Submitted to *Transportation Science*
manuscript (Please, provide the manuscript number!)

Joint Vehicle and Crew Routing and Scheduling

Edward Lam
Monash University and CSIRO Data61, edward.lam@monash.edu,

Pascal Van Hentenryck
Georgia Institute of Technology, pascal.vanhentenryck@isye.gatech.edu,

Phil Kilby
CSIRO Data61, philip.kilby@data61.csiro.au,

Traditional vehicle routing problems implicitly assume that only one crew operates a vehicle for the entirety of its journey. However this assumption is violated in many applications arising in humanitarian and military logistics. This paper considers a Joint Vehicle and Crew Routing and Scheduling Problem, in which crews are able to interchange vehicles, resulting in space and time interdependencies between vehicle routes and crew routes. The problem is formulated as Mixed Integer Programming (MIP) and a Constraint Programming (CP) models that overlay crew routing constraints over a standard vehicle routing problem. The constraint program uses a novel optimization constraint to detect infeasibility and to bound crew objectives. The paper also explores methods using large neighborhood search over the MIP and CP models. Experimental results indicate that modeling the vehicle and crew routing problems jointly and supporting vehicle interchanges for crews may bring significant benefits in cost reduction compared to a method that sequentializes these decisions.

Key words: vehicle routing, vehicle scheduling, crew routing, crew scheduling, synchronization, mixed integer programming, constraint programming, hybrid optimization, large neighborhood search
History:

1. Introduction

A vehicle routing problem (VRP) models a fleet of vehicles that visit various locations to perform their duties. The task is to find least-cost routes that adhere to various restrictions on the routes and on the vehicles. Even though VRPs can be solved in isolation, in practice, they typically appear within a sequential optimization process that first optimizes vehicle routes and then schedules crews given the vehicle routes (Barnhart, Lu, and Shenoi 1998). This sequential process has the advantage of reducing the computational complexity. However, by designing vehicle routes first,

sequential approaches may lead to suboptimal or even infeasible crew schedules since decisions in the vehicle routing phase may ignore crew constraints and objectives. Therefore, it is desirable to simultaneously consider vehicle and crew constraints and objectives, particularly for cases in which crew constraints are tight or crew costs exceed vehicle costs.

This paper proposes the Joint Vehicle and Crew Routing and Scheduling Problem (JVCRSP), which adds a second layer of routing for crews to a classical VRP. In many applications of VRPs, goods are moved from one location to another, usually across the course of a day. The JVCRSP is motivated by applications in humanitarian and military logistics; in these contexts, vehicles (e.g., airplanes) travel long routes and transport food and medical supplies across time horizons that can span several days, and hence, determining crew schedules becomes an important part of the problem. For example, vehicles must be operated by crews, which have limitations on their duty times. Crews are able to interchange vehicles at different locations and to travel as passengers before and after their duty times. The JVCRSP is extremely challenging computationally because vehicle routes and crew routes are interdependent. Allowing crews to interchange vehicles adds an additional time element to the problem since two vehicles must be synchronized in order for an exchange to proceed. It is thus necessary to decide whether vehicles wait and for how long at a location because both vehicles must be present at the same location for an interchange to occur.

This paper develops a mixed integer programming and a constraint programming formulation of the JVCRSP that jointly optimize vehicle and crew routing and scheduling in the hope of remedying some limitations of sequential approaches. The formulations overlay crew routing constraints over the Pickup and Delivery Problem with Time Windows (PDPTW) and add a number of synchronization constraints to link the vehicles and crews. In addition, the constraint programming formulation includes a novel global optimization constraint that uses a linear relaxation to check whether the current crew partial routes are feasible and to bound crew costs, which is crucial in the early stages of the search when the focus is on vehicle routing.

Both the mixed integer programming and constraint programming models are each developed into two additional models that sequentialize the vehicle routing and scheduling components in order to evaluate the impacts of simultaneously optimizing these decisions. These six models are then solved using a regular branch-and-bound complete tree search and a large neighborhood search, giving a total of twelve methods.

