Efficient Duration-Based Workload Balancing for Interdependent Vehicle Routes

Carlo S. Sartori[1]
Department of Computer Science, KU Leuven, Belgium

Pieter Smet
Department of Computer Science, KU Leuven, Belgium

Greet Vanden Berghe
Department of Computer Science, KU Leuven, Belgium

Abstract

Vehicle routing and scheduling problems with interdependent routes arise when some services must be performed by at least two vehicles and temporal synchronization is thus required between the starting times of these services. These problems are often coupled with time window constraints in order to model various real-world applications such as pickup and delivery with transfers, cross-docking and home care scheduling. Interdependent routes in these applications can lead to large idle times for some drivers, unnecessarily lengthening their working hours. To remedy this unfairness, it is necessary to balance the duration of the drivers' routes. However, quickly evaluating duration-based equity functions for interdependent vehicle routes with time windows poses a significant computational challenge, particularly when the departure time of routes is flexible. This paper introduces models and algorithms to compute two well-known equity functions in flexible departure time settings: min-max and range minimization. We explore the challenges and algorithmic complexities of evaluating these functions both from a theoretical and an experimental viewpoint. The results of this paper enable the development of new heuristic methods to balance the workload of interdependent vehicle routes with time windows.

2012 ACM Subject Classification Applied computing → Transportation; Computing methodologies → Temporal reasoning; Mathematics of computing → Graph algorithms

Keywords and phrases Vehicle scheduling, Workload balancing, Route duration, Interdependent routes, Time windows

Digital Object Identifier 10.4230/OASIcs.ATMOS.2021.1

Funding Financial support provided by *Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen Programme* and project *Data-driven logistics* (FWO-S007318N).

Acknowledgements Editorial consultation provided by Luke Connolly (KU Leuven).

1 Introduction

Concerns regarding workload balancing in Vehicle Routing Problems (VRPs) have recently received attention in the literature [14, 15]. Most of this research addresses the VRP with route balancing [10], where the routes of multiple drivers are balanced according to some *equity function* in order to fairly distribute the workload between all workers. Route balancing is a challenging problem since it typically takes place in the context of a bi-objective VRP, where conflicting objectives such as total cost and workload imbalance must both be minimized. The difficulty of the problem increases when time windows are incorporated [16], in which case the workload is typically measured in terms of the route duration: from the departure time of the route until its completion, which includes possible idle periods of the driver.

[1] Corresponding author.

Existing literature has mainly focused on balancing workload between *independent* vehicle routes. By contrast, problems in which *interdependent* routes are considered have rarely been explored in terms of workload balancing. Interdependent routes occur when the start time of one driver's service depends on the completion time of another driver's service. In other words, there are temporal precedence constraints between tasks in different vehicle routes which therefore requires those routes to be synchronized somehow. Many real-world applications contain such interdependencies: pickup and delivery with transfers [17], equipment delivery and installation [1, 9] and home health care [8]. In all these applications, complications arise from the combination of time windows and interdependent routes. These difficulties are further compounded by the fact that we consider departure times of routes to be flexible. The combination of these three characteristics means that the computation of duration-based equity functions for workload balancing represents a nontrivial question and one which has not been previously addressed in the literature.

In this paper, we will consider two duration-based equity functions: min-max and range. These functions are often applied within decision support tools because they are very intuitive for decision makers [14]:

(1) **Min-max**: minimization of the longest route duration;

(2) **Range**: minimization of the difference between the longest and shortest route durations.

Evaluating these equity functions requires computing the minimum duration for all routes in a VRP solution. For VRPs with independent routes, such as the VRP with time windows, evaluating these durations can be performed in constant time after a preprocessing step [18]. In contrast, when routes are interdependent then these techniques for independent routes fail to correctly optimize functions (1) or (2). Indeed, [7] has noted that they are unaware of any constant-time method to update these duration-based equity functions that accommodate interdependent vehicle routes. When departure times are fixed, we can compute (1) and (2) with a linear time algorithm as detailed in Section 2. However, we have been unable to find studies concerning specialized algorithms with any complexity to correctly evaluate these functions when departure times are flexible.

The contributions of this paper are twofold. First, we describe how computing duration and corresponding equity functions of interdependent routes is challenging. Second, we introduce algorithms based on established methods in the literature to compute the duration-based workload balance of these routes along with their algorithmic complexity. A series of computational experiments provides additional understanding concerning the algorithmic performance in practice. The introduced algorithms can be incorporated within heuristic methods in which new solutions must be quickly evaluated with respect to workload balance. Hence, our contributions also open new research avenues for other researchers who would like to heuristically address vehicle routing problems which feature interdependent routes, time windows and workload balancing.

2 The interdependent route scheduling problem

This section defines the *Interdependent Route Scheduling Problem* (IRSP). The IRSP is defined over a graph $G = (V, A)$, where V is the set of nodes and A is the set of arcs that define temporal precedence constraints between pairs of nodes. Additionally, a set of fixed vehicle routes R is defined in G. A route $r_k \in R$ is a sequence of nodes $r_k = (\lambda_1, \dots, \lambda_{|r_k|})$ where $\lambda_i \in V$. All nodes in V belong to exactly one route. For a route $r_k \in R$, its first and last nodes are the origin and destination locations and denoted o_k and d_k, respectively.

Every node $i \in V$ has an associated time window $[e_i, l_i]$ which indicates the earliest time e_i and latest time l_i that service is allowed to begin at node i. A vehicle is allowed to arrive at i before e_i and wait for service to start, but it may never arrive later than l_i. The service duration at i is w_i units of time. Furthermore, there is a time horizon H so that all services, including departure and completion time of the routes, must lie within $[0, H]$.

