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

Timothy Roch
Université de Montréal
Montréal, QC, Canada
timothy.roch@umontreal.ca

With supervision from

Primous Pomalegni
Université de Montréal
Montréal, QC, Canada

Henri Joël Akougbe
Université Laval
Québec, QC, Canada

Abstract

We study the single-vehicle pickup and delivery problem with time windows (PDPTW), in which each pickup must precede its corresponding dropoff and all service starts must lie within prescribed time windows. We present a constraint programming (CP) formulation based on successor variables, a global Circuit constraint, position variables for route order, and conditional difference constraints that activate travel-time separations only on selected arcs. Because the route is closed, we introduce a separate return-time variable rather than propagating timing back into the depot.

The central perspective of the paper is that routing decisions induce a temporal constraint graph. As successor variables are fixed, the active timing constraints form an evolving simple temporal network whose edges correspond to selected travel and service separations. This view clarifies how route choices reveal temporal structure, how propagation tightens feasible service-time bounds, and why some partial assignments become infeasible before a complete tour is specified. Within this framework, we derive necessary infeasibility conditions, including subset-based time-window bounds, and present a baseline mixed-integer linear programming formulation for comparison.

We complement the formulation with a controlled empirical study over synthetic PDPTW instances. The experiments enumerate all precedence-respecting routes for instances with three to five pickup–dropoff pairs and examine feasible-route percentages, failure depths, prefix-failure profiles, slack sensitivity, and travel-time structure. The results support the modeling perspective developed in the paper: active timing constraints expose temporal structure during search, and this structure helps explain both early infeasibility detection and the sensitivity of feasibility to time-window slack.

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 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.

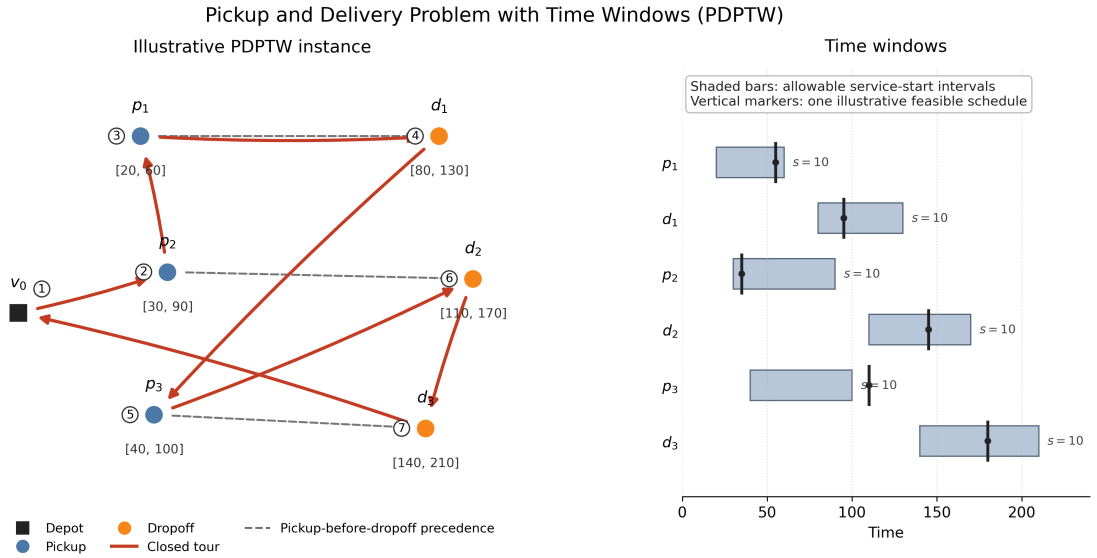

Figure 1: Illustrative PDPTW instance with pickup–dropoff pairs, precedence relations, and time windows.

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 present a CP formulation that avoids big-M timing couplings 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. Thus, a routing decision does more than choose the next vertex: it also adds a temporal edge of the form

$$T_j - T_i \geq s_i + t_{ij}$$

to the currently active timing graph.

This induced temporal graph is the main interpretive object of the paper. For any partial assignment of successor variables, the selected arcs define a partial system of difference constraints over the

service-time variables. As search progresses, this system evolves into a simple temporal network whose active edges encode the temporal consequences of the route decisions already made. This perspective helps clarify how propagation tightens time bounds, why certain partial routes become infeasible early, and how time-window slack affects the size of the feasible route set.

The paper develops this perspective in three steps. First, we define the CP formulation and the corresponding conditional timing constraints. Second, we interpret the active timing constraints as an evolving simple temporal network and derive necessary infeasibility conditions, including subset-based time-window bounds. Third, we study a controlled family of synthetic PDPTW instances in which all precedence-respecting routes are enumerated for three to five pickup–dropoff pairs. The empirical analysis examines feasible-route percentages, failure depths, prefix-failure profiles, slack sensitivity, and travel-time distributions in order to show how routing structure and temporal constraints interact.

2 Scope and contributions

This paper is primarily a modeling and interpretive study of the single-vehicle PDPTW. Its focus is not on proposing a new state-of-the-art solver, but on making explicit the temporal structure induced by a CP formulation with conditional timing constraints. The main contributions are as follows.

1. A closed-tour CP formulation that combines successor variables, a global Circuit constraint, position variables, and conditional difference constraints, while avoiding big-M timing couplings and avoiding temporal propagation back into the depot.

2. An induced temporal-network interpretation of the formulation. Under a partial or complete assignment of successor variables, the active timing constraints define a simple temporal network over service-start variables. This view clarifies how the temporal graph evolves during search and how propagation tightens service-time bounds or detects infeasible partial assignments.

3. Necessary feasibility conditions, including subset-based time-window bounds that relate travel times, service durations, and window widths. These conditions connect the temporal-network perspective to explicit infeasibility certificates.