Experimental results on instances with up to 100 requests and three cost functions indicate that (1) the joint optimization of vehicle and crew routing can produce considerable benefits over sequential methods, (2) the combination of constraint programming and large neighborhood search scales significantly better than pure constraint programming and mixed integer programming approaches, and (3) vehicle interchanges are critical for obtaining high-quality solutions. These findings indicate

that it is now in the realm of optimization technology to simultaneously optimize vehicle and crew routing and scheduling in a single model, and that concurrently modeling vehicle and crew routing may bring significant benefits in cost reduction compared to methods that sequentialize these decisions.

The remainder of this paper is organized as follows. Section 2 reviews existing work on related problems. Section 3 describes the JVCRSP. Section 4 discusses a high-level model of the problem. Sections 5 and 6 concretizes the high-level model as a mixed integer programming model and a constraint programming model. Section 7 describes the large neighborhood search common to both the mixed integer programming and constraint programming models. Section 8 summarizes the experimental results, which are detailed in the online appendix, and Section 9 concludes this paper. This paper is significantly expanded from a previously published conference paper (Lam, Van Hentenryck, and Kilby 2015).

2. Literature Review

Simultaneous vehicle routing and crew scheduling problems have not attracted much interest in the literature at this point, probably due to their inherent complexity (Drexl 2012). This section reviews three relevant problems from the literature.

Kim, Koo, and Park (2010) considered a problem in which vehicles transport teams to service customers. Vehicles are able to move without any team on board. The problem features three types of tasks that must be serviced in order, and all customers have one task of each type. Each team can only service one compatible type of task. The mixed integer programming formulation has variables indexed by five dimensions and is intractable. The paper develops a simple local search algorithm that is embedded within a particle swarm metaheuristic. This approach was developed specifically for the problem and cannot easily accommodate side constraints.

Hollis, Forbes, and Douglas (2006) solved a mail distribution problem that features multiple depots at which vehicle routes begin and end. The model was solved using a two-stage heuristic column generation approach, which cannot be guaranteed to solve the problem to optimality. In the first stage, trips that begin and end at the depots are computed. The second stage takes a scheduling approach and assigns vehicles and crews to the trips. Vehicle interchange can only occur at the depots at the start or end of a trip. In addition, the model features a 24-hour cyclic time period and variables indexed by discretized blocks of time.

Drexl et al. (2013) considered a problem that includes European legislation and relay stations where drivers rest and interchange vehicles. Vehicles must wait a fixed amount of time upon reaching a relay station. Vehicle interchange can only occur if other drivers arrive at this relay station during this time interval. The problem also provides a shuttle service, separate from the fleet of vehicles,

that can be used to move drivers between relay stations. The problem is solved using a two-stage large neighborhood search method. In the first stage, a vehicle routing problem is solved and the resulting routes form the customers of another vehicle routing problem in the second stage, in which the crews perform the role of vehicles. Observe that this approach also cannot jointly optimize vehicle and crew routing. The model features several fixed parameters, such as the duration during which a vehicle waits at a relay station, and a search procedure that only explores a limited number of nearby relay stations. Both these restrictions greatly reduce the search space but negatively impact vehicle interchange, leading the authors to conclude that allowing for vehicle interchange does not significantly improve the objective value.

Drexl (2012) classified simultaneous vehicle routing and crew scheduling problems as a Vehicle Routing Problem with Multiple Synchronization constraints (VRPMS). Synchronization is a feature present in some vehicle routing problems, in which decisions about one object (e.g., vehicle, route, request) imply actions that may or must be taken on other objects.

Drexl (2014, 2007) developed a VRPMS called the Vehicle Routing Problem with Trailers and Transshipments (VRPTT). It features two vehicle classes: lorries which can move independently, and trailers which must be towed by an accompanying lorry. All lorries begin at a single depot with or without a trailer. A lorry can detach its trailer at transshipment locations to visit customers who are unable to accommodate a trailer (e.g., due to size). Lorries can transfer load into and/or attach with any trailer at any transshipment location. A lorry that has detached its trailer can also return to the depot without reattaching a trailer, leaving its trailer behind at a transshipment location to be collected by another lorry at a future time. Several sophisticated mixed integer programming formulations were presented, which were solved using branch-and-cut on instances with up to eight customers, eight transshipment locations and eight vehicles. Drexl (2013) argued that simultaneous vehicle routing and crew scheduling problems can be reformulated into the VRPTT by casting crews as lorries with zero capacity, and vehicles as trailers.