Arcs are subdivided into two sets $A = A_R \cup A_P$. Arc set A_R contains route arcs (i, j, t_{ij}) which connect nodes i and j belonging to the same route. They represent trips of duration t_{ij}. Meanwhile, set A_P contains *interdependency arcs* (u, v, δ_{uv}) which connect nodes u and v belonging to two different vehicle routes. The start time of service at u and v is captured by means of Equation 1, where variable h_i denotes the start time of service at node $i \in V$ and where δ_{uv} is a parameter.

$$h_u + \delta_{uv} \leq h_v \tag{1}$$

Correctly defining δ_{uv} enables us to model the five most common interdependency constraints encountered in practice [6]. For example, setting $\delta_{uv} = w_u$ (the service duration at u) creates the *minimum difference* interdependency found in VRPs with transfers [13, 17]. Meanwhile, creating two arcs (u, v, δ_{uv}) and (v, u, δ_{vu}) models *general synchronization* constraints occurring in some delivery and installation problems [9]. When $\delta_{uv} = 0$ and $\delta_{vu} = -2\text{h}$ service at u and v may start simultaneously or with a difference of at most 2h. Similarly, if $\delta_{uv} = \delta_{vu} = 0$ then *strict synchronization* of the services at u and v is required, which is encountered in home health care problems [8]. In this paper, we present examples using the minimum difference interdependency, but the algorithms and models are valid for any constraint so long as it can be represented with the interdependency arcs in A_P. Interested readers are referred to Appendix A for more information on parameter δ_{uv}.

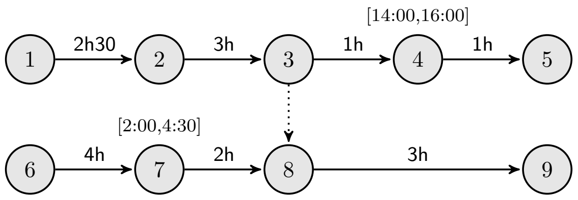
The goal of the IRSP is to produce a schedule where the starting time of service h_i complies with all of the time window constraints for every node $i \in V$. This includes deciding the departure and completion times of the routes at their origin and destination locations. Furthermore, we introduce three variants of the IRSP in this paper, which induce additional constraints to the decision of the starting times of service. The variants are:

(1) **Feasibility**: All routes $r_k \in R$ must comply with a maximum duration M.
(2) **Min-max**: produce a schedule that minimizes the longest duration $x_{\max}$ across all routes;
(3) **Range minimization**: produce a schedule that minimizes the difference between the longest duration $x_{\max}$ and the shortest duration $x_{\min}$.

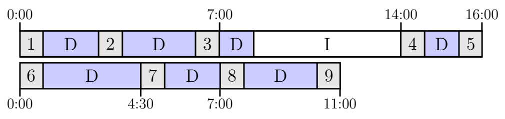
When departure times are fixed, these three variants reduce to computing the completion time of all routes and this can be trivially solved in $O(|V|)$ by assigning a start time of service to each node in a topological ordering of G (see Appendix B). However, we consider departure times to be additional decision variables in the IRSP, thereby increasing the search space and the complexity of solving the problem. Despite substantially complicating the evaluation of route duration, flexible departure times are encountered in many real-world applications [18] and are of significant importance for ensuring the best use of all resources.

Figure 1(a) illustrates an instance of the IRSP with two routes: $r_1 = (1, 2, 3, 4, 5)$ and $r_2 = (6, 7, 8, 9)$. The service duration is $w_i = 0\text{h}30$, $\forall i \in V$ and the departure and completion times of the routes must lie within $[0{:}00, 23{:}59]$ (time horizon $H = 24\text{h}$). Only nodes 4 and 7 have associated time windows. There is one minimum difference interdependency $(3, 8, 0\text{h}30) \in A_P$ which indicates that service at node 8 can only begin 0h30 after the start of service at node 3. Figures 1(b)–(e) depict four different solutions for the instance outlined in Figure 1(a). In these solutions, grey rectangles are service periods, blue rectangles (D) are driving periods and white rectangles (I) are idle (or waiting) periods.

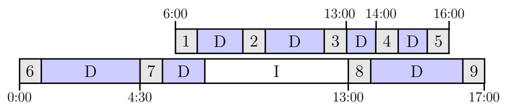
Figure 1(b) presents a solution to the IRSP in which all drivers depart at time $t = 0{:}00$. Due to the time window at node 4, route r_1 has 6h of idle time and a total duration of 16h. Note that removing the idle time in r_1 requires delaying the start of service at node 3 which consequently delays the start time of service at node 8, thereby lengthening the duration of route r_2. Indeed, if all of the idle time in r_1 is removed, then the duration of route r_2 is increased to 17h, as illustrated by Figure 1(c). This effectively increases both the Min-max and the Range equity functions compared to 1(b). Furthermore, to comply with a maximum duration of $M = 15$h, route r_1 must be postponed by an hour, which delays start of service at node 8 by an hour as well. This lengthens the duration of r_2 to 12h, as shown in Figure 1(d). The optimal schedule for both Min-max and Range is depicted in Figure 1(e), where a balance is achieved between the durations of routes r_1 and r_2. In this schedule, any further reduction concerning the duration of route r_1 would increase the duration of r_2, leading to suboptimal solutions. The optimal schedule is obtained by postponing the departure time of route r_1 by 2:30, which is not an intuitive solution.



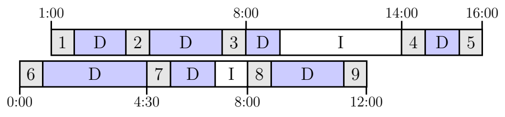
(a) Instance with two routes. Solid arcs represent direct trips where the weight is the trip's duration. Meanwhile, the dotted arc represents an interdependency constraint between the two routes.



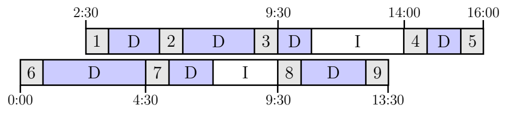
(b) Schedule obtained after computing earliest service times. Longest duration $x_{\max} = 16$h. Range $x_{\max} - x_{\min} = 5$h. This schedule is optimal with respect to both the min-max and range equity functions if departure times are fixed at $t = 0{:}00$.



(c) Schedule obtained by removing all idle time from route r_1. This reduces the duration of r_1 to 10h, but also increases the duration of r_2 to 17h. The longest duration is $x_{\max} = 17$h and the range is $x_{\max} - x_{\min} = 7$h.