4. A controlled empirical analysis over synthetic PDPTW instances, using exact enumeration of precedence-respecting routes for $n \in \{3, 4, 5\}$. The experiments measure feasible-route percentages, normalized failure depths, prefix-failure profiles, slack sensitivity, and travel-time structure.

The paper also includes a baseline MILP formulation for comparison. This formulation is used to contrast big-M timing couplings with the conditional temporal structure of the CP model, rather than to claim a computational benchmark against MILP solvers.

3 Notation and preliminaries

We introduce the notation used throughout the paper and briefly review difference constraints, simple temporal networks, and fixpoint propagation. Table 1 summarises the main symbols that will be used.

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

Symbol	Description
Sets and vertices	
J	Set of jobs, $J = \{1, \dots, n\}$.
p_k	Pickup vertex for job k.
d_k	Dropoff vertex for job k.
v_0	Depot vertex.
V	Vertex set, $V = \{v_0\} \cup \{p_k, d_k : k \in J\}$.
A	Directed arc set, $A = \{(i, j) \in V \times V : i \neq j\}$.
Travel and time-window data	
t_{ij}	Travel time from vertex i to vertex j.
s_i	Service duration at vertex i.
$[e_i, \ell_i]$	Time window at vertex i.
H	Global planning horizon.
Constraint programming model	
succ_i	Successor of vertex i in the route.
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.
MILP model and enumeration	
x_{ij}	Binary indicator equal to 1 if the vehicle travels from i to j.
M_{ij}	Big-M constant used in the MILP linearisation.
Π	Set of precedence-respecting permutations.

3.1 Difference constraints and temporal graphs

A difference constraint is an inequality of the form

$$T_j - T_i \geq c_{ij}, \tag{1}$$

where T_i and T_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 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. Prop-

agation 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 $T_{v_1} \ge u$ and $T_{v_{2|K|}} \le v$, we obtain $v - u \ge \alpha_K$.

Combining $v - u \le \beta_K$ and $v - u \ge \alpha_K$ shows that feasibility implies $\alpha_K \le \beta_K$. 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 $J = \{1, \dots, n\}$, 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:

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

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

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

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

The return to the depot after servicing the final vertex is modeled separately through a return-time variable. This variable is introduced in the CP formulation in Section 5 and used again in the baseline MILP formulation in Section 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 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 $P_i \in \{1, \dots, |V| - 1\}$ 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 impose the global constraint

$$\text{AllDifferent}\big(\{P_i : i \in V\}\big),$$

which requires all position variables to take pairwise distinct values. Thus, each vertex occupies a unique position in the tour. 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

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

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 $O_i \in \{0, \dots, |V| - 1\}$ and post the implications

$$\mathsf{succ}_i = v_0 \Rightarrow O_i = |V| - 1, \qquad \forall i \in V \setminus \{v_0\}, \tag{18}$$

$$\mathsf{succ}_i = j \Rightarrow O_j = O_i + 1, \qquad \forall (i, j) \in A \text{ with } j \neq v_0. \tag{19}$$

together with $O_{p_k} < O_{d_k}$ for all $k \in J$. The circuit constraint together with (19) 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:

$$\min \sum_{i \in V} t_{i,\mathsf{succ}_i}. \tag{20}$$

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

6 Temporal-network interpretation and theoretical analysis

In this section we make explicit the temporal-network structure induced by the CP formulation. We first describe the temporal graph associated with a partial assignment of routing decisions. We then explain how propagation on this graph detects infeasibility, before recalling complexity results for the full PDPTW and for the precedence-only variant.

6.1 Induced temporal network under a partial route

The CP formulation in Section 5 contains two coupled components. The successor and position variables describe the route structure, while the time variables describe service starts. The link between these components is given by the conditional difference constraints

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

Thus, assigning a successor does not only select an arc in the route. It also activates a temporal constraint between two service-start variables.

Let α be a partial assignment of successor variables. We define the active non-return arc set induced by α as

$$A_\alpha = \{(i, j) \in A : \alpha(\mathsf{succ}_i) = j,\ j \neq v_0\}.$$

Each active arc $(i, j) \in A_\alpha$ induces the difference constraint

$$T_j - T_i \geq s_i + t_{ij}.$$

The corresponding induced temporal graph is

$$G_\alpha = (V, A_\alpha),$$

where each edge $(i, j) \in A_\alpha$ has weight $s_i + t_{ij}$. The time windows remain bounds on the service-start variables:

$$e_i \le T_i \le \ell_i, \qquad \forall i \in V.$$

Return arcs are handled separately. If $\alpha(\mathsf{succ}_i) = v_0$, the model does not add a constraint from T_i back into T_{v_0}. Instead, it activates the return-time constraint

$$T_{\text{return}} \ge T_i + s_i + t_{iv_0}.$$

This distinction is essential in the closed-tour setting. If timing constraints were propagated around the entire tour and back into T_{v_0}, then any tour of positive total duration would create an inconsistent positive cycle under the $\ge$ convention. The return-time variable allows the routing structure to remain a closed tour while keeping the temporal graph over service-start variables well-defined.

As search progresses, more successor variables become assigned and the active set A_α grows. The induced graph G_α can therefore be viewed as an evolving simple temporal network. We use this terminology only as an interpretive lens for conditional difference constraints revealed by routing decisions. We do not refer to dynamic controllability, contingent constraints, or other extensions of temporal-network theory.

A complete route induces a fixed temporal network containing one active timing edge for each selected non-return arc. A partial route induces a partial temporal network, which may already impose strong lower bounds on some service-start variables. This is the sense in which temporal feasibility can be studied before the full Hamiltonian tour has been instantiated: route decisions progressively reveal a temporal graph, and propagation checks whether that graph remains consistent with the time-window bounds.

