A Temporal-Network Constraint Programming Model for the Pickup and Delivery Problem with Time Windows

Timothy Roch

timothy.roch@umontreal.ca

March 2026

Abstract

We study the single-vehicle pickup and delivery problem with time windows (PDPTW), where each pickup must occur before its corresponding dropoff and service must begin within prescribed time windows. Classical mixed-integer linear programming (MILP) formulations typically link routing and timing through big-M constraints. We instead propose a constraint programming (CP) formulation based on successor variables, a global Circuit constraint, position variables for route order, and conditional difference constraints that enforce travel-time separations only on selected arcs. Because the route is a closed tour, we introduce a separate return-time variable instead of propagating time back into the depot.

We also interpret the active timing constraints as an evolving temporal network during search, derive necessary infeasibility conditions including subset-based time-window bounds, and present a baseline MILP formulation for comparison. Finally, we study a three-job instance built from publicly available locations, enumerate all precedence-respecting sequences, and report travel-time statistics, failure indices, and a sensitivity analysis under widened time windows.

Keywords: pickup and delivery problem, constraint programming, temporal networks, routing with time windows.

1 Introduction

The *pickup and delivery problem with time windows* (PDPTW) models a routing setting in which a single vehicle must serve a set of pickup–dropoff pairs subject to temporal and precedence constraints. Each location i is associated with a time window $[e_i, \ell_i]$ within which service must begin, and each pickup must be visited before its corresponding dropoff. The vehicle departs from a depot, visits all locations exactly once, and may wait to satisfy early time windows. We consider the variant in which the vehicle returns to the depot after completing all service, yielding a closed tour anchored at the depot. Closed routes arise naturally when vehicles must return to a base for operational or regulatory reasons. While some formulations consider open routes, the closed-tour assumption aligns directly with the global Circuit constraint used in our model.

The PDPTW combines routing, scheduling, and sequencing decisions. Mixed integer linear programming (MILP) formulations typically use binary arc variables and continuous arrival times, coupled through big-M constraints [1, 2]. Constraint programming (CP) approaches instead operate on variables with explicit domains and use propagation to eliminate infeasible partial assignments [8, 9]. Hybrid approaches combine these paradigms with metaheuristic search and relaxation techniques [10, 6].

In this work, we develop a CP formulation that avoids big-M coupling by modeling travel and service relations through *conditional difference constraints*. Successor variables define the routing structure via a global Circuit constraint, while position variables provide an explicit ordering of vertices along the tour. Temporal constraints are activated only when arcs are selected, so that arrival times are governed by a set of difference constraints that evolves during search. This interpretation allows the model to be viewed as an incrementally constructed simple temporal network (STN), which clarifies how propagation tightens time bounds and detects infeasibility.

To evaluate the formulation, we construct a reproducible three-job instance using travel times between Montreal and Quebec. We enumerate all 90 precedence-respecting permutations, analyze travel-time distributions and failure indices, and study how feasibility changes as time windows are relaxed. These experiments provide a concrete view of how routing decisions interact with temporal constraints in this setting.

2 Contributions

This paper studies the PDPTW from a constraint programming perspective and focuses on the interaction between routing decisions and temporal constraints. The main contributions are:

1. **Constraint programming formulation.** We propose a CP model based

on successor variables, a global Circuit constraint, position variables, and conditional difference constraints. The formulation represents a closed tour anchored at the depot and enforces precedence both in the route and in time without relying on big-M constants.

2. **Conditional temporal structure.** Travel-time constraints are activated only when arcs are selected, which leads to an evolving set of difference constraints during search. This provides a natural interpretation of the model in terms of temporal constraint graphs and explains how propagation tightens time bounds.

3. **Feasibility conditions.** We derive necessary conditions for feasibility, including subset-based bounds that relate travel times, service durations, and time-window widths.

4. **Empirical illustration.** We construct a small instance from publicly available data, enumerate all precedence-respecting sequences, and analyse travel times, failure indices, and sensitivity to time-window relaxation.

5. **MILP baseline.** We present a standard MILP formulation for comparison, highlighting the difference between big-M linearisation and constraint-based temporal propagation.

3 Notation and preliminaries

We review notation, difference constraints, simple temporal networks and fixpoint propagation. Throughout the paper we use the notation summarised in Table 1. Sets are denoted by uppercase letters, scalar parameters by Greek or lowercase Latin letters, and decision variables by uppercase Latin letters.

3.1 Difference constraints and temporal graphs

A *difference constraint* is an inequality of the form

$$x_j - x_i \geq c_{ij}, \tag{1}$$

where x_i and x_j are real variables and $c_{ij} \in \mathbb{R}$ is a constant. Systems of difference constraints can be represented as directed weighted graphs: vertices correspond to variables and an edge $(i \to j)$ is labelled with c_{ij}. The system (1) is consistent if and only if the graph contains no cycle whose total weight is strictly positive [12]. When the graph is acyclic, a feasible assignment can be found by topologically ordering the vertices and successively computing minimal values for x_j using longest paths. Algorithms such as Bellman–Ford or Floyd–Warshall compute feasible assignments

Table 1: Key notation used in the paper. The depot is denoted by v_0.

Symbol	Description
J	Set of jobs $\{1, \ldots, n\}$
p_k	Pickup vertex for job k
d_k	Dropoff vertex for job k
v_0	Depot vertex
V	Vertex set $\{v_0\} \cup \{p_k, d_k : k \in J\}$
A	Directed arc set $\{(i, j) : i, j \in V, i \neq j\}$
t_{ij}	Travel time from vertex i to j
s_i	Service duration at vertex i
$[e_i, \ell_i]$	Time window at vertex i
H	Global planning horizon
succ_i	Successor of vertex i in the route (CP model)
CIRCUIT$(\cdot)$	Global constraint enforcing a single Hamiltonian cycle
T_i	Start time of service at vertex i
P_i	Position index of vertex i along the tour (CP model)
x_{ij}	Binary indicator: vehicle travels from i to j (MILP)
M_{ij}	Big constant for MILP linearisation
Π	Set of precedence respecting permutations

and detect positive cycles. In scheduling, difference constraints express minimal separations between start times; for example, if task j cannot start until i finishes and a delay d, we write $T_j - T_i \geq d$.

3.2 Simple temporal networks and fixpoints

A *simple temporal network* (STN) [12] is a set of difference constraints among variables representing time points. An STN has a feasible schedule if and only if its constraint graph has no positive cycle. The earliest start times of an STN correspond to the longest paths in the graph when weights are reversed.

In classical STNs the set of difference constraints is fixed. In our CP model the temporal graph is *conditional*: an edge of the form $T_j - T_i \geq c_{ij}$ becomes part of the STN only when the routing decision $\mathsf{succ}_i = j$ is chosen. As search progresses and successors are instantiated, the STN grows incrementally. We refer to such evolving graphs as *dynamic STNs*. We use the term *dynamic STN* purely as an interpretive lens for conditional difference constraints revealed during search; we do not refer to dynamic controllability, contingent constraints, or other temporal-network extensions. Each partial assignment yields a different STN, and fixpoint propagation computes earliest and latest start times relative to that partial graph. The following lemma

summarises a fundamental property of (static) STNs.