(d) Schedule in which all routes comply with maximum duration $M = 15$h. This is only possible if departure times are flexible since route r_1 must start at $t = 1{:}00$. Note that this delay increases the duration of r_2 from 11h to 12h.



(e) Schedule with optimal $x_{\max} = 13$h30 and minimum difference $x_{\max} - x_{\min} = 0$ when departure times are flexible. This solution is obtained by delaying the departure time of route r_1 and the completion time of r_2 by 2h30.

Figure 1 An IRSP instance and four possible solutions.

Note that when considering the VRP with time windows, minimizing route duration is equivalent to minimizing total waiting time [18], however this is not the case for the IRSP. Indeed, the total waiting time in the four solutions outlined in Figure 1 is the same: 5h30. The key difference is in how this total waiting time is distributed across all the routes. Therefore, simply minimizing total waiting time could lead to any of the four solutions in Figures 1(b)–(e), which is not the desirable outcome.

Finally, as the number of interdependent routes increases, the complex interactions between routes become more difficult to manage. This motivates us to examine whether it is possible to design efficient algorithms to effectively schedule interdependent vehicle routes.

3 The feasibility problem

The feasibility problem is the decision-version of IRSP for which an algorithm must provide an answer to the following question: can all routes comply with a given maximum duration M? This section introduces a Mathematical Programming (MP) formulation to precisely describe the feasibility problem along with two special-purpose algorithms to solve it.

3.1 Mathematical formulation

A Linear Program (LP) for the feasibility IRSP is:

$$h_i - h_j \leq -w_i - t_{ij}, \quad \forall\ (i, j, t_{ij}) \in A_R \tag{2}$$

$$h_u - h_v \leq -\delta_{uv}, \quad \forall\ (u, v, \delta_{uv}) \in A_P \tag{3}$$

$$h_i \geq e_i, \quad \forall\ i \in V \tag{4}$$

$$h_i \leq l_i, \quad \forall\ i \in V \tag{5}$$

$$(h_{d_k} - h_{o_k}) \leq M, \quad \forall r_k \in R \tag{6}$$

Several general-purpose methods can be employed to solve this LP. For example, the Simplex algorithm, Karmarkar's algorithm [11] or more recent approaches whose worst-case time complexity make them more efficient in theory [3]. However, special-purpose algorithms exist which are capable of solving the LP much quicker.

3.2 Simple temporal networks

A *Simple Temporal Network* (STN) is a graph which comprises of nodes that are events and arcs between these nodes enable us to capture temporal relations between them. STNs have been used in the past to check feasibility of VRP solutions with interdependent routes such as the the dial-a-ride problem with transfers [13]. The formulation defined by Constraints (2)–(6) can be represented as an STN. In order to do so, we define a special node α as the beginning of time $t = 0$ and we replace Constraints (4) and (5) with:

$$h_\alpha - h_i \leq -e_i, \quad \forall\ i \in V \tag{7}$$

$$h_i - h_\alpha \leq l_i, \quad \forall\ i \in V \tag{8}$$

Constraints (2),(3),(6),(7) and (8) define a *Simple Temporal Problem* (STP) [5], which has an associated STN. This network is a distance graph $G_D = (V_D, A_D)$, where $V_D = V \cup \{\alpha\}$ is the set of vertices and where A_D is the set of arcs that represent the constraints of the STP formulation. Note that all constraints are binary, meaning they all contain exactly two variables. A constraint of the form $h_i - h_j \leq \omega_{ij}$ induces an arc from node j to i with weight ω_{ij} in G_D. Figure 2 depicts the STN associated with the instance illustrated in Figure 1(a).

Let $\tau_{i\alpha}$ denote the shortest path distance from i to α in G_D. Then, setting $h_i = -\tau_{i\alpha}$ provides the earliest feasible schedule for the routes of the corresponding IRSP instance. In other words, the LP can be solved by computing shortest paths in G_D [5]. Note, however, that the graph contains cycles and arcs of negative weight. Therefore, one must use methods that can detect negative cycles in graphs, such as the Bellman-Ford algorithm. If G_D has a negative-cost cycle then the STP is inconsistent, implying that the IRSP instance has no feasible solution.

The asymptotic time complexity of the Bellman-Ford algorithm over G_D is $O(|V_D||A_D|)$. The number of arcs $|A_D|$ is $O(|V|)$ given that all nodes $i \in V$ have no more than three outgoing arcs and node α has no more than $|V|$ outgoing arcs. This means that the complexity of determining feasibility of an IRSP instance is $O(|V|^2)$.

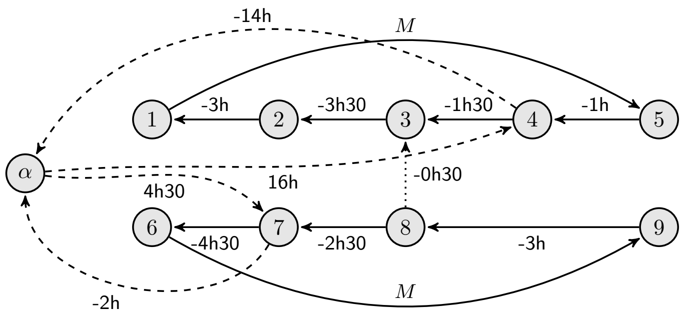


Figure 2 STN created from the instance in Figure 1(a). Dashed arcs denote time window constraints. Service durations have been included in the travel times between nodes.

3.3 Surrogate graph

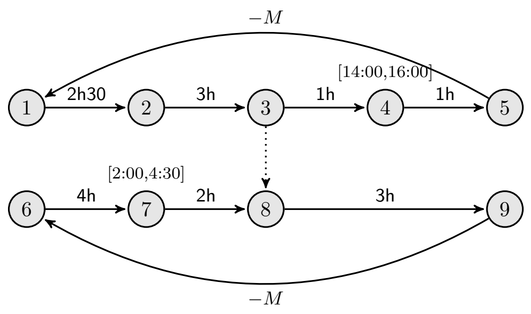
The graph depicted in Figure 1(a) is a directed acyclic graph (DAG). Similar to STNs, the introduction of maximum duration constraints in this DAG creates cycles, as illustrated in Figure 3(a). Exactly $|R|$ maximum duration arcs must be included: one per route.