6.2 Propagation and early infeasibility detection

In the notation of Section 6.1, propagation acts on the currently active temporal graph G_α together with the time-window bounds. Each active arc $(i, j) \in A_\alpha$ imposes

$$T_j \ge T_i + s_i + t_{ij}.$$

More generally, every directed path in G_α gives a lower bound on the service start at its endpoint. If a path from i to j has total weight w, then any feasible assignment must satisfy

$$T_j \ge T_i + w.$$

Thus, as the active graph grows, propagation tightens the lower bounds of service-start variables. If the resulting lower bound for some vertex j exceeds its latest time ℓ_j, the partial route is infeasible.

Equivalently, bound propagation repeatedly applies updates of the form

$$e_j \leftarrow \max\{e_j, e_i + s_i + t_{ij}\}$$

for active arcs (i, j). Backward propagation can also tighten upper bounds through

$$\ell_i \leftarrow \min\{\ell_i, \ell_j - s_i - t_{ij}\}.$$

If these updates produce an empty domain, meaning $e_i > \ell_i$ for some vertex i, then the current partial assignment of successor variables cannot be extended to a feasible schedule.

This is the temporal-network interpretation of early pruning in the CP model. Route decisions activate temporal edges, and these edges may already make the remaining time-window domains inconsistent before the full tour has been specified. The following proposition gives a simple local certificate of infeasibility. It captures the idea that if one vertex must precede another, then travel and service time between them may already exceed the latter vertex's available time window.

Proposition 6.1. 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

$$e_i + s_i + t_{ij} > \ell_j. \tag{21}$$

Then no feasible schedule exists.

Proof. If in every feasible route i precedes j, then any schedule must start service at j no earlier than the earliest possible start at i, plus the service duration at i, plus the travel time from i to j. Hence

$$T_j \geq e_i + s_i + t_{ij}.$$

If this quantity exceeds ℓ_j, then service at j cannot begin within its time window. Therefore no feasible schedule exists.

Proposition 6.1 is local, but it illustrates the general mechanism of temporal propagation. In the induced temporal network, infeasibility appears through paths whose accumulated weight forces a service start beyond its latest time. The ordering relation between i and j may come from explicit pickup–dropoff precedence, from position-domain propagation, or from routing decisions that have already been fixed.

The same idea extends beyond pairs of vertices. Lemma 3.3 gives a subset-level necessary condition: if the minimum travel and service time needed to traverse a subset of jobs exceeds the spread of the corresponding time windows, then no ordering of those jobs can be feasible. In temporal-network terms, such a violation means that any route-induced graph over that subset must accumulate more time than the available window bounds permit. Thus Proposition 6.1 and Lemma 3.3 can both be viewed as infeasibility certificates derived from the same underlying temporal structure.

6.3 Feasibility complexity

The temporal-network view explains how a fixed or partially fixed route can be checked and propagated. It does not remove the combinatorial difficulty of choosing the route itself. The decision version of the PDPTW remains NP–complete even with one vehicle.

Proposition 6.2. 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 of all pickup and dropoff vertices, equivalently a successor mapping. Given such an ordering, one can verify feasibility by propagating arrival times along the route 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 $U = \{1, \ldots, m\}$. We construct a PDPTW instance with $n = m$ jobs. For each $k \in \{1, \ldots, m\}$ create a pickup vertex p_k and a dropoff vertex d_k, and include a depot v_0. Set all service durations to zero:

$$s_i = 0, \qquad \forall i \in V.$$

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, for instance $H = 10m$. Set

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

Finally, impose the usual precedence constraints

$$p_k \prec d_k, \qquad \forall k.$$

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 every dropoff has earliest time $m + 1$, whereas every pickup has latest time m. Hence any feasible schedule must visit all pickups before visiting any dropoff. 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 on whether the vehicle can visit all pickups $p_1, \dots, p_m$ 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 contains 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, with 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 $(u_1, u_2, \dots, u_m)$, consider the route that starts at v_0, visits the pickups in the order

$$p_{u_1}, p_{u_2}, \dots, p_{u_m},$$

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 constructed 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.

This result separates two aspects of the problem. Once a route is fixed, the induced temporal network can be checked by propagating difference constraints and time-window bounds. The hard part is selecting a route whose induced temporal network remains consistent. The temporal-network interpretation therefore explains the structure of feasibility for a given partial or complete route, while the proposition shows that the global route-selection problem remains computationally difficult.

6.4 Precedence-only optimisation

To separate temporal feasibility from route-order complexity, we also consider the precedence-only variant. 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 the 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 are symmetric, nonnegative, and satisfy the 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 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

$$p_a \prec d_a, \qquad \forall a \in C.$$

Service durations are zero.

Metric travel times with co-located pairs. Intuitively, p_a and d_a represent the same physical location as 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 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.

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. Construct a new tour by removing d_a from its current position, replacing arcs $y \to d_a$ and $d_a \to z$ by $y \to z$, and 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 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. 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. Since $t_{p_{u_i},d_{u_i}} = 0$ 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.

The precedence-only variant isolates the combinatorial ordering component of the problem. Without time windows, a route determines feasibility with respect to precedence, and arrival times can be computed after the order has been chosen. Adding time windows introduces the temporal feasibility layer studied earlier in this section. A route no longer merely has a travel cost: it induces a temporal network whose active constraints must be consistent with the time-window bounds. Thus the full PDPTW combines the route-ordering difficulty of the precedence-only problem with the temporal feasibility structure described by the induced difference constraints.