Lemma 3.1 (Earliest start time under anchoring)**.** *Consider an STN given by difference constraints $T_j - T_i \geq c_{ij}$ on a set of real-valued time variables $(T_i)_{i \in V}$, represented by a directed weighted graph with an edge $(i \to j)$ of weight c_{ij}. Assume there is a distinguished reference vertex $r \in V$ whose time is fixed, e.g. $T_r = 0$, and that every vertex $j \in V$ is reachable from r in the constraint graph (equivalently, one may add zero-weight edges $(r \to j)$ for all $j \in V$). If the constraint graph contains no strictly positive cycle, then the componentwise-minimal feasible assignment exists and is unique. Moreover, for each $j \in V$,*

$$T_j^* \;=\; \max_{\text{directed paths } r \to j} \sum c_{pq},$$

where the sum is over edges (p, q) along the path.

Proof. Because there is no strictly positive cycle, the maximum path-weight from r to any reachable vertex is well-defined (finite). By assumption every $j \in V$ is reachable from r, so the set of directed paths $r \to j$ is nonempty and the maximum is well-defined for all j.

Define $T_r^* = 0$ and, for each $j \neq r$, define T_j^* as the maximum of $\sum c_{pq}$ over all directed paths from r to j. For any edge $(i \to j)$, every directed path from r to i can be extended by the edge $(i \to j)$, hence

$$T_j^* \;\geq\; T_i^* + c_{ij},$$

so T^* satisfies all constraints. Let T' be any feasible assignment with $T'_r = 0$. Along any directed path $r \to j$, feasibility implies $T'_j \geq \sum c_{pq}$, hence $T'_j \geq T_j^*$. Therefore T^* is componentwise-minimal, and uniqueness follows immediately. □

Lemma 3.1 highlights a central idea: once all difference constraints of an STN are known, there is a single tight way to assign the earliest possible start times. In our setting the constraint graph grows as routing decisions are made. Each partial assignment of the successor variables induces a *partial* temporal constraint graph that can be interpreted as an evolving simple temporal network (STN). Lemma 3.1 applies to the fully instantiated tour, where all conditional travel constraints are active and the temporal graph is complete.

During search, standard CP propagation operates on the currently active difference constraints by tightening variable bounds. This can be viewed as maintaining local consistency in the evolving STN. If the solver is equipped with a dedicated difference-constraints or STN consistency propagator, inconsistencies such as positive cycles (under our $\geq$ convention) can lead to early infeasibility detection when sufficient constraints are active. Otherwise, bound propagation alone still provides

substantial pruning by tightening earliest and latest feasible service times as routing decisions are fixed.

In CP, difference constraints are typically associated with domains $[e_i, \ell_i]$ for each variable. Propagation repeatedly narrows these domains by applying the updates

$$T_j \geq T_i + c_{ij}, \quad \text{so } e_j \leftarrow \max(e_j, e_i + c_{ij}), \tag{2}$$

$$T_i \leq T_j - c_{ij}, \quad \text{so } \ell_i \leftarrow \min(\ell_i, \ell_j - c_{ij}). \tag{3}$$

The process continues until no bound can be tightened or inconsistency is detected ($e_i > \ell_i$ for some i). The resulting fixpoint is analogous to the earliest assignment in Lemma 3.1.

3.3 Combinatorics of precedence respecting permutations

The number of permutations of $2n$ pickup and dropoff vertices that respect precedence is $(2n)!/2^n$. A direct pairing argument shows this: for each job, swapping the positions of its pickup and dropoff maps a precedence-violating permutation to a precedence-satisfying one, and these swaps commute across jobs. Consequently the $2n!$ permutations of the $2n$ vertices partition into 2^n equivalence classes of equal size, each class consisting of exactly those permutations that can be obtained from one another by swapping the pickup and dropoff of each job. The rapid growth of $|\Pi|$ illustrates the combinatorial explosion inherent in enumeration and motivates intelligent search and propagation.

Proposition 3.2 (Number of precedence-respecting permutations)**.** *For n pickup–dropoff pairs, the number of permutations of the $2n$ vertices that respect the precedence relations $p_k \prec d_k$ for $k = 1, \ldots, n$ is $(2n)!/2^n$.*

Proof. For each job k, define an involution τ_k that swaps the positions of p_k and d_k in a permutation. These involutions commute, so they generate an action of the group $G = \mathbb{Z}_2^n$ on the set of all $(2n)!$ permutations.

Each orbit of this action has size 2^n: applying any subset of the swaps produces a distinct permutation because p_k and d_k are distinct labels. Within each orbit, exactly one permutation satisfies all precedence constraints $p_k \prec d_k$ (namely the one in which, for every k, the pickup appears before its corresponding dropoff). Therefore the number of precedence-respecting permutations equals the number of orbits, which is $(2n)!/2^n$. □

3.4 Necessary conditions based on time windows

The following lemma yields a nontrivial necessary feasibility bound for PDPTW feasibility based on time windows and minimal travel times. It generalises a simple one-job bound to subsets of jobs.

Lemma 3.3 (Subset feasibility bound)**.** *Let $K \subseteq J$ be a nonempty subset of jobs and let*

$$V_K = \{p_k, d_k : k \in K\}.$$

Assume that t_{ij} represents shortest travel times between locations (in particular, it satisfies the triangle inequality). Define α_K as the minimum, over all precedence-respecting sequences $\sigma = (v_1, \ldots, v_{2|K|})$ that list each vertex in V_K exactly once, of the elapsed route time *from the first to the last vertex in the sequence:*

$$\alpha_K \;=\; \min_{\sigma} \left(\sum_{r=1}^{2|K|-1} \left(s_{v_r} + t_{v_r, v_{r+1}} \right) \right),$$

where the minimum ranges over all sequences σ such that for every $k \in K$, the pickup p_k appears before the corresponding dropoff d_k. Let

$$\beta_K \;=\; \max_{i \in V_K} \ell_i \;-\; \min_{i \in V_K} e_i$$

be the spread between the latest latest-time and the earliest earliest-time among vertices in V_K. If $\alpha_K > \beta_K$, then no feasible PDPTW schedule exists.

Proof. Assume there exists a feasible PDPTW schedule. Let $u = \min_{i \in V_K} T_i$ and $v = \max_{i \in V_K} T_i$ be respectively the earliest and latest *service start times* among the vertices in V_K in this schedule. By feasibility, $T_i \in [e_i, \ell_i]$ for all $i \in V_K$, hence

$$v - u \;\le\; \max_{i \in V_K} \ell_i - \min_{i \in V_K} e_i \;=\; \beta_K.$$

Now consider the subsequence of visits to vertices in V_K induced by the feasible route, in the order they are visited:

$$\sigma = (v_1, \ldots, v_{2|K|}),$$

where each $v_r \in V_K$ and the sequence respects precedence because the route does. For each r, let Δ_r be the actual elapsed time between the start of service at v_r and the start of service at v_{r+1} along the full route (which may pass through vertices outside V_K). Feasibility implies $\Delta_r \ge s_{v_r}$, and the travel portion of Δ_r is at least the shortest travel time from v_r to v_{r+1}, namely $t_{v_r, v_{r+1}}$. Therefore,

$$T_{v_{r+1}} \;=\; T_{v_r} + \Delta_r \;\ge\; T_{v_r} + s_{v_r} + t_{v_r, v_{r+1}} \qquad \text{for } r = 1, \ldots, 2|K| - 1.$$

Summing these inequalities yields

$$T_{v_{2|K|}} - T_{v_1} \;\ge\; \sum_{r=1}^{2|K|-1} \left(s_{v_r} + t_{v_r, v_{r+1}} \right) \;\ge\; \alpha_K,$$

where the last inequality holds because α_K is the minimum of that sum over all precedence-respecting sequences. Since $T_{v_1} \geq u$ and $T_{v_{2|K|}} \leq v$, we obtain $v - u \geq \alpha_K$.