Computing shortest paths in a DAG, or in the IRSP the earliest feasible start times of service, is straightforward and can be efficiently performed in $O(|V|)$ time (see Appendix B). We are therefore interested in removing the $|R|$ maximum duration arcs that were introduced in order to remove the cycles induced by them while still ensuring compliance with the maximum duration M. To remove these arcs, we employ the strategy introduced by [19] for almost acyclic graphs. We define an associated *surrogate graph* G_S where a new source node α is created. Then, each maximum duration arc of the form $(d_k, o_k, -M)$ is replaced with an arc $(\alpha, o_k, 0)$. In doing so, G_S becomes a DAG. This is illustrated in Figure 3(b).

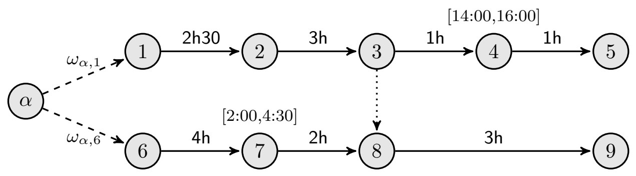
Once G_S has been defined, we can solve the LP (2)–(6) by means of shortest paths employing the *Surrogate Algorithm* [19] outlined in Algorithm 1. This procedure needs to perform no more than $|R| + 1$ iterations of the for-loop (lines 2–9). In each iteration, the start time of service is computed in $O(|V|)$ via the procedure in line 3 (Appendix B), which returns `true` if no time window has been violated and `false` otherwise. At the end of each iteration, the departure time of each route $r_k \in R$ is updated using the current completion time at the destination node d_k and the maximum route duration M (line 5). Updating the departure times corresponds to dynamically updating the weights $\omega_{\alpha o_k}$ of the surrogate arcs $(\alpha, o_k, \omega_{\alpha o_k})$ in G_S. Since every iteration of the for-loop takes $O(|V|)$, the total complexity of the Surrogate Algorithm is $O(|V||R|)$.

Algorithm 1 Surrogate Algorithm.

```
Input: An instance of the IRSP and maximum duration $M$.
Output: Returns true if all routes comply with $M$, and false otherwise.
 1: $p \leftarrow$ true
 2: for i = 0 until $|R|$ do
 3:     $p \leftarrow$ ComputeStartTimeOfService($G_S$)
 4:     if $p =$ true then
 5:         $\omega_{\alpha o_k} \leftarrow \max\{0, h_{d_k} - M\}, \forall r_k \in R$
 6:     else
 7:         goto 10
 8:     end if
 9: end for
10: return $p$
```



(a) Instance modified by introducing maximum duration arcs for each route.



(b) Maximum duration arcs replaced with surrogate arcs from a dummy source node α. Surrogate arcs have variable weights $\omega_{\alpha i}$, which will be updated during the execution of the Surrogate Algorithm.

Figure 3 The surrogate graph of the instance in Figure 1(a).

The correctness and complexity of Algorithm 1 follow directly from [19]. Note that computing the earliest feasible start time of a service corresponds to computing the longest path from α to any node in G_S, which can be accomplished in linear time over a DAG [4].

4 The min-max problem

In the Min-max problem, we seek to minimize the longest duration so as to alleviate the working hours of the drivers who work the most. This is performed even though the duration of some shorter routes is increased in the process. The methods presented to solve Min-max build upon those of the feasibility problem (Section 3).

4.1 Mathematical formulation

The Min-max IRSP can be formulated as an LP by defining a continuous variable $x_{\max}$ to represent the longest duration. The model is:

$$\min \quad x_{\max} \tag{9}$$

$$\text{constraints (2)–(5)}$$

$$x_{\max} \geq (h_{d_k} - h_{o_k}), \quad \forall r_k \in R \tag{10}$$

The current best general-purpose LP algorithm that can solve Min-max is not asymptotically faster than $O^*(|V|^{2.37} \log(|V|/\gamma))$, for a given precision $0 < \gamma \leq 1$ [3][2]. Therefore, we are interested in determining whether it is possible to solve Min-max more efficiently.

[2] Complexity O^* is based on the notation by [3] to hide extra factors (for example, $n^{o(1)}$).

4.2 A special-purpose algorithm

Algorithm 2 outlines a simple procedure to solve Min-max. This algorithm is based on the research introduced by [12] and performs a binary search over the space of route durations in the range $[a, b]$, which is initially $[0, H]$. For every mid-point m in this range, feasibility with respect to maximum duration m is checked using procedure `DurationFeasibility` (line 5). This test can be implemented using any of the feasibility algorithms outlined in Section 3. Limits a and b are subsequently updated according to the feasibility of m (line 6). These steps are repeated as long as $b - a > \epsilon$ for a given precision value $\epsilon > 0$.

In practice, m, a, b and ϵ are floating-point variables and therefore permit only a finite value representation. This implies that Algorithm 2 is guaranteed to finish executing in a finite number of steps. The number of iterations performed in the algorithm is $O(\log H)$. The complexity of each iteration depends on the algorithm employed at line 5. If the STN method is employed, then Algorithm 2 has complexity $O(|V|^2 \log H)$. However, if the Surrogate Graph is used, the complexity is reduced to $O(|V||R| \log H)$ because $|V| > |R|$.

Algorithm 2 Duration minimization.

```
Input: An instance of the IRSP.
Output: Minimum longest duration $x_{\max}$.
1: $a \leftarrow 0$
2: $b \leftarrow H$
3: while $(b - a) > \epsilon$ do
4:     $m \leftarrow (b + a) \cdot 0.5$
5:     $p \leftarrow$ DurationFeasibility($m$)
6:     if $p =$ true then $b \leftarrow m$ else $a \leftarrow m$
7: end while
8: return $b$
```

5 The range minimization problem

The minimization of range is a complicated problem to formulate using an MP when time windows are incorporated [16]. This is because routes may be artificially lengthened by increasing the waiting time at service locations, thereby decreasing the difference between the longest and shortest routes. To avoid unnecessary waiting times, a formulation that forces the start time of all services to be as early as possible was proposed by [16]. However, their scheduling problem was much simpler than the IRSP because (i) routes were independent and (ii) departure times were fixed at $t = 0$. The same modeling ideas thus cannot be applied to the IRSP due to the combination of flexible departure times and interdependent routes.