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:

$$\min \quad \sum_{(i,j)\in A} t_{ij} x_{ij} \tag{22}$$

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

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

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

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

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

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

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

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

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

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

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

where M_{ij} is a big constant satisfying $M_{ij} \ge \ell_i + s_i + t_{ij} - e_j$ for $j \ne v_0$, and M_i^{return} satisfies $M_i^{\text{return}} \ge H - e_i - s_i - t_{i v_0}$. 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 study the behaviour of the formulation on a controlled family of synthetic single-vehicle PDPTW instances. The goal of the empirical analysis is not to benchmark solvers, but to examine the temporal structure induced by routing decisions. In particular, we focus on feasibility, early detection of infeasible route prefixes, sensitivity to time-window relaxation, and the effect of instance structure on the size of the feasible route set.

8.1 Experimental design

The main study consists of synthetic instances with $n \in \{3, 4, 5\}$ pickup and dropoff pairs. For each value of n, we consider three time-window profiles and three spatial-separation profiles. Each parameter combination is generated with 50 independent replicates, yielding

$$3 \times 3 \times 3 \times 50 = 1350$$

instances in total. The master random seed is fixed at 20260429. For a parameter combination (n, t, g, r), where t is the time-window profile, g is the separation profile, and r is the replicate index, the per-instance seed is computed deterministically from the first eight bytes of the SHA-256 hash of the string

$$\texttt{20260429} \,|\, n \,|\, t \,|\, g \,|\, r.$$

Thus each replicate is a distinct instance within its parameter cell, while the full study remains reproducible.

Each instance consists of a depot, n pickup vertices, n dropoff vertices, service durations, time windows, and a complete travel-time matrix. The depot is fixed at

$$v_0 = (0, 0).$$

Pickup vertices are sampled around a pickup cluster centred at $x = 10$. Dropoff vertices are sampled around a second cluster centred at $x = 10 + \gamma$, where the separation profile determines the horizontal gap

$$\gamma_{\text{low}} = 18, \qquad \gamma_{\text{medium}} = 36, \qquad \gamma_{\text{high}} = 60.$$

Equivalently, the dropoff-cluster centres are located at $x = 28$, $x = 46$, and $x = 70$. For each job k, the generator first samples a shared vertical baseline

$$Y_k \sim \text{Unif}[-12, 12].$$

It then samples the pickup and dropoff coordinates as

$$p_k = \big(\text{round}(N(10, 6^2), 3),\ \text{round}(Y_k + N(0, 3^2), 3)\big),$$

and

$$d_k = \big(\text{round}(N(10 + \gamma, 6^2), 3),\ \text{round}(Y_k + N(0, 3^2), 3)\big).$$

All three separation profiles therefore use the same within-cluster spread and differ only in the horizontal distance between pickup and dropoff clusters.

Travel times are derived from the Euclidean geometry of the generated instance. If vertices i and j have coordinates x_i and x_j, the travel time is

$$t_{ij} = \begin{cases} 0, & i = j, \\ \max\{1, \text{round}(\|x_i - x_j\|_2)\}, & i \neq j. \end{cases}$$

The resulting travel times are integer-valued and symmetric. They are interpreted as minutes. Since Euclidean distances are rounded to integers, the generator does not explicitly enforce an exact triangle inequality after discretization. Service durations are integer-valued: each pickup and dropoff receives a service time drawn uniformly from $\{5, 6, \ldots, 12\}$ minutes, while the depot service time is fixed at 0.

Time windows are generated from a randomly selected precedence-respecting reference route. The reference route is drawn uniformly from the exact route-template set for the corresponding value of n. Its anchor service-start times are computed by propagating travel and service times from the depot,

without waiting and without pre-existing time windows. If a_i denotes the anchor start time of vertex i, the window centre is

$$c_i = a_i + b_t + \delta_i + \xi_i,$$

where $t \in \{\text{tight}, \text{medium}, \text{loose}\}$ is the time-window profile, b_t is a profile-wide centre bias, δ_i is a vertex-type adjustment, and ξ_i is an integer jitter term. For pickups, $\delta_i = 0$. For dropoffs, $\delta_i = -3$ in the tight profile, $\delta_i = -2$ in the medium profile, and $\delta_i = 0$ in the loose profile. The generator then samples an early width α_i and a late width β_i, and sets

$$e_i = \max\{0, c_i - \alpha_i\}, \qquad \ell_i = \max\{e_i, c_i + \beta_i\}.$$

The numerical parameters of the three time-window profiles are shown in Table 2. The reference route determines the temporal anchor points, but it is not imposed as a feasible solution. Depending on the sampled windows, an instance may admit many feasible routes, very few feasible routes, or no feasible route at all.

Table 2: Time-window generation profiles used in the synthetic study. All ranges are integer ranges in minutes.

Profile	b_t	ξ_i	α_i	**Pickup** β_i	**Dropoff** β_i
tight	-5	$\{-10, \dots, 10\}$	$\{0, \dots, 13\}$	$\{7, \dots, 19\}$	$\{6, \dots, 18\}$
medium	-4	$\{-8, \dots, 8\}$	$\{5, \dots, 20\}$	$\{11, \dots, 24\}$	$\{10, \dots, 23\}$
loose	0	$\{-5, \dots, 5\}$	$\{28, \dots, 68\}$	$\{55, \dots, 95\}$	$\{59, \dots, 99\}$

For each n, we enumerate all precedence-respecting routes. By Proposition 3.2, the number of such routes is

$$|\Pi_n| = \frac{(2n)!}{2^n}.$$

Thus the enumeration contains 90 routes for $n = 3$, 2520 routes for $n = 4$, and 113400 routes for $n = 5$. The route-template generator uses the canonical numbering in which the pickup of job j is indexed by $2j - 1$ and the dropoff of job j is indexed by $2j$. It recursively appends either an unvisited pickup or the dropoff of a job whose pickup has already appeared, thereby generating exactly the precedence-respecting permutations. No instances with $n \geq 6$ are included in the main study.