Combining $v - u \leq \beta_K$ and $v - u \geq \alpha_K$ shows that feasibility implies $\alpha_K \leq \beta_K$. Hence if $\alpha_K > \beta_K$ no feasible PDPTW schedule exists. □

Lemma 3.3 highlights the role of cumulative service and travel durations in feasibility. When jobs have tight windows and long travel times between clusters, as in our illustrative instance, α_K can exceed β_K even for small subsets.

The subset bound provides an effective pruning mechanism based on job-level temporal structure. It relies on information local to the subset K and can therefore be evaluated independently of how jobs outside K are interleaved in the route, which allows it to be applied early during search. For small subsets, a CP solver can compute α_K offline using dynamic programming or lightweight heuristics and compare it to the window spread β_K during propagation. Whenever the bound is violated for some K, no ordering of those jobs within a larger tour can satisfy all time windows. In practice, the most informative subsets are pairs and triples whose pickups or dropoffs are geographically separated, as these induce large minimal travel times.

4 Problem statement

We formalise the PDPTW as follows. The input consists of n jobs $J = \{1, \ldots, n\}$, pickup vertices p_k, dropoff vertices d_k, a depot v_0, service durations $s_i \geq 0$, travel times $t_{ij} \geq 0$ for all distinct $i, j \in V$, and time windows $[e_i, \ell_i] \subseteq [0, H]$. A *schedule* assigns a permutation of the non-depot vertices $V \setminus \{v_0\}$; the vehicle departs from the depot v_0, visits each pickup and dropoff exactly once, and finally returns to the depot.

Service at each vertex $i \in V$ starts at time T_i, with $T_{v_0} \in [e_{v_0}, \ell_{v_0}]$. Timing constraints are imposed for all arcs traversed by the vehicle *except the final return to the depot*:

$$T_j \geq T_i + s_i + t_{ij} \qquad \text{for any traversed arc } (i,j) \text{ with } j \neq v_0, \tag{4}$$

$$e_i \leq T_i \leq \ell_i \qquad \text{for all } i \in V, \tag{5}$$

$$T_{d_k} \geq T_{p_k} + s_{p_k} \qquad \text{for all } k \in J, \tag{6}$$

$$p_k \text{precedes } d_k \qquad \text{in the permutation.} \tag{7}$$

The return to the depot after servicing the final vertex is modeled separately via a return-time variable, introduced explicitly in Sections 5 and 7, in order to avoid an inconsistent time cycle on the closed tour.

Consequently, the permutation together with the depot defines a closed tour anchored at the depot rather than an open path. The objective is to minimise the total travel time $\sum_{i \in V} t_{i,\mathsf{succ}_i}$, where succ_i is the successor of i in the schedule. A schedule

is *feasible* if all constraints are satisfied. The decision version asks whether a feasible schedule exists.

5 Constraint programming formulation

We now present a CP model for the PDPTW. The formulation emphasises logical correctness and uses global constraints to enforce routing structure. We divide the model into decision variables, constraints and objective.

5.1 Decision variables

- *Successor variables.* For each $i \in V$ introduce an integer variable $\mathsf{succ}_i \in V$ representing the next vertex visited after i. To disallow self loops, impose $\mathsf{succ}_i \neq i$.
- *Arrival time variables.* In practical implementations all time quantities are integral (minutes); we use real-valued domains here solely for notational convenience. For each $i \in V$ introduce a real variable T_i with initial domain $[e_i, \ell_i]$. This variable denotes the start time of service at vertex i.
- *Position variables.* For each $i \in V$ we introduce an integer variable P_i representing the position of vertex i along the tour. We fix $P_{v_0} = 0$ to anchor the depot at position zero and restrict $P_i \in \{1, \ldots, |V| - 1\}$ for all $i \in V \setminus \{v_0\}$. These variables are tied to the successor mapping by constraints described below: whenever $\mathsf{succ}_i = j$ the position increases by one, and if i is the last vertex before returning to the depot then its successor is v_0 and the position wraps around to zero. Precedence constraints will require that the pickup of each job precedes its dropoff in this order.

5.2 Global constraint for routing

To ensure that the mapping $\mathsf{succ} : V \to V$ defines a single Hamiltonian cycle visiting every vertex exactly once, we impose the global *circuit* constraint

$$\text{CIRCUIT}(\mathsf{succ}_0, \mathsf{succ}_1, \ldots, \mathsf{succ}_{|V|-1}), \tag{8}$$

where the vertices are indexed arbitrarily as $v_0, \ldots, v_{|V|-1}$. The circuit constraint is satisfied if and only if the directed graph formed by edges $(v_i, \mathsf{succ}_{v_i})$ consists of one cycle covering all vertices. Many CP solvers provide global constraints for routing structure and propagate partial assignments effectively [9].

In addition to the circuit, we maintain the integer position variables P_i described above. These variables satisfy $P_{v_0} = 0$ and, for each $i \in V$, whenever $\mathsf{succ}_i = j$ we

require $P_j \equiv P_i + 1 \pmod{|V|}$. The mod operation reflects that the successor of the last vertex in the tour is the depot with position zero. The position variables break the rotational symmetry of the circuit, support explicit precedence constraints of the form $P_{p_k} < P_{d_k}$, and help propagate ordering decisions. Combined with the circuit constraint, they ensure that the successor mapping forms a closed tour anchored at the depot and that the order of vertices is well defined up to rotation.

5.3 Route ordering and precedence constraints

The position variables are linked to the successor variables by channeling constraints. We anchor the depot at position zero and require positions to advance by one along the tour, with wrap-around at the depot. The routing structure is defined primarily by the global Circuit constraint; the position variables provide an explicit linearisation of the tour order and support precedence reasoning.

We impose the following channeling constraints:

$$P_{v_0} = 0, \tag{9}$$

$$\mathsf{succ}_i = v_0 \Rightarrow P_i = |V| - 1, \quad \forall i \in V \setminus \{v_0\}, \tag{10}$$

$$\mathsf{succ}_i = j \Rightarrow P_j = P_i + 1, \quad \forall (i,j) \in A \text{ with } j \neq v_0. \tag{11}$$

Finally, we post an AllDifferent constraint on $\{P_i : i \in V\}$ so that each vertex occupies a unique position. Together with the circuit constraint and the anchoring $P_{v_0} = 0$, these implications ensure that the position variables consistently describe the tour order induced by the successor mapping. These implications do not define the tour independently of the Circuit constraint; rather, given any fully instantiated Hamiltonian cycle, they admit a unique linearisation anchored at v_0 and are used primarily as a channeling device to support precedence propagation during search.