The VRPTT is most closely related to the JVCRSP since lorry routes and trailer routes are jointly computed, and the search space is not artificially limited. A key difference is that the VRPTT includes load synchronization, which is not considered in the JVCRSP. Load synchronization refers to the ability to either transfer load from one vehicle to another while both are present, or partially or fully transporting load to a transshipment location and then having a different vehicle retrieve the load to deliver it to its final destination.

Finally, observe that vehicle and crew *scheduling* problems are thoroughly studied (e.g., Haase, Desaulniers, and Desrosiers 2001, Mesquita and Paias 2008, Freling, Wagelmans, and Paixão 1999, Freling, Huisman, and Wagelmans 2001, 2003, Cordeau et al. 2001, Mercier, Cordeau, and Soumis 2005, Mercier and Soumis 2007). These problems aim to assign vehicles and crews to a

predetermined set of trips, with each trip consisting of a fixed route and usually with fixed arrival and departure times. Trips in these problems correspond to parts of a route in the JVCRSP, which are not available a priori, but instead, must be computed during search, thereby increasing the computational challenges.

3. Problem Description

This section describes the application, reviews several classical VRPs, and introduces the JVCRSP as an abstraction of interesting elements of the original problem. Given the complexity of the overall problem, the research methodology is to find effective solution approaches to interesting aspects of the application before addressing the problem as a whole.

Motivating Application The JVCRSP is motivated by an air transportation problem faced by the Royal Australian Air Force. The challenge is to design effective routes for moving goods, known as parcels, around the world. The majority of parcels originate in Australia and are destined for the Asia-Pacific region. Most parcels only have a due date for arrival; parcels rarely have a due date for departure, possibly because of ample storage space at military bases. The parcels are associated with a weight and a 3-dimensional size of length, width and height. The sizes of the parcels vary: they can be as small as a spare part for a plane or as large as a tank. They can also include doctors moving to other bases in order to perform urgent medical procedures on patients, moving soldiers on and off rotations, moving food and medical supplies to bases, keeping inventories of spare parts, delivering mail, etc. To simplify the problem, this study only considers the weights and due dates. Obviously, the full problem contains 3D-packing constraints for the airplanes as well.

The planes are operated by crews, who have limited flying time. They will often fly to a base and hand over the plane to another crew. The crew will then spend some recovery time at that base and then continue in the next plane that arrives. Crews can also be moved on planes to other bases if needed. The abstracted problem considered in this paper only accounts for one duty period. The full problem expands the number of duty periods, which is modeled exactly as in the paper with the addition of a minimum rest time between duty periods. The minimum rest time depends on the previous flight times, e.g., 12 hours rest for 24 hours of flight, 60 hours rest for 96 hours of flight, and 4 days rest for 14 days of flight.

Additionally, the full problem restricts the number of planes that can occupy an airfield at any given time because of the limited availability of taxiing and parking space. This aspect of the motivating application was studied by Lam and Van Hentenryck (2016) using a branch-and-price-and-check approach.

The JVCRSP aims at capturing interesting elements of the motivating application surrounding the two layers of interdependent vehicle and crew routing. The remainder of this section describes several

classical VRPs and generalize them to the JVCRSP. Readers seeking an extensive introduction to vehicle routing problems can consult the book by Vigo and Toth (2014).

Classical VRPs The Capacitated Vehicle Routing Problem (CVRP) is a basic problem that requires vehicles to depart a central depot to pick up goods and then deliver them to the same depot at the end of their routes. Each pickup, called a *request*, is associated with a weight, called its *load*. All vehicles are identical, and each vehicle can carry requests up to a maximum total weight, called the *vehicle capacity*. The goal of the CVRP is to find routes for these vehicles that pick up all requests while minimizing the total travel distance.