8.2 Route simulation and metrics

For each generated instance and each precedence-respecting route

$$\pi = (v_1, \dots, v_{2n}),$$

we simulate the service sequence

$$(v_0, v_1, \dots, v_{2n})$$

starting from $T_{v_0} = 0$ at the depot. At each visited vertex i, the simulator computes the arrival time from the previous completion time and the travel time from the previous vertex. If the vehicle arrives before the earliest time e_i, it waits until e_i, so that the service start time is

$$S_i = \max\{A_i, e_i\}.$$

If $S_i > \ell_i$, the route is declared infeasible and the first violated position is recorded. Otherwise, the clock is advanced by adding the service duration at vertex i.

The reported travel-time and schedule-duration statistics follow the convention of the experimental pipeline: they include the leg from the depot to the first visited vertex and all inter-vertex legs in

the service sequence, but not the final return leg to the depot. This convention does not affect the feasibility metrics studied here, since no time window is imposed on the return to the depot in the experiment. It should be distinguished from the closed-tour formulation of the model, where a separate return-time variable is used to represent the final return without propagating time back into T_{v_0}.

The main route-level quantities are feasibility, total travel time, total waiting time, total schedule duration, and failure index. For an infeasible route, the failure index f is the first visited position at which a time window is violated. To compare instances with different values of n, we also use the normalized failure depth

$$\frac{f}{2n}.$$

Small values of this quantity indicate that infeasibility is detected after a short route prefix, while larger values indicate that the route remains temporally consistent until later in the sequence.

For each instance, we aggregate the route-level results into feasibility rates, failure-depth profiles, slack-sensitivity curves, and travel-time summaries. Slack sensitivity is evaluated on the fixed grid

$$\Delta \in \{0, 30, 60, \ldots, 360\}.$$

A uniform slack value Δ is applied by replacing each latest time ℓ_i by $\ell_i + \Delta$ for every non-depot vertex, while leaving earliest times unchanged. Equivalently, for each route π we compute the minimum uniform slack required for feasibility,

$$m(\pi) = \max_i (S_i - \ell_i)_+.$$

The slack-sensitivity curve at level Δ is obtained by counting routes with $m(\pi) \leq \Delta$.

8.3 Representative instance

Figure 2 shows one representative feasible instance from the generated study. The left panel displays the route geometry, including the depot, pickup vertices, dropoff vertices, and the order of visits. The right panel displays the corresponding time windows and service start times along the route. This figure is included to make the synthetic instance structure concrete before turning to aggregate statistics.

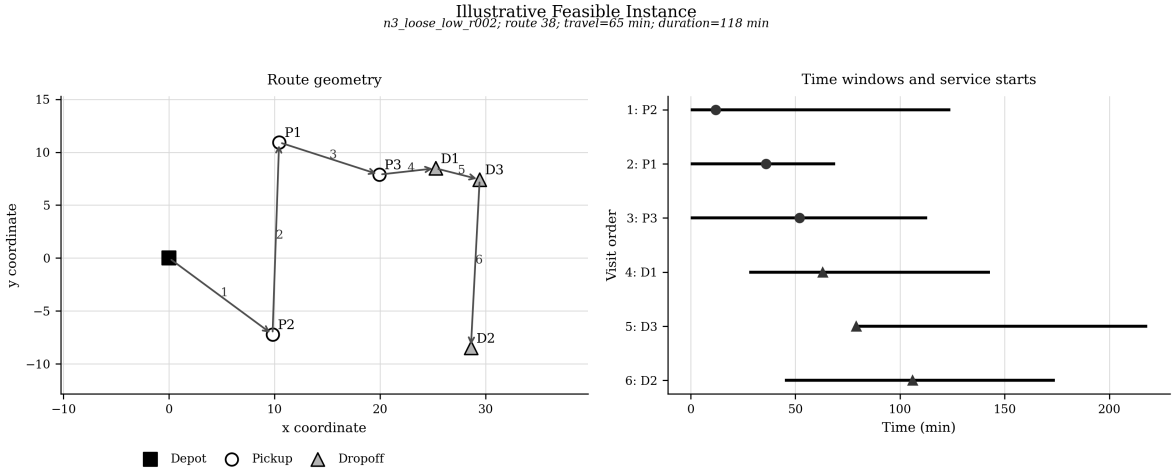

Figure 2: Representative feasible generated instance. The left panel shows the route geometry, while the right panel shows time windows and service start times along the selected route.

The example illustrates the two aspects of the model that are central to the empirical study. First, the route must respect the pickup-before-dropoff structure. Second, the timing constraints are revealed along the selected route and may become infeasible before all vertices have been assigned. This is the empirical analogue of the evolving temporal-network view discussed in the previous sections.

8.4 Feasible-route percentages

Figure 3 reports the mean percentage of feasible precedence-respecting routes across the generated instances. The figure is faceted by n, with rows corresponding to the time-window profile and columns corresponding to the cluster-separation profile.

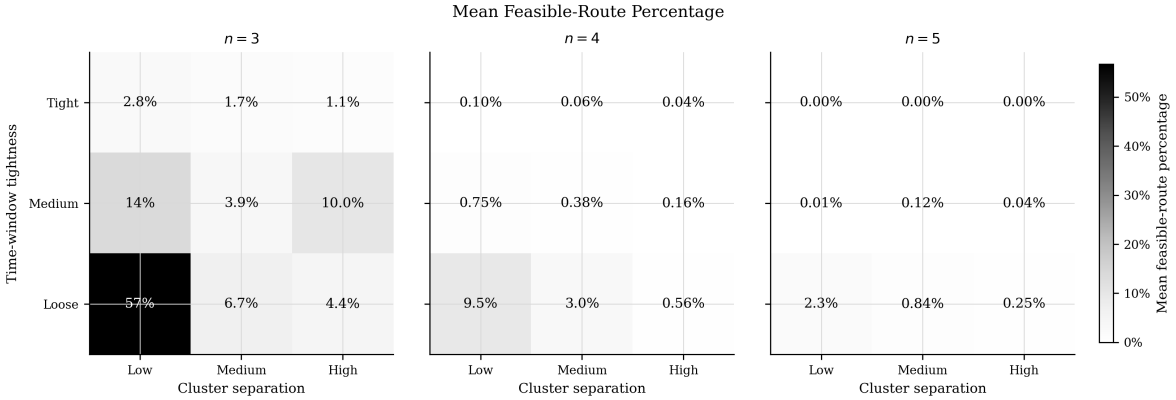

Figure 3: Mean percentage of feasible precedence-respecting routes. Each cell averages over generated instances with the corresponding number of jobs, time-window profile, and cluster-separation profile.