Precedence between pickups and dropoffs is enforced directly on positions: for each job $k \in J$ we require

$$P_{p_k} < P_{d_k}. \tag{12}$$

In the presence of time windows we also impose the temporal condition $T_{d_k} \geq T_{p_k} + s_{p_k}$; the position constraint prevents solutions that visit a dropoff before its pickup while satisfying temporal precedence through waiting.

5.4 Time propagation constraints

Because the routing variables define a *closed* Hamiltonian tour (via (8)) that includes the depot v_0, posting travel-time implications on *every* arc of the tour would create a positive cycle in the temporal constraints whenever the tour has positive total length. To model a closed tour without introducing an impossible time cycle, we treat T_{v_0} as the *start* time at the depot and introduce a separate variable for the *return* time.

Return-time variable. Introduce a real variable T_{return} with initial domain $[0, H]$ (or more generally $[e_{v_0}, H]$ if departure cannot occur before e_{v_0}) representing the time at which the vehicle returns to the depot after visiting the last vertex.

Conditional time constraints. For each ordered pair $(i, j) \in A$ with $j \neq v_0$, we impose the implication

$$\big(\mathsf{succ}_i = j\big) \;\Rightarrow\; T_j \geq T_i + s_i + t_{ij}. \tag{13}$$

For the unique arc that returns to the depot, we do *not* propagate into T_{v_0}; instead we link it to the return-time variable:

$$\big(\mathsf{succ}_i = v_0\big) \;\Rightarrow\; T_{\text{return}} \geq T_i + s_i + t_{iv_0}, \qquad \forall i \in V \setminus \{v_0\}. \tag{14}$$

Constraints (13) and (14) ensure that temporal propagation is well-defined along the tour while avoiding a contradictory positive cycle involving v_0. In CP, implication constraints of the form $(X = a) \Rightarrow Y \geq Z$ are posted as logical constraints and propagated when X is assigned.

5.5 Time window constraints

Each arrival time variable is restricted to its time window:

$$e_i \leq T_i \leq \ell_i, \quad \forall i \in V. \tag{15}$$

In addition, we anchor the schedule at the depot by fixing the depot start time:

$$T_{v_0} = e_{v_0}, \tag{16}$$

which is typically 0 in our experiments. If propagation tightens any T_i beyond ℓ_i or below e_i the branch is pruned.

5.6 Precedence constraints

For each job k we require that the dropoff start no earlier than the pickup start plus the pickup service duration:

$$T_{d_k} \geq T_{p_k} + s_{p_k}, \quad \forall k \in J. \tag{17}$$

Constraint (17) ensures that the dropoff cannot start earlier in time than the pickup. On its own, however, this temporal inequality does not guarantee that the dropoff is visited after the pickup in the tour: a solver might choose a route that visits the dropoff first but waits long enough at the pickup to satisfy the inequality. For this reason we combine the temporal inequality with the ordering constraint (12);

together they ensure correct precedence in both the route and time dimensions. The inequality is minimal in the sense that t_{p_k,d_k} does not appear; the route may travel elsewhere between p_k and d_k. When travel times satisfy the triangle inequality, (17) can be strengthened by incorporating a valid lower bound on travel from p_k to d_k, for example

$$T_{d_k} \geq T_{p_k} + s_{p_k} + t_{p_k,d_k}.$$

We do not use this optional strengthening in our baseline model but mention it for completeness.

5.7 Precedence-only optimisation variant

When time windows are ignored we can drop arrival time variables and the difference constraints associated with them. In this variant the ordering of vertices alone determines feasibility with respect to precedence. One can therefore introduce order variables $O_i \in \{0, \ldots, |V| - 1\}$ and post the implications

$$\big(\mathsf{succ}_i = j\big) \;\Rightarrow\; O_j = O_i + 1, \tag{18}$$

together with $O_{p_k} < O_{d_k}$ for all $k \in J$. The circuit constraint together with (18) ensures that order values increase along the route and that pickups precede dropoffs. In our main model we instead use the position variables P_i to link ordering and timing; this variant illustrates that the ordering constraints can be used alone when time windows play no role.

5.8 Objective

The objective is to minimise the total travel time along the tour:

$$\min \sum_{i \in V} t_{i,\mathsf{succ}_i}. \tag{19}$$

Service durations and waiting do not contribute directly to the objective, though they affect feasibility through propagation.

6 Theoretical analysis

In this section we analyse the computational complexity of the PDPTW, describe propagation properties of the CP model, and present nontrivial conditions for infeasibility and precedence only optimisation.

6.1 Feasibility complexity

The decision version of the PDPTW is NP–complete even with one vehicle. We recall the following result.

Proposition 6.1. *The problem of deciding whether a feasible schedule exists for the single vehicle PDPTW is NP–complete, even when service durations are zero.*

Proof. Membership in NP is immediate: a certificate is an ordering (equivalently, a successor mapping) of all vertices, and one can verify feasibility by propagating arrival times along the tour and checking all time windows and precedence constraints in polynomial time.

For NP–hardness we reduce from the *Hamiltonian path* problem in undirected graphs, which is NP–complete. Let $G = (U, E)$ be an undirected graph with $|U| = m$ and vertex set $U = \{1, \dots, m\}$. We construct a PDPTW instance with $n = m$ jobs. For each $k \in \{1, \dots, m\}$ create a pickup vertex p_k and a dropoff vertex d_k, and include a depot v_0. Set all service durations to zero: $s_i = 0$ for all i.

Travel times. Define symmetric travel times t_{ij} on the vertex set

$$V \;=\; \{v_0\} \;\cup\; \{p_k, d_k : k = 1, \dots, m\}$$

as follows:

$$\begin{aligned}
&t_{v_0,p_k} = 1, && \forall k,\\
&t_{p_k,p_\ell} = \begin{cases} 1, & \text{if } \{k,\ell\} \in E,\\ 2, & \text{if } \{k,\ell\} \notin E, \end{cases} && \forall k \neq \ell,\\
&t_{p_k,d_\ell} = 3, && \forall k, \ell,\\
&t_{d_k,d_\ell} = 1, && \forall k \neq \ell,\\
&t_{v_0,d_k} = 3, && \forall k.
\end{aligned}$$

Time windows. Let H be a sufficiently large horizon, e.g. $H = 10m$. Set

$$[e_{v_0}, \ell_{v_0}] = [0, 0], \qquad [e_{p_k}, \ell_{p_k}] = [0, m] \;\; \forall k, \qquad [e_{d_k}, \ell_{d_k}] = [m+1, H] \;\; \forall k.$$

Finally, impose the usual precedence constraints $p_k \prec d_k$ (equivalently $T_{d_k} \geq T_{p_k}$ since all services are zero) for all k.

Correctness of the reduction. We claim that the constructed PDPTW instance is feasible if and only if G contains a Hamiltonian path.