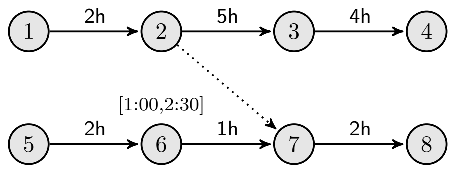
5.1 Mathematical formulation

A naive MP formulation to minimize Range uses the following objective function:

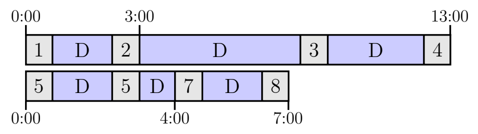
$$\min \quad x_{\max} - x_{\min} \tag{11}$$

However, this function minimizes $x_{\max}$ at the same time that it *maximizes* $x_{\min}$. This, in turn, leads to the situation depicted in Figure 4. The instance in Figure 4(a) has two routes: $r_1 = (1, 2, 3, 4)$ and $r_2 = (5, 6, 7, 8)$. Only node 6 has an associated time window. Service durations are $w_i = 0\text{h}30$, $\forall i \in V$ and the length of the time horizon is $H = 24\text{h}$. There is one minimum difference interdependency $(2, 7, 0\text{h}30) \in A_P$.

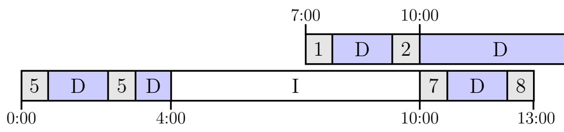
Assuming a departure time for both routes at $t = 0{:}00$, we can produce the solution in Figure 4(b) where the range is 6h. By contrast, the naive MP formulation would produce the solution depicted in Figure 4(c) in which the range is optimal: 0h. Note that such an optimal solution is only possible by incurring 6h of idle time in r_2, even though this is unnecessary. Although mathematically optimal, the solution in 4(c) is very unlikely to be accepted in practice given that it delays several services and it forces the second driver to be in route for a much longer period even though almost half of their working time is idle. Furthermore, this naive formulation could guide a VRP solver to produce solutions with a mathematically perfect balance by creating a lot of idle time in some routes, while other routes would be completely exhausted with working time. From the perspective of the workers, this would be seen as a great *imbalance* in workloads, thereby negatively impacting their morale.



(a) Instance considered in the example.



(b) Solution where $x_{\max} - x_{\min} = 13\text{h} - 7\text{h} = 6\text{h}$.



(c) Solution where $x_{\max} - x_{\min} = 13\text{h} - 13\text{h} = 0\text{h}$.

Figure 4 Instance for which a naive MP fails to correctly minimize the range.

The modeling approach proposed by [16] cannot be applied to the IRSP because the earliest start time of service at interdependent nodes depends on the departure time of the routes, which is flexible. For example, the start time of service at node 2 (and subsequently node 7) depends on the departure time of route r_1. In the IRSP, it does not appear to be possible to force values for the start times of services without sacrificing optimality.

To correctly minimize the range by means of an MP, we propose a two-stage approach. First, we solve the LP from Section 4 to obtain $x_{\max}$. Then, we obtain $x_{\min}$ by solving the following Mixed-Integer Linear Programming (MILP) formulation:

$$\min \quad x_{\min} \tag{12}$$

$$\text{constraints (2)–(5)}$$

$$X_{\max} \geq (h_{d_k} - h_{o_k}), \qquad \forall\, r_k \in R \tag{13}$$

$$x_{\min} \geq (h_{d_k} - h_{o_k}) + H(y_k - 1), \qquad \forall\, r_k \in R \tag{14}$$

$$\sum\nolimits_{k=1}^{|R|} y_k \geq 1 \tag{15}$$

Here $X_{\max}$ refers to a constant value equal to the min-max duration $x_{\max}$. Meanwhile, for each route $r_k \in R$, a binary variable $y_k = 1$ if route r_k has the shortest duration among all in R, otherwise $y_k = 0$. This effectively (de)activates Constraints (14) which set the value of variable $x_{\min}$. Unfortunately, solving MILP (12)–(15) in addition to LP (9)–(10) can create a significant computational overhead. Therefore, we are interested in determining whether a special-purpose algorithm can be defined to minimize range.

5.2 A special-purpose algorithm

Range minimization can also be achieved by Algorithm 3. `DurationMinimizer` is any method capable of solving Min-max, such as those detailed in Section 4. Here, this procedure takes three values as input: a set of routes $R' \subseteq R$ for which the longest duration is to be minimized in addition to the lower and upper bounds (a and b) for the duration of each route.

Algorithm 3 begins by computing $x_{\max}$ (line 1): the min-max duration considering all routes in R. The loop spanning lines 3–6 then attempts to minimize the duration of each route $r_k \in R$ independently in order to produce the minimum duration $x_{\min}$ considering all routes in R. In line 4, `DurationMinimizer` receives as input $R' = \{r_k\}$, $a = 0$ and $b = x_{\max}$, and computes the minimum duration x_k for route r_k. However, the computation of x_k may modify the duration of other routes in R because of the interdependencies, which can subsequently increase the longest duration $x_{\max}$. To avoid this, the duration of all routes $r_z \in R : r_z \neq r_k$ is constrained to be at most $x_{\max}$ when computing x_k. Moreover, the minimization taking place in line 4 is performed without considering the results of previous iterations so as to not interfere with the computation of x_k. The result is then used to update variable $x_{\min}$ (line 5). Finally, the minimum range $x_{\max} - x_{\min}$ is returned at line 7.