The feasible-route percentage decreases sharply as the number of jobs increases. For $n = 3$, some loose-window and low-separation settings admit a substantial fraction of feasible routes. For $n = 4$, feasible routes are already sparse. For $n = 5$, the feasible fraction is often close to zero, even when at least one feasible route exists. This distinction is important: the existence of a feasible route does not imply that feasibility is common in the search space. Instead, feasible schedules may occupy a very small portion of the precedence-respecting permutations.

The effect of separation is also visible. Increasing the cluster gap from 18 to 36 to 60 increases the travel burden of moving between pickup and dropoff regions, which makes it harder to satisfy time windows after inter-cluster moves. The time-window profile interacts with this effect: wider windows increase feasibility, but they do not eliminate the combinatorial difficulty created by longer travel times and larger route spaces.

8.5 Failure depth

Figure 4 reports the average normalized failure depth among infeasible routes. This metric summarizes how far an infeasible route typically progresses before a time-window violation is encountered.

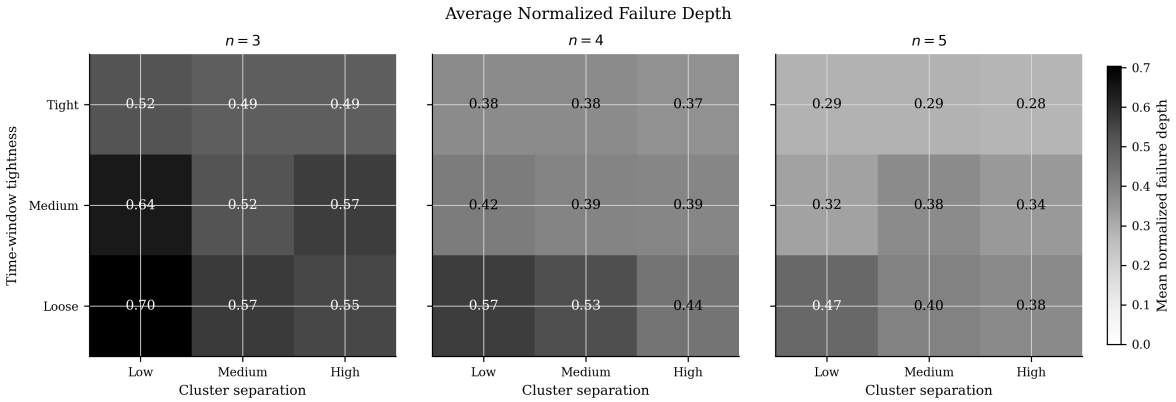

Figure 4: Average normalized failure depth among infeasible routes. Lower values indicate that infeasibility is detected earlier in the route prefix.

The results show that infeasibility is often detected before a complete route is constructed. This is especially pronounced for larger instances, where the normalized failure depth tends to be smaller. In these cases, a route prefix can already contain enough temporal information to violate a time window, even though many vertices remain unvisited.

This behaviour supports the interpretation of the active timing constraints as an evolving temporal network. As route decisions are fixed, new difference constraints become active. These constraints tighten earliest feasible service times, and in many cases they reveal infeasibility before the full Hamiltonian tour has been specified.

8.6 Failure profiles by prefix length

The average failure depth compresses each setting into a single value. Figure 5 gives a more detailed view by plotting the cumulative fraction of routes that have failed by each normalized prefix length.