First note that, because every dropoff has earliest time $m + 1$ whereas every pickup has latest time m, any feasible schedule must visit *all* pickups before visiting

any dropoff. Indeed, once time exceeds m, no pickup can be served within its window, and no dropoff can be served before time $m+1$.

Thus feasibility depends entirely on whether the vehicle can visit all pickups $p_1, \ldots, p_m$ within time m, starting from the depot at time 0. Since $t_{v_0,p_k} = 1$ for every k, the first pickup starts no earlier than time 1. To serve m pickups, the route must contain exactly $m-1$ travel legs between consecutive pickups. Each such leg has length either 1 or 2. Therefore, if the route ever uses a non-edge transition between pickups (cost 2), then the arrival time at the last pickup is at least

$$1 + (m-2) \cdot 1 + 2 \;=\; m+1,$$

which violates the latest time m of the pickup windows. Consequently, in any feasible schedule every consecutive pickup-to-pickup transition must have travel time 1, which by construction means the corresponding vertices are adjacent in G. Hence the order in which pickups are visited induces a Hamiltonian path in G.

Conversely, if G has a Hamiltonian path $(u_1, u_2, \ldots, u_m)$, then consider the route that starts at v_0, visits the pickups in the order $p_{u_1}, p_{u_2}, \ldots, p_{u_m}$, and then visits all dropoffs in any order. Along the pickup segment, the travel time is $t_{v_0,p_{u_1}} = 1$ followed by $m-1$ edges of length 1, so the mth pickup is reached at time exactly m, which lies within $[0, m]$. After that, we may wait until time $m+1$ if necessary and then visit the dropoffs; since their windows start at $m+1$ and extend to H, and all remaining travel times are finite, this yields a feasible schedule satisfying all precedence constraints.

Therefore the PDPTW instance is feasible if and only if G has a Hamiltonian path. Since Hamiltonian path is NP–complete, PDPTW feasibility is NP–hard. Combined with membership in NP, the decision problem is NP–complete. □

6.2 Propagation and early infeasibility detection

Propagation through difference constraints can detect infeasibility early. A simple one-job bound (presented in previous versions) captured the idea that a single pickup–dropoff pair may be infeasible if the dropoff's latest time is earlier than the pickup's earliest time plus service. We extend this intuition to pairs of jobs.

Proposition 6.2. *Let $k, l \in J$ be distinct jobs. Suppose there exist vertices $i \in \{p_k, d_k\}$ and $j \in \{p_l, d_l\}$ such that in any feasible route i must precede j and*

$$e_i + s_i + t_{ij} > \ell_j. \tag{20}$$

Then no feasible schedule exists.

Proof. If in every feasible route i precedes j, then the earliest the vehicle can start service at j is at least $e_i + s_i + t_{ij}$. If this quantity exceeds ℓ_j, then service at j cannot

begin within its window, hence no feasible schedule exists. The condition i precedes j may be deduced from precedence relations or from the structure of optimal routing; propagation can infer such precedence dynamically based on domain reductions. □

Proposition 6.2 allows the solver to prune branches that assign a pickup or dropoff too late relative to another vertex. The condition (20) is purely local: it examines a single ordering relation between two vertices and a corresponding lower bound on travel and service time. In combination with the position-based precedence constraint (12), it ensures that if a vertex i must precede j in every feasible route, then T_i cannot be pushed arbitrarily late without violating j's window. In a dynamic STN, these bounds translate into cycles of weight violation and thus enable immediate pruning. Combined with Lemma 3.3, which considers arbitrary subsets, Proposition 6.2 provides strong necessary conditions for feasibility that can be checked before committing to a complete tour.

6.3 Precedence-only optimisation

Before analysing the combinatorial structure of precedence-only routing, it is useful to formalise its computational complexity. In the absence of time windows a feasible solution is any tour that satisfies the pickup–dropoff precedence relations, and the objective is to minimise the sum of travel times along this tour. The following proposition shows that this optimisation problem inherits the hardness of the travelling salesman problem.

Proposition 6.3. *Consider the precedence-only version of the single-vehicle PDPTW in which service durations are zero, there are no time windows, and travel times form a metric (triangle inequality). The associated decision problem—determining whether there exists a precedence-respecting closed tour of total travel time at most B—is NP–complete. Consequently, the optimisation problem is NP–hard.*

Proof. Membership in NP is immediate: a certificate is a permutation of the $2n$ vertices satisfying $p_k \prec d_k$ for all k, and its tour cost can be computed in polynomial time.

For NP–hardness we reduce from the (metric) travelling salesman problem (TSP) decision problem. Let (C, d) be a metric TSP instance with cities $C = \{1, \dots, m\}$ and distances d_{ab} satisfying the triangle inequality, and let B be the bound.

We build a precedence-only PDPTW instance with $n = m$ jobs and vertex set

$$V \;=\; \{v_0\} \;\cup\; \{p_a, d_a : a \in C\},$$

where job a corresponds to the pickup–dropoff pair (p_a, d_a) and we impose precedence constraints $p_a \prec d_a$ for all a. Service durations are zero.

Metric travel times (co-located pairs). Intuitively, p_a and d_a represent the same physical location (city a). Define a map $\phi : V \setminus \{v_0\} \to C$ by $\phi(p_a) = \phi(d_a) = a$. Fix an arbitrary depot city $1 \in C$ and set $\phi(v_0) = 1$. Define travel times for all distinct $u, w \in V$ by

$$t_{uw} \;=\; \begin{cases} 0, & \text{if } \{\phi(u), \phi(w)\} = \{a\} \text{ and } \{u, w\} = \{p_a, d_a\} \text{ for some } a, \\ d_{\phi(u),\phi(w)}, & \text{otherwise.} \end{cases}$$

In words: the distance between any two vertices is the TSP distance between their associated cities, except that moving directly between the co-located pair p_a and d_a costs 0. Because d is a metric and p_a, d_a are co-located, the resulting travel times t also satisfy the triangle inequality on V.

Key normal form. We claim that for any precedence-respecting tour on V, there exists another precedence-respecting tour of *no larger* cost in which, for every $a \in C$, the vertex d_a appears immediately after p_a.

To prove this, take any precedence-respecting tour and fix a job a. If d_a already follows p_a immediately, do nothing. Otherwise, the tour contains a subpattern

$$\cdots \to p_a \to x \to \cdots \to y \to d_a \to z \to \cdots$$

where x is the successor of p_a and y is the predecessor of d_a (so $x \neq d_a$ and $y \neq p_a$). Construct a new tour by (i) removing d_a from its current position (replacing arcs $y \to d_a$ and $d_a \to z$ by $y \to z$) and (ii) inserting d_a immediately after p_a (replacing arc $p_a \to x$ by $p_a \to d_a \to x$). Precedence is preserved because d_a is moved *earlier* but still remains after p_a.

Now compare costs. Removing d_a changes the cost by

$$t_{yz} - (t_{yd_a} + t_{d_a z}) \;\leq\; 0$$

by the triangle inequality (since $t_{yz} \leq t_{yd_a} + t_{d_a z}$). Inserting d_a after p_a changes the cost by

$$(t_{p_a d_a} + t_{d_a x}) - t_{p_a x} \;=\; 0 + d_{\phi(d_a),\phi(x)} - d_{\phi(p_a),\phi(x)} \;=\; 0$$

because $\phi(d_a) = \phi(p_a) = a$ and $t_{p_a d_a} = 0$. Hence the total tour cost does not increase. Repeating this operation for each job yields the claimed normal form.

