Wäscher. Multi-compartment vehicle routing problems: State-of-the-art, modeling framework and future directions. *European Journal of Operational Research*, 292(3):799–817, 2021.
- 21 Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery models part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58:81–117, 2006.
- 22 Yuan Qu and Jonathan F Bard. The heterogeneous pickup and delivery problem with configurable vehicle capacity. *Transportation Research Part C: Emerging Technologies*, 32:1–20, 2013.
- 23 Oscar Tellez, Samuel Vercraene, Fabien Lehuédé, Olivier Péton, and Thibaud Monteiro. The fleet size and mix dial-a-ride problem with reconfigurable vehicle capacity. *Transportation Research Part C: Emerging Technologies*, 91:99–123, 2018.
- 24 Marjolein Veenstra, Kees Jan Roodbergen, Iris FA Vis, and Leandro C Coelho. The pickup and delivery traveling salesman problem with handling costs. *European Journal of Operational Research*, 257(1):118–132, 2017.
- 25 Weixiong Zhang. Complete anytime beam search. In *AAAI/IAAI*, pages 425–430, 1998.