The Vehicle Routing Problem with Time Windows (VRPTW) extends the CVRP with time constraints. Every request is associated with a time frame, called its *time window*, within which the request must be picked up. Vehicles can arrive at the location of a request prior to the opening of its time window but must wait until its time window opens before commencing service.

The Vehicle Routing Problem with Pickup and Delivery and Time Windows (VRPPDTW), also known as the Pickup and Delivery Problem with Time Windows (PDPTW), generalizes the VRPTW. The PDPTW models *pickup-delivery pairs* instead of single pickup requests. Each pair is associated with both a pickup request and a delivery request. Every pickup must be brought to its destination during a route instead of the central depot at the end of a route. The load of a vehicle cannot exceed its capacity at any time along its route; although making a delivery will reduce its load and free up the capacity to pick up other requests. Since all pickups must be delivered, all vehicles return empty to the central depot.

The JVCRSP extends the PDPTW with two major additions. First, it groups requests by location, i.e., every request has an attribute for its location. This contrasts with traditional vehicle routing problems, in which locations are synonymous with requests. Grouping requests by location makes it possible to model crews interchanging vehicles because the model can recognize if two vehicles are present at the same location.

Second, the JVCRSP adds crews to the PDPTW. Crews must travel on a vehicle when traveling from one location to another, and are free to switch vehicles at any location. Hence, crews can exit a location on a vehicle different to the one on which they entered. Every vehicle can carry an unlimited number of crews onboard as passengers, and one of the crews onboard, known as the driver, must operate the vehicle whenever it travels from one location to another.

Each crew is restricted to at most one driving segment, which is defined as the time period from the beginning of a crew's first drive to the end of the crew's last drive. The driving segment is limited to a maximum duration. During the driving segment, the crew may interchange vehicles to drive on other vehicles and may travel as a passenger to reach a vehicle before recommencing driving on this vehicle. The time taken for traveling as a passenger is included within the driving

segment. Crews can travel on any vehicle to reach the location of their first drive, and crews can travel on any vehicle back to the depot after driving. No distance or time limitations are placed on crews before and after the driving segment.

The JVCRSP seeks to minimize a weighted sum of the number of vehicles and crews used, as well as the total vehicle and crew travel distances.

4. High-Level Modeling Concepts

This section presents several modeling decisions underlying the constraint programming and mixed integer programming models and concepts relevant to solving the JVCRSP.

Figure 1 illustrates an example of several vehicle routes and crew routes for a problem with three vehicles, five crews and five locations. Every crew begins at the common crew start node (CS) and either moves directly to the crew end node (CE), signifying that the crew is unused, or proceeds to a vehicle start node (S1 to S3) to board a vehicle. The vehicles and crews visit the locations (L1 to L5) to service requests and then proceed to the vehicle end nodes (E1 to E3), where the vehicles complete their routes. The crews then disembark the vehicles and proceed to the crew end node (CE). Observe that the yellow crew changes vehicles at L3 and then drives the blue vehicle. Also, observe that the blue crew changes vehicles at L3, travels as a passenger on the red vehicle and then changes vehicles at L5 to drive on the green vehicle. Because the driving segment of a crew is defined from the moment that it starts driving to the moment that it finishes driving, the driving segment of the blue crew is its entire route from S1 to E3.

Like in conventional vehicle routing problems, a route in the JVCRSP is a sequence of requests. However, each request is associated with an additional input parameter that maps it to a location. Along a route, a subsequence of requests at the same location can be thought of as a visit to the location, with an entry at the first request in the subsequence and an exit at the last request in the subsequence.

Figure 2 illustrates a route that visits two locations. A vehicle departs the starting node S to service requests 1 to 3 at a location and then moves to the next location to service requests 4 to 6 before finishing at the end node E. Even though a vehicle route is a sequence of requests, semantically, the vehicle enters the first location at request 1, departs it at request 3, enters the second location at request 4 and departs it at request 6.

The model employs a number of crew routing constraints, which mirror the vehicle routing constraints. Vehicles and crews are synchronized using constraints that require

- crews to move with a vehicle when moving from one location to another,
- vehicles to have exactly one driver onboard when moving from one location to another, and
- the driver of a vehicle to be one of the crews onboard.