Equivalence to TSP. In the normal form, the tour alternates

$$v_0,\, p_{u_1}, d_{u_1},\, p_{u_2}, d_{u_2},\, \ldots,\, p_{u_m}, d_{u_m},\, v_0$$

for some ordering $(u_1, \ldots, u_m)$ of the cities. Because $t_{p_{u_i}, d_{u_i}} = 0$ and $t_{d_{u_i}, p_{u_{i+1}}} = d_{u_i, u_{i+1}}$, the tour cost equals

$$d_{1,u_1} + \sum_{i=1}^{m-1} d_{u_i, u_{i+1}} + d_{u_m, 1},$$

which is exactly the cost of the corresponding TSP tour on C starting and ending at city 1. Therefore, there exists a precedence-respecting tour of cost at most B if and only if the original TSP instance has a tour of cost at most B.

Thus the precedence-only PDPTW decision problem is NP–hard. Combined with membership in NP, it is NP–complete, and the optimisation version is NP–hard. □

When time windows are relaxed, the PDPTW reduces to finding a tour that respects precedence and minimises travel time. As shown in Proposition 6.3, this problem is NP–hard. However, structure can still be exploited. For example, if travel times satisfy the triangle inequality, then any minimal tour under precedence cannot contain crossing arcs: if the route visits p_k, p_l, d_k, d_l in that order with $k \neq l$, then one can exchange d_k and p_l to reduce travel time while maintaining precedence. Such properties motivate neighbourhood heuristics such as the 2–exchange moves used in metaheuristics [6]. In our empirical analysis we examine the minimal precedence tour and its schedule length.

The precedence-only variant is instructive because it isolates the combinatorial structure of the routing problem. Without time windows, a solution is determined entirely by the permutation of pickups and dropoffs; the arrival times can be computed deterministically once the route is fixed. This separation allows us to study properties such as the symmetry of feasible permutations, the presence of forbidden patterns and the benefit of local exchange moves without the additional complexity of temporal windows. Moreover, lower bounds on the precedence-only problem provide bounds on the full PDPTW, since any feasible time-respecting schedule must respect precedence and has travel time at least as large as the precedence-only optimum.

7 Mixed integer linear programming formulation

To contrast the CP representation, we present a canonical MILP formulation for the PDPTW. Let $x_{ij} \in \{0, 1\}$ be a binary variable indicating whether the vehicle travels from vertex i to j. Let $T_i \in \mathbb{R}$ denote the service start time at vertex i. Because the route is a closed tour that returns to the depot, propagating time around the entire cycle would create an impossible positive time cycle. We therefore treat T_{v_0} as the departure time from the depot and introduce a separate return-time variable $T_{\text{return}} \in \mathbb{R}$ representing the time at which the vehicle returns to the depot after the final visit. Let $u_i \in \mathbb{Z}$ be a position variable for subtour elimination (Miller–Tucker–

Zemlin formulation). The MILP is:

$$\min \quad \sum_{(i,j)\in A} t_{ij}x_{ij} \tag{21}$$

$$\text{s.t.} \quad \sum_{j\in V\setminus\{i\}} x_{ij} = 1 \qquad \forall i \in V \setminus \{v_0\}, \tag{F1}$$

$$\sum_{i\in V\setminus\{j\}} x_{ij} = 1 \qquad \forall j \in V \setminus \{v_0\}, \tag{F2}$$

$$\sum_{j\in V\setminus\{v_0\}} x_{v_0 j} = 1, \quad \sum_{i\in V\setminus\{v_0\}} x_{i v_0} = 1, \tag{F0}$$

$$u_i - u_j + (2n+1)x_{ij} \le 2n \qquad \forall (i,j) \in A,\ i \neq v_0,\ j \neq v_0, \tag{F3}$$

$$T_j \ge T_i + s_i + t_{ij} - M_{ij}(1 - x_{ij}) \qquad \forall (i,j) \in A \text{ with } j \neq v_0, \tag{F4a}$$

$$T_{\text{return}} \ge T_i + s_i + t_{iv_0} - M_i^{\text{return}}(1 - x_{iv_0}) \qquad \forall i \in V \setminus \{v_0\}, \tag{F4b}$$

$$e_i \le T_i \le \ell_i \qquad \forall i \in V \setminus \{v_0\}, \tag{F5}$$

$$T_{v_0} = e_{v_0}, \tag{F5a}$$

$$e_{v_0} \le T_{\text{return}} \le H, \tag{F5b}$$

$$T_{d_k} \ge T_{p_k} + s_{p_k} \qquad \forall k \in J, \tag{F6}$$

$$x_{ij} \in \{0,1\},\ u_i \in \{1,\dots,2n\} \qquad \forall i \in V \setminus \{v_0\},\ \forall j \in V, \text{ and fix } u_{v_0} = 0. \tag{F7}$$

where M_{ij} is a big constant satisfying $M_{ij} \ge \ell_j - e_i - s_i - t_{ij}$ for $j \neq v_0$, and M_i^{return} satisfies $M_i^{\text{return}} \ge H - e_i - s_i - t_{iv_0}$. Constraints (F1)–(F2), together with (F0), ensure exactly one outgoing and one incoming arc at each vertex, including the depot. Subtour elimination constraints (F3) prevent cycles that do not include the depot by assigning ordering variables u_i. Timing constraints (F4a)–(F4b) couple service start times to routing decisions while avoiding a positive time cycle on the closed tour. Precedence constraint (F6) ensures that each pickup precedes its corresponding dropoff in time. The big constants M_{ij} can be tightened based on time windows and travel times; poor calibration results in weak relaxations and slow convergence [2, 4]. Strengthening families include time window cuts that use precedence relationships and time windows to derive additional inequalities, and precedence cuts that enforce an order on certain pairs of jobs [4]. Exact algorithms often embed the MILP formulation in a branch-and-price scheme that generates columns corresponding to feasible routes and solves a master problem via cutting planes [4].

8 Empirical analysis

We now present an empirical study of a realistic three job instance. Our goal is to illustrate the structural properties of the search space, the behaviour of propagation and the impact of time window modifications. All data are defined within this section so that the reader can reproduce the results independently.

8.1 Data construction

The instance comprises three jobs ($n = 3$). Table 2 specifies the pickup and dropoff addresses, service durations and time windows. The depot coincides with the pickup of Job 1.

Depot co-location convention. The depot v_0 is co-located with the pickup vertex p_1. Accordingly, travel times involving the depot are defined by

$$t_{v_0,j} := t_{p_1,j} \quad \text{and} \quad t_{j,v_0} := t_{j,p_1} \qquad \forall j \in \{p_1, d_1, p_2, d_2, p_3, d_3\},$$

with $t_{v_0,p_1} = t_{p_1,v_0} = 0$. All route simulations and statistics reported below use this convention. Travel times between the six non depot vertices are given in Table 3. Service durations and time windows reflect plausible handling times and availability windows. Travel times were derived from mapping services and rounded to the nearest minute; cross region travel between Montreal and Quebec clusters is set to 165 minutes. Travel between sites within the Quebec cluster takes 7–10 minutes, and between the pickup and dropoff of Job 1 (Montreal) is 17 minutes. The matrix is complete and symmetric where appropriate. Lemma 3.3 is a theoretical necessary condition; the empirical enumeration below does not rely on triangle inequality and remains valid under rounded travel times.