Algorithm 3 Range minimization.

```
Input: An instance of the IRSP.
Output: Minimum value for range $x_{\max} - x_{\min}$.
1: $x_{\max} \leftarrow$ DurationMinimizer($R, 0, H$)
2: $x_{\min} \leftarrow +\infty$
3: for each $r_k \in R$ do
4:     $x_k \leftarrow$ DurationMinimizer($\{r_k\}, 0, x_{\max}$)
5:     $x_{\min} \leftarrow \min\{x_{\min}, x_k\}$
6: end for
7: return ($x_{\max} - x_{\min}$)
```

The complexity of Algorithm 3 depends on that of `DurationMinimizer`. If STNs are employed, then the algorithm's complexity is $O(|V|^2|R| \log H)$. However, when using Surrogate Graphs it is $O(|V||R|^2 \log H)$. Alternatively, one could employ a general-purpose LP solver as `DurationMinimizer` by trivially modifying the formulation in Section 4. This would result in a complexity of $O^*(|V|^{2.37}|R| \log(|V|/\gamma))$. However, solving $|R|$ LPs is likely to incur a prohibitive computational overhead despite the polynomial time complexity. In all of these algorithmic variants, the additional $|R|$ derives from the for-loop spanning lines 3–6.

6 Computational experiments

Table 1 summarizes the worst-case asymptotic time complexities when solving the IRSP variants by employing each of the algorithms described in this paper. These complexities indicate that the Surrogate approach represents the fastest method of all the options because the relation $|R| < |V|$ is always valid.

Table 1 Worst-case asymptotic time complexity for the algorithms. Recall that V is the set of nodes and R the set of routes in the IRSP instance, while H denotes the length of the time horizon. Value γ is the desired precision for the LP solver [3].

Problem	STN	Surrogate	MP
Feasibility	$O(\|V\|^2)$	$O(\|V\|\|R\|)$	$O^*(\|V\|^{2.37} \log(\|V\|/\gamma))$
Min-max	$O(\|V\|^2 \log H)$	$O(\|V\|\|R\| \log H)$	$O^*(\|V\|^{2.37} \log(\|V\|/\gamma))$
Range min.	$O(\|V\|^2\|R\| \log H)$	$O(\|V\|\|R\|^2 \log H)$	$O^*(\|V\|^{2.37}\|R\| \log(\|V\|/\gamma))$

In addition to these theoretical results, we have also performed a computational study of the algorithms to examine their processing times for real-sized instances. We implemented all algorithms in `C++` and compiled them using `g++` 7.5 with optimization flag `-O3`. The MP components were implemented using the Gurobi 9 API for `C++` which is a state-of-the-art solver, even though it does not necessarily implement the LP algorithm introduced by [3]. All executions were restricted to a single thread on a computer equipped with an Intel i7-8850H processor at 2.6 GHz, 32 GB of RAM and Ubuntu 18.04 LTS operating system.

IRSP instances were obtained by solving the VRP with multiple synchronization constraints [9]. To produce solutions for the VRP, we employed the Slack Induction by String Removals heuristic [2]. For each new solution, the Min-max and Range equity functions were evaluated using the three algorithms. The IRSP instances that were generated had characteristics with the following ranges: $|V| \leq 300$, $|R| \leq 35$ and $|A_P| \leq 100$. Note that these are already large scale instances for most real-world purposes.

Let us begin the analysis by considering the worst-case performance observed during the experiments. This deserves focus because the algorithms must run as fast as possible even in their worst-case to be safely employed in practice. The graphs in Figure 5 report the maximum recorded execution time in microseconds (μs)[3] according to the number of nodes $|V|$ in the IRSP instance. Due to the significant differences across the algorithms, the graphs are presented in logarithmic scale. The raw data points are plotted directly, while the curves were produced by polynomial interpolation in order to more easily analyze the results.

The MP approach is 20–30 times slower than the other two methods. This is not surprising because Gurobi is a general-purpose solver which incurs significant overhead when addressing structurally simple problems such as the IRSP. Meanwhile, STN solves Min-max 50% quicker and is almost twice as fast when minimizing Range compared to the Surrogate Algorithm. These results clearly contradict the theoretical worst-case time complexity. The reason for this is that STN can detect infeasible maximum durations much faster than Surrogate. This is reflected in the processing times of the two algorithms since, particularly for Range, many feasibility tests must be performed to obtain the optimal solution.

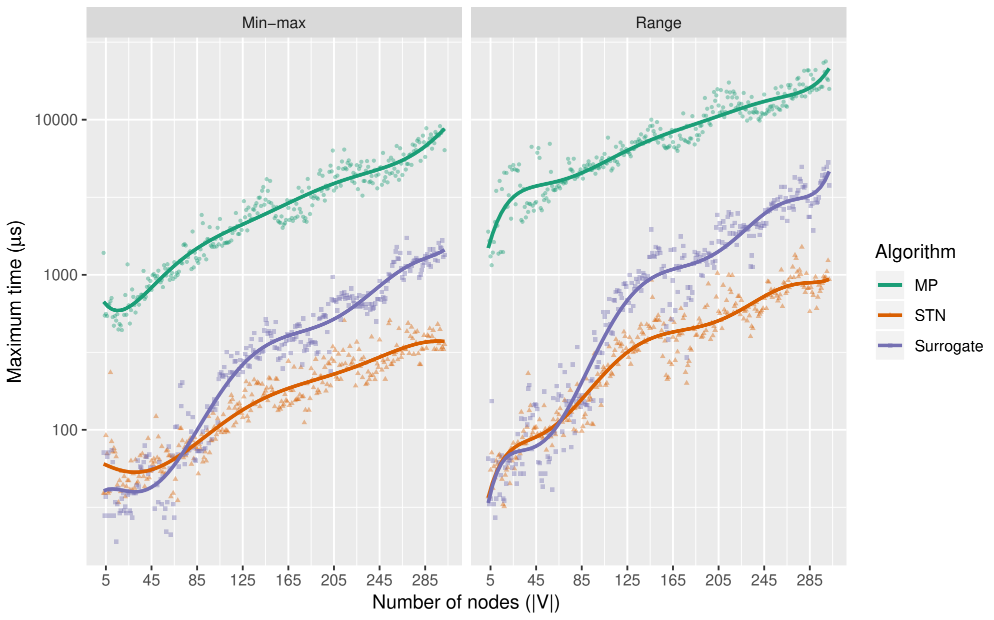


Figure 5 Maximum processing times in microseconds of the three algorithms (logarithmic scale).

[3] Processing times were measured using the `C++` library `std::chrono::high_resolution_clock`.

Alternatively, let us now consider the average processing times of the algorithms. The graphs in Figure 6 report the raw data points for the average processing times as well as an interpolation of the data according to the number of nodes in the IRSP instance, similar to the graphs in Figure 5.

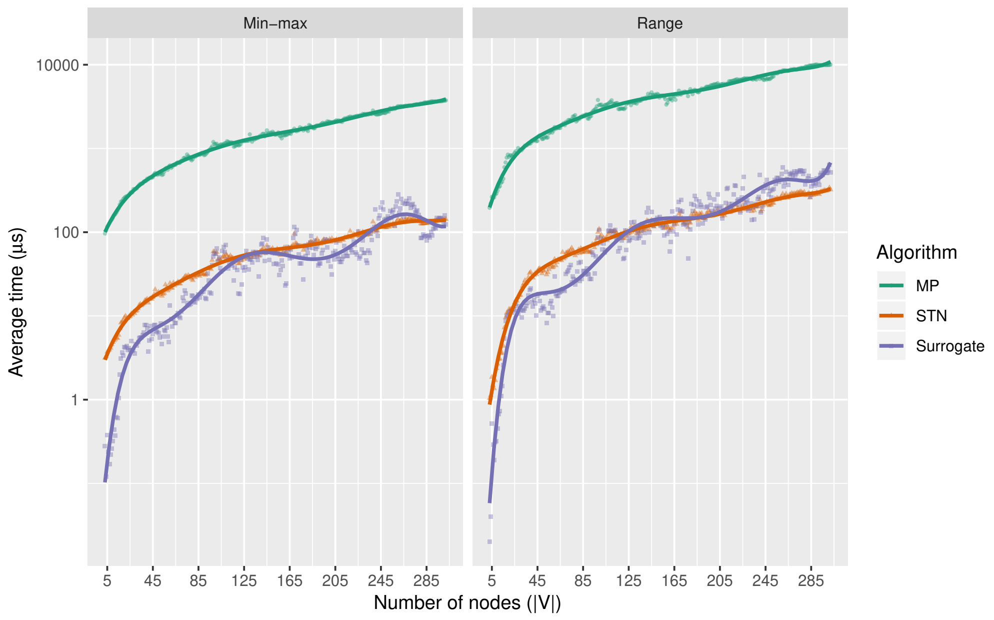


Figure 6 Average processing times in microseconds (μs) of the algorithms (logarithmic scale).

On the one hand, the MP is again 20–25 times slower than the other two methods, on average. On the other hand, comparison of STN and Surrogate is more subtle this time around. Surrogate is 16% faster than the STN when solving Min-max, whereas when minimizing Range the STN is 17% faster. These results are significantly different from the worst-case because Surrogate has more variability in its processing times, while both MP and STN are consistent. Once again, these observations are explained by the fact that Surrogate sometimes requires many iterations to prove infeasibility of a maximum duration M. Meanwhile, in some other instances, Surrogate benefits from its reduced complexity and quickly provides the optimal solution. All of these reasons help explain why, on average, the differences between STN and Surrogate are reduced.

Finally, the experiments indicate that minimizing Range is 2–3 times slower than solving Min-max, which is what one would expect given the time complexities outlined in Table 1. Hence, it may be worth exploring the differences of employing Min-max and Range when balancing workloads, similar to the study conducted by [15] for independent vehicle routes.

7 Conclusion

Interdependent route scheduling is a nontrivial problem when both time windows and flexible departure times must be taken into account. The problem becomes even more challenging when duration-based workload balance between these interdependent routes is desired given how the decisions made for one route can have unforeseen impacts on others, potentially leading to unfair schedules for the drivers. To overcome these challenges, this paper introduced complementary optimization models for balancing duration-based workload among drivers in addition to algorithms for the efficient evaluation of the corresponding equity functions.

The resulting evaluation methods may be employed within, for example, local-search heuristics to produce balanced vehicle routes. Balanced routes help improve the working conditions and morale of the drivers. There are many real-world applications that can benefit from these methods: logistics, transportation, home health care and workforce scheduling.

In spite of our results, many questions remain open. Are there more efficient algorithms to evaluate the Min-max and Range equity functions? Are there alternative approximations that can be employed to compute them faster? Can we extend the methods to address multiple time windows per customer node? More broadly, how can we model an entire VRP with interdependent routes such that the range is minimized? Is it possible to use only one MILP? All of these exciting research opportunities are open for researchers to explore in future studies.

References

1 Ousmane Ali, Jean-François Côté, and Leandro C. Coelho. Models and algorithms for the delivery and installation routing problem. *European Journal of Operational Research*, 291(1):162–177, 2021. doi:10.1016/j.ejor.2020.09.011.

2 Jan Christiaens and Greet Vanden Berghe. Slack induction by string removals for vehicle routing problems. *Transportation Science*, 54(2):417–433, 2020. doi:10.1287/trsc.2019.0914.

3 Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *Journal of the ACM*, 68(1):3:1–3:39, 2021. doi:10.1145/3424305.

4 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT press, 2009.

5 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. doi:10.1016/0004-3702(91)90006-6.

6 Anders Dohn, Matias Sevel Rasmussen, and Jesper Larsen. The vehicle routing problem with time windows and temporal dependencies. *Networks*, 58(4):273–289, 2011. doi:10.1002/net.20472.

7 Michael Drexl. A generic heuristic for vehicle routing problems with multiple synchronization constraints. *Gutenberg School of Management and Economics–Discussion Paper Series: Mainz, Germany*, 1412:43, 2014.

8 Christian Fikar and Patrick Hirsch. Home health care routing and scheduling: A review. *Computers & Operations Research*, 77:86–95, 2017. doi:10.1016/j.cor.2016.07.019.

9 Hossein Hojabri, Michel Gendreau, Jean-Yves Potvin, and Louis-Martin Rousseau. Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. *Computers & Operations Research*, 92:87–97, 2018. doi:10.1016/j.cor.2017.11.011.

10 Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem. In Juan Julián Merelo Guervós, Panagiotis Adamidis, Hans-Georg Beyer, José Luis Fernández-Villacañas Martín, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference, Granada, Spain, September 7-11, 2002, Proceedings*, volume 2439 of *Lecture Notes in Computer Science*, pages 271–280. Springer, 2002. doi:10.1007/3-540-45712-7_26.

11 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984. doi:10.1007/BF02579150.

12 Lina Khatib, Paul H. Morris, Robert A. Morris, and Francesca Rossi. Temporal constraint reasoning with preferences. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 322–327. Morgan Kaufmann, 2001.

13 Renaud Masson, Fabien Lehuédé, and Olivier Péton. The dial-a-ride problem with transfers. *Computers & Operations Research*, 41:12–23, 2014. doi:10.1016/j.cor.2013.07.020.

14 Piotr Matl, Richard F. Hartl, and Thibaut Vidal. Workload equity in vehicle routing problems: A survey and analysis. *Transportation Science*, 52(2):239–260, 2018. doi:10.1287/trsc.2017.0744.

15 Piotr Matl, Richard F. Hartl, and Thibaut Vidal. Workload equity in vehicle routing: The impact of alternative workload resources. *Computers & Operations Research*, 110:116–129, 2019. doi:10.1016/j.cor.2019.05.016.

16 Belén Melián-Batista, Alondra De Santiago, Francisco Ángel-Bello, and Ada M. Alvarez. A bi-objective vehicle routing problem with time windows: A real case in tenerife. *Applied Soft Computing*, 17:140–152, 2014. doi:10.1016/j.asoc.2013.12.012.

17 Snežana Mitrović-Minić and Gilbert Laporte. The pickup and delivery problem with time windows and transshipment. *INFOR: Information Systems and Operational Research*, 44:217–227, 2006.

18 Martin W. P. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *INFORMS Journal on Computing*, 4(2):146–154, 1992. doi:10.1287/ijoc.4.2.146.

19 Donald K. Wagner. Shortest paths in almost acyclic graphs. *Operations Research Letters*, 27(4):143–147, 2000. doi:10.1016/S0167-6377(00)00054-7.

A Types of temporal interdependency

The five most common types of temporal interdependencies encountered in practice as described by [6] can be represented with Equation 1 by correctly parameterizing δ values. Table 2 details the setting of these parameters for each type of constraint when relating two nodes u and v which belong to two different vehicle routes.

Table 2 Definition of δ weights for temporal interdependencies. N/A denotes that no value is assigned (no relation or arc is defined). Here, $\alpha_{\min}$ and $\alpha_{\max}$ are parameters defining the desired minimum and maximum time differences. Table adapted from [6].

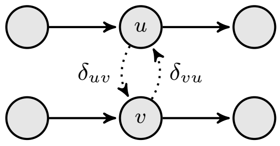
Interdependency	δ_{uv}	δ_{vu}
strict synchronization	0	0
overlap	$-w_v$	$-w_u$
minimum difference	$\alpha_{\min}$	N/A
maximum difference	N/A	$-\alpha_{\max}$
general synchronization	$\alpha_{\min}$	$-\alpha_{\max}$

Appendix B describes how these interdependencies can be modeled in a precedence graph and the impact they can have on the computation of service start times. Particularly, note that *minimum* and *maximum difference* are both unidirectional constraints, while the other three are all bidirectional constraints.

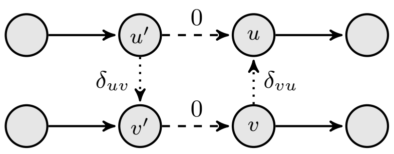
B Computing service start times

Given an instance of the IRSP with a graph $G = (V, A)$, we can compute earliest feasible start time of service h_i at every node $i \in V$ by following a topological ordering of G [4]. In doing so, the computation is guaranteed to be performed in $O(|V|)$ time.

Before going into details about the procedure, we must note that in order to obtain a topological ordering, G must be a DAG. However, Table 2 shows that some interdependencies are bidirectional and therefore incur cycles when represented as a graph. These *cyclic interdependencies* arise whenever two interdependency arcs are required to represent them in a graph. In other words, whenever for two nodes u and v there are arcs (u, v, δ_{uv}) and (v, u, δ_{vu}) in set A_P. Figure 7 illustrates the cycles and how we can trivially eliminate them to obtain a DAG. Figure 7(a) depicts one of the three interdependencies on a graph with a cycle of size two. Fortunately, these cycles can be removed by duplicating nodes as per Figure 7(b), where $w_{u'} = w_{v'} = 0$.



(a) Cyclic interdependency in graph form.



(b) An equivalent directed acyclic graph.

Figure 7 Cyclic interdependencies on a graph.

Once a topological order of G is obtained, we can compute start time of service h_i at every node $i \in V$. First, we set the departure times at origins $o_k, \forall r_k \in R$. For simplicity purposes we assume $h_{o_k} = 0$ for all routes, but in practice any departure time can be set if known or previously computed (for example after each iteration of the Surrogate Algorithm). Then, for each node j in the topological ordering (and such that j is not an origin location), the start time of service h_j is computed by:

$$h_j = \max\{e_j, h_i + w_i + t_{ij}\}, \quad (i, j, t_{ij}) \in A_R$$

However, if j is part of an interdependency constraint, that is, $(u, j, \delta_{uj}) \in A_P$, then we must also take into account the relation captured by Equation 1:

$$h_j = \max\{h_j, h_u + \delta_{uj}\}, \quad \text{if } (u, j, \delta_{uj}) \in A_P \tag{16}$$

Value h_u is always known when computing h_j in Equation 16 thanks to the topological order. In this way, every value h_i, $\forall i \in V$ is computed exactly once and all interdependency relations are respected. If, however, $h_j > l_j$ for any node $j \in V$ then there is an infeasibility and the procedure terminates.