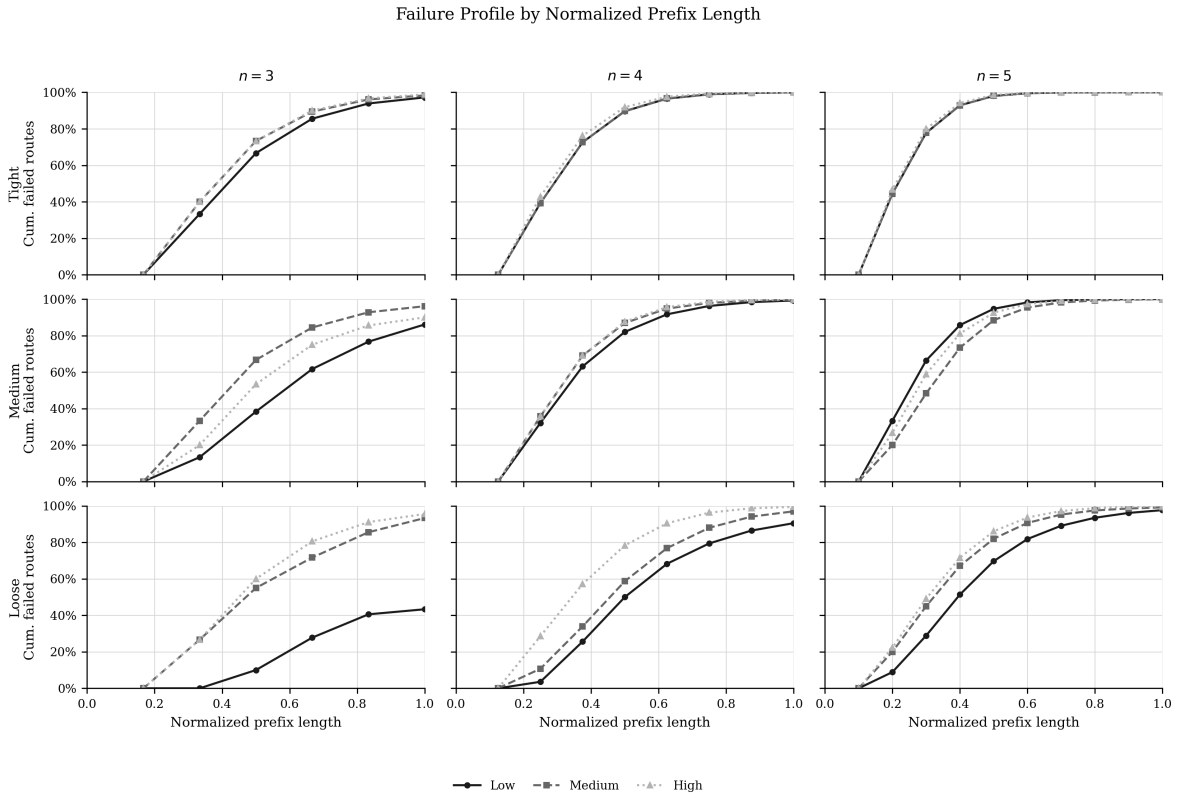

Figure 5: Cumulative failure profile by normalized prefix length. Each curve shows the fraction of routes that have violated a time window by a given fraction of the route prefix.

The curves show that many infeasible routes fail after only a moderate fraction of the route has been assigned. In tighter settings, the cumulative failure rate rises quickly. In looser settings, routes survive longer, but many still fail before completion when travel times and service durations accumulate unfavourably.

These profiles make the pruning mechanism more explicit. The route prefix determines a partial temporal graph. When the accumulated travel and service constraints push a service start beyond its latest time, the branch can be discarded. Thus, the failure profile provides an empirical measure of how quickly the temporal component of the model can rule out infeasible route prefixes.

8.7 Slack sensitivity

To measure the effect of time-window relaxation, we add a uniform slack value Δ to the latest time of every non-depot vertex and evaluate the resulting feasible-route percentage on the grid $\Delta \in \{0, 30, \ldots, 360\}$. Figure 6 reports the resulting slack-sensitivity curves.

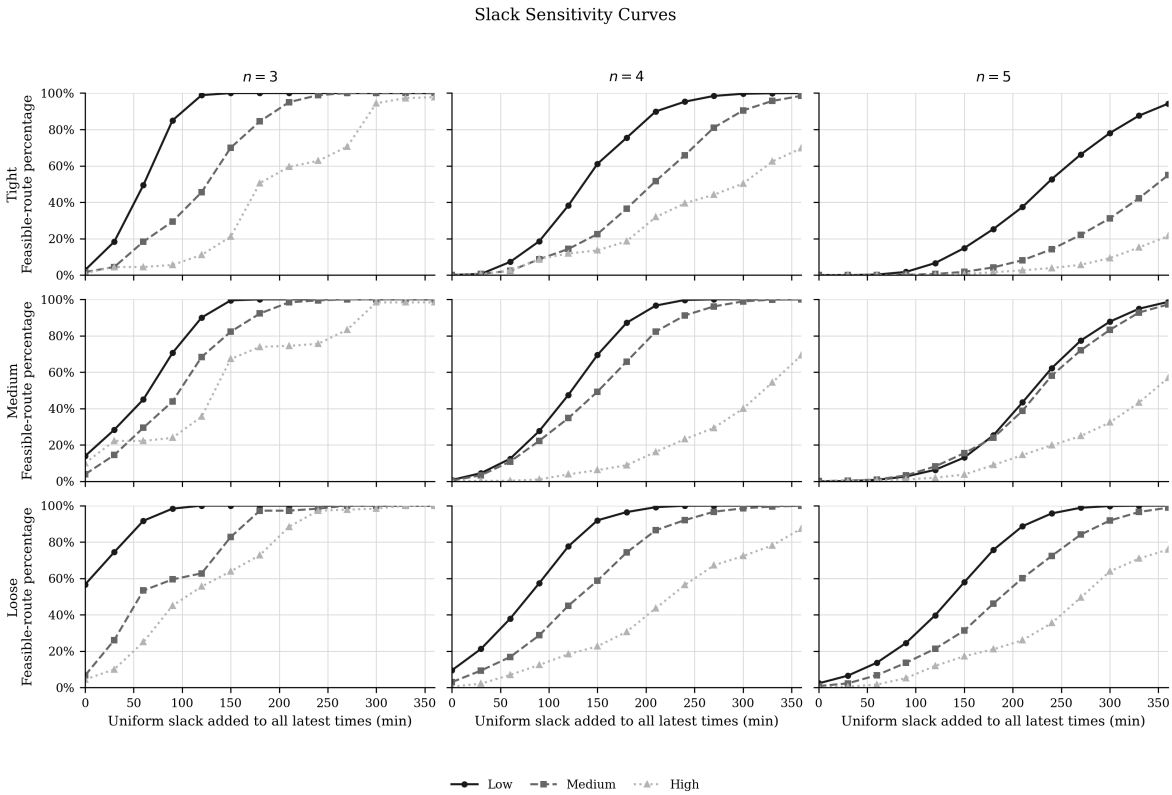

Figure 6: Slack-sensitivity curves. The horizontal axis gives the uniform slack added to all latest times, and the vertical axis gives the percentage of precedence-respecting routes that become feasible.

The curves exhibit a threshold-like behaviour. For small slack values, many settings have few feasible routes. As Δ increases, feasibility rises, but the amount of slack required depends strongly on n, the time-window profile, and cluster separation. Low-separation instances become feasible more quickly, while high-separation instances often require much larger slack. Larger instances also require more slack before a substantial portion of the route space becomes feasible.

This experiment complements the subset feasibility bound in Lemma 3.3. When travel and service durations consume more time than the spread of the relevant time windows, feasibility is impossible for certain route structures. Adding slack increases the window spread and can restore feasibility, but the slack required reflects the accumulated temporal burden of the route.

8.8 Travel-time distributions

Figure 7 reports travel-time distributions across the generated instances. The first panel summarizes mean travel time over all precedence-respecting routes, the second panel summarizes travel time among feasible routes, and the third panel summarizes the best feasible travel time in each instance.

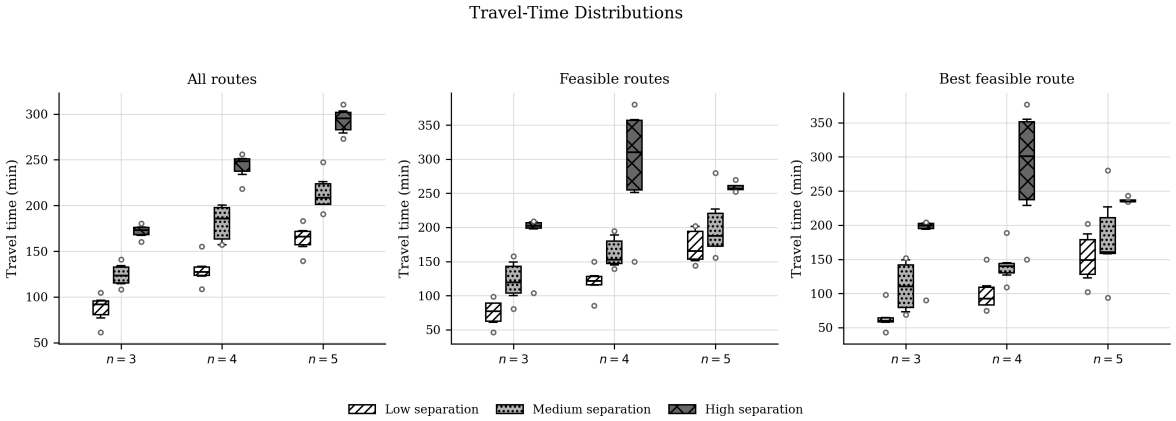

Figure 7: Travel-time distributions across generated instances. The panels compare all routes, feasible routes, and the best feasible route under different cluster-separation profiles.

The distributions confirm that cluster separation changes the geometry of the routing problem. High-separation instances have larger travel times, both across all routes and among feasible routes. This provides a structural explanation for the feasibility patterns observed above: longer inter-cluster travel consumes more of the available time-window slack, making temporal violations more likely.

The travel-time distributions are not intended as a solver-performance benchmark. Rather, they provide context for the temporal results. They show that the synthetic generator produces instances whose routing difficulty varies systematically with spatial structure, which in turn affects feasibility, failure depth, and slack sensitivity.

8.9 Summary

The empirical results support three main observations. First, feasible routes become sparse as the number of jobs increases, even when at least one feasible route exists. Second, infeasible routes are often detected after a relatively short prefix, which is consistent with the temporal-network view of conditional difference constraints becoming active during search. Third, uniform slack relaxation produces structured feasibility transitions whose shape depends on the time-window profile, cluster separation, and route size.

Together, these observations illustrate how routing decisions and temporal constraints interact in the single-vehicle PDPTW. The experiments do not claim broad computational superiority over alternative formulations. Instead, they provide empirical support for the modeling perspective developed in this paper: active timing constraints reveal temporal structure during search, and this structure can explain both early infeasibility and the sensitivity of feasibility to time-window slack.

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 Conclusion

This paper presented a constraint programming formulation for the single-vehicle pickup and delivery problem with time windows. The formulation uses successor variables and a global Circuit constraint to represent the route, position variables to encode route order, and conditional difference constraints to link selected arcs with service-start times. Because the route is closed, the return to the depot is handled through a separate return-time variable rather than by propagating timing constraints back into the depot.

The main modeling perspective developed in the paper is that routing decisions induce a temporal constraint graph. When a successor variable is fixed, the corresponding travel and service separation becomes an active difference constraint between service-start variables. As a result, each partial route determines an induced temporal network, and propagation can be interpreted as bound tightening on this evolving graph. This view helps explain why certain partial assignments become infeasible before the complete tour has been specified.

The theoretical analysis formalised this interpretation and connected it to feasibility. Local and subset-based necessary conditions show how accumulated travel and service durations can exceed the available time-window spread, yielding early certificates of infeasibility. At the same time, the complexity results show that temporal propagation does not remove the combinatorial difficulty of the problem: even for one vehicle, PDPTW feasibility remains NP–complete, and the precedence-only optimisation variant is NP–hard.

The empirical study examined these ideas on a controlled family of synthetic instances. By enumerating all precedence-respecting routes for instances with three to five pickup–dropoff pairs, the experiments measured feasible-route percentages, normalized failure depths, prefix-failure profiles, slack sensitivity, and travel-time structure. The results illustrate that feasible routes can become sparse as the number of jobs grows, that many infeasible routes are eliminated after relatively short prefixes, and that uniform time-window relaxation produces structured feasibility transitions.

Overall, the paper positions the CP formulation not only as a way to model the single-vehicle PDPTW, but also as a way to expose the temporal structure created by routing decisions. The induced temporal-network view provides a useful language for understanding propagation, early infeasibility detection, and the role of time-window slack. Future work could extend the formulation to multiple vehicles, vehicle capacities, and stochastic travel times, and could study specialised propagators or hybrid CP–MILP methods on larger benchmark instances.

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, editor, *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.