Table 2: Job data for the empirical instance. Times are in minutes past midnight. The depot v_0 is co-located with the pickup of Job 1.

Job	Site	Service s_i	Earliest e_i	Latest ℓ_i	Comments
1	p_1	45	540	630	Montreal pickup (depot)
	d_1	30	600	720	Montreal dropoff
2	p_2	40	660	780	Quebec pickup
	d_2	30	720	900	Quebec dropoff
3	p_3	30	510	570	Quebec pickup
	d_3	25	540	660	Quebec dropoff
Depot	v_0	0	0	1440	starting location (coincides with p_1)

Table 3: Directed travel time matrix t_{ij} (minutes) for the six non depot vertices. Cross region travel times (Montreal–Quebec) are 165 minutes. Within Quebec, p_2 and d_3 are connected by 10 minutes, p_2 and p_3 by 9 minutes, and d_2 and p_3 by 7 minutes. The matrix is symmetric for pairs within the same cluster. The diagonal entries are omitted.

	p_1	d_1	p_2	d_2	p_3	d_3
p_1	–	17	165	165	165	165
d_1	17	–	165	165	165	165
p_2	165	165	–	13	9	10
d_2	165	165	13	–	7	16
p_3	165	165	9	7	–	16
d_3	165	165	10	16	16	–

Key arcs driving infeasibility. The long travel times of 165 minutes between the Montreal pickup/dropoff (p_1, d_1) and the Quebec cluster (p_2, d_2, p_3, d_3) are the primary cause of infeasibility. Servicing Job 1 in Montreal first and then travelling to Quebec leaves little time to meet the early pickup window of Job 3. Conversely, going to Quebec first makes it difficult to return to Montreal in time for Job 1. Within the Quebec cluster, $d_2 \to p_3$ takes only 7 minutes and $p_2 \to p_3$ only 9 minutes, making those transitions relatively easy. Table 3 highlights these key arcs.

8.2 Enumeration procedure

To audit the instance, we enumerated all $|\Pi| = 90$ permutations that respect pickup-before-dropoff order (Proposition 3.2). For each permutation $\pi = (v_1, \ldots, v_6)$ we simulated a schedule starting at a fixed depot departure time $T_{v_0} = 0$.

The vehicle first travels from the depot to the first visited vertex, then follows the permutation, and finally returns to the depot. Concretely, we evaluate the route

$$(v_0, v_1, v_2, \ldots, v_6, v_0).$$

For each leg (a, b) in this sequence we add travel time t_{ab}; at each visited vertex $b \in V \setminus \{v_0\}$ we compute the service start time

$$T_b = \max\{e_b,\ T_a + s_a + t_{ab}\},$$

and then advance time by adding the service duration s_b.

For *every* permutation we record: (i) the *total travel time*

$$\sum_{k=0}^{6} t_{v_k, v_{k+1}} \quad \text{with } v_0 \text{ as depot and } v_7 = v_0,$$

and (ii) an *unconstrained total schedule duration* defined as the completion time upon returning to the depot when the above update rule is applied throughout the entire route.

In addition, feasibility with respect to time windows is evaluated separately: if for some visited vertex b the computed service start time satisfies $T_b > \ell_b$, the permutation is declared infeasible and the index of the first violated site is recorded as the failure index.

8.3 Travel time and schedule statistics

Table 4 summarises the distribution of travel times and total schedule durations over all 90 permutations. Travel times range from 379 to 1006 minutes, reflecting whether the route crosses between the Montreal and Quebec clusters only once in each direction or oscillates between clusters multiple times. Service durations and waiting extend total schedule durations to between 1024 and 1705 minutes. The mean travel time is 695.89 minutes, while the median is 700 minutes. The first and third quartiles (Q1 and Q3) show that most permutations cluster around these values, and the standard deviations indicate significant dispersion.

Several qualitative insights emerge from these statistics. First, the gap between the minimum and maximum travel times (over five hours) underscores the impact of ordering decisions: a poorly chosen route that oscillates between clusters incurs heavy cross-region travel, whereas a well structured route that completes all tasks in one region before moving to the next keeps the travel time low. Second, the fact that service and waiting inflate the total schedule duration by roughly six to eight hours shows that temporal feasibility is dominated by the windows rather than by pure travel time. Finally, the moderate difference between the mean and the median indicates a slightly skewed distribution; the right tail comprises routes with multiple unnecessary inter-region traversals, which propagation can quickly eliminate.

Table 4: Travel time and total schedule statistics over all $|\Pi| = 90$ precedence respecting permutations. Travel time is the sum of travel times along the closed tour $(v_0, v_1, \ldots, v_6, v_0)$, with v_0 co-located with p_1. Total schedule time adds travel, waiting, and service durations, starting from $T_{v_0} = 0$. Q1 and Q3 denote the 25% and 75% quantiles.

	Min	Max	Mean	Median	Q1	Q3	Std Dev
Travel time (min)	379	1006	695.89	700	683	706	193.94
Total schedule (min)	1024	1705	1373.77	1397	1251	1426	176.59

8.4 Failure index distribution

For each of the 90 precedence-respecting permutations, we record the index $f \in \{1, \ldots, 6\}$ of the first vertex at which a time window is violated. As shown in Table 5, 48 permutations (53.3%) fail at the second visited vertex, 33 (36.7%) at the third, and 8 (8.9%) at the fourth. Only one permutation (1.1%) remains feasible until the fifth site.

These early violations are primarily driven by the 165-minute transit between the Montreal and Quebec clusters. In sequences where the vehicle services the Montreal job (p_1, d_1) before travelling to Quebec, the arrival time at p_3 frequently exceeds its latest start time ($\ell_{p_3} = 570$).

The concentration of failures at $f \leq 3$ indicates that infeasibility is typically detected after assigning only a short prefix of the route, suggesting that the conditional difference constraints provide significant pruning of the search space before a complete tour is constructed.

Table 5: Distribution of failure indices for the 90 permutations. The entry in column f counts the number of permutations that violate a time window at the fth visited vertex.

Failure index f	1	2	3	4	5	6
Count	0	48	33	8	1	0

8.5 Sensitivity experiment

To explore the impact of time window widths, we conducted a sensitivity experiment in which the latest time ℓ_i of every non-depot site was increased by Δ minutes (earliest times e_i and all travel times t_{ij} were unchanged). For $\Delta \in \{60, 120, 180\}$ no feasible precedence-respecting permutation was found. When $\Delta = 240$ minutes, four permutations became feasible; when $\Delta = 300$ minutes, nine permutations were feasible.

For each feasible permutation we report the travel time of the closed tour $(v_0, v_1, \ldots, v_6, v_0)$, which is independent of Δ, and the total schedule duration (travel + waiting + service), which varies with Δ through waiting. Among the feasible schedules, the minimum travel time is 382 minutes for $\Delta = 240$ and 379 minutes for $\Delta = 300$; the corresponding mean travel times are 544.25 and 492.00 minutes. These results illustrate a threshold effect: several hours of additional slack are required to restore feasibility, and the set of feasible routes expands as Δ increases.

Our sensitivity experiment exhibits a pronounced threshold effect. The gap between the largest infeasible slack ($\Delta = 180$) and the smallest feasible slack ($\Delta = 240$) reveals that roughly four hours of additional slack are required to compensate for

the long cross-region travel and service durations. Beyond that point feasibility remains fragile: only a handful of permutations (four or nine out of ninety) remain viable even when every window is widened by four or five hours. This highlights that schedule feasibility is not merely a function of total slack but also of the ordering of jobs. In practical applications, modest window relaxations may still leave the instance infeasible unless the route structure is aligned carefully with the available windows. Such sensitivity analyses therefore provide valuable guidance on how much slack must be introduced to restore feasibility and which orders become possible when slack is added.

9 Related work

The pickup and delivery problem with time windows (PDPTW) has been studied extensively in the operations research and artificial intelligence literature. Surveys of vehicle routing and pickup–delivery problems are given in [1, 11]. Early modelling work appears in [2], while benchmark instances for time-window routing were introduced by Solomon [3].

Exact methods for PDPTW are typically based on mixed integer linear programming (MILP) formulations combined with branch-and-price, column generation, and cutting-plane techniques [4]. Strengthening inequalities exploiting precedence and time-window structure improve linear relaxations. Closely related variants, such as dial-a-ride problems, are surveyed in [5].

Constraint programming (CP) approaches model routing and scheduling decisions using domain variables and propagation. Hybrid methods combine CP propagation with MILP bounding and metaheuristic search [10, 6]. Metaheuristics such as large neighborhood search and guided local search often integrate CP propagation to filter infeasible moves efficiently [7, 6]. General treatments of constraint-based scheduling are given in [8], and propagation-guided search strategies in CP solvers are described in [9].

Simple temporal networks (STNs), introduced by Dechter, Meiri, and Pearl [12], provide a graph-based framework for reasoning about temporal constraints. In our work, this framework is adapted to a setting in which temporal constraints are activated by routing decisions, yielding a temporal network that is constructed incrementally during search.

10 Discussion

The CP formulation presented in this paper separates routing structure and temporal feasibility through successor variables, position variables, and conditional difference constraints. This separation prevents infeasible schedules in which temporal con-

straints are satisfied only through waiting, and allows precedence to be enforced independently of time propagation.

Interpreting the active constraints as a temporal network provides insight into how propagation operates during search. As successor variables are instantiated, additional difference constraints become active and progressively tighten feasible time intervals. Infeasibility can often be detected early, either through local violations or through the accumulation of incompatible constraints.

The empirical analysis is limited to a small instance, but it illustrates how time windows and travel times interact to restrict feasible orderings, in particular the effect of the long cross-region transit between Montreal and Quebec. In this instance, more than half of the precedence-respecting permutations fail after assigning only two vertices, suggesting that temporal propagation can significantly reduce the search space when windows are tight.

Several directions for further work follow from this formulation. On the modelling side, the approach could be extended to include additional constraints such as vehicle capacities or multiple vehicles. On the algorithmic side, it would be useful to study how different propagation strategies for difference constraints affect performance, and whether specialised propagators for conditional temporal constraints can improve early detection of infeasibility. Finally, larger computational experiments would be needed to evaluate how the formulation behaves on standard benchmark instances and to compare it more directly with existing MILP and hybrid methods.

11 Conclusion

This paper presented a constraint programming formulation for the single-vehicle pickup and delivery problem with time windows based on successor variables, a global Circuit constraint, position variables, and conditional difference constraints. The model represents routing and timing within a unified framework while avoiding big-M linearisation.

The formulation highlights how routing decisions and temporal constraints interact. Conditional difference constraints become active as arcs are selected, which progressively restricts feasible time assignments. The use of position variables ensures that pickup–dropoff precedence is enforced in the route independently of timing, while the return-time variable avoids introducing an inconsistent time cycle at the depot.

The theoretical analysis provides necessary conditions for feasibility, including subset-based bounds that relate travel times, service durations, and time-window widths. The empirical study on a small instance illustrates how these constraints operate in practice: most precedence-respecting sequences are eliminated early due to time-window violations, and feasibility depends strongly on the structure of travel times and the available slack.

Overall, the paper shows how a CP formulation can represent this PDPTW variant and how the associated temporal constraints explain infeasibility on the test instance. Further work is needed to evaluate the approach on larger instances and to study how propagation and search strategies affect performance.

References

[1] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002. doi:10.1137/1.9780898718515.

[2] M. W. P. Savelsbergh and M. Sol. "The general pickup and delivery problem". *Transportation Science*, 29(1):17–29, 1995. doi:10.1287/trsc.29.1.17.

[3] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):254–265, 1987. doi:10.1287/opre.35.2.254.

[4] S. Ropke and J.-F. Cordeau. Branch-and-cut-and-price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009. doi:10.1287/trsc.1090.0272.

[5] J.-F. Cordeau and G. Laporte. "The dial-a-ride problem: Models and algorithms". *Annals of Operations Research*, 153(1):29–46, 2007. doi:10.1007/s10479-007-0170-8.

[6] R. Bent and P. Van Hentenryck. "A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows". *Computers & Operations Research*, 33(4):875–893, 2006. doi:10.1016/j.cor.2004.08.001.

[7] P. Kilby, P. Prosser and P. Shaw. "Guided local search for the vehicle routing problem with time windows". In S. Voss, S. Martello, I. H. Osman and C. Roucairol, editors, *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 473–486, Kluwer Academic Publishers, Boston, 1999.

[8] P. Baptiste, C. Le Pape and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. International Series in Operations Research & Management Science, Vol. 39, Springer, New York, 2001. doi:10.1007/978-1-4615-1479-4.

[9] L. Perron, P. Shaw and V. Furnon. "Propagation Guided Large Neighborhood Search". In M. Wallace (ed.), *Principles and Practice of Constraint Programming – CP 2004*, Lecture Notes in Computer Science, vol. 3258, pages 468–481. Springer, Berlin, Heidelberg, 2004. doi:10.1007/978-3-540-30201-8_35.

[10] S. Demassey, C. Artigues and P. Michelon. "Constraint-propagation-based cutting planes: An application to the resource-constrained project scheduling problem". *INFORMS Journal on Computing*, 17(1):52–65, 2005. doi:10.1287/ijoc.1030.0043.

[11] S. N. Parragh, K. F. Doerner and R. F. Hartl. "A survey on pickup and delivery problems: Part I — Transportation between customers and depot". *Journal für Betriebswirtschaft*, 58(1):21–51, 2008. doi:10.1007/s11301-008-0033-7.

[12] R. Dechter, I. Meiri and J. Pearl. "Temporal constraint networks". *Artificial Intelligence*, 49(1–3):61–95, 1991. doi:10.1016/0004-3702(91)90006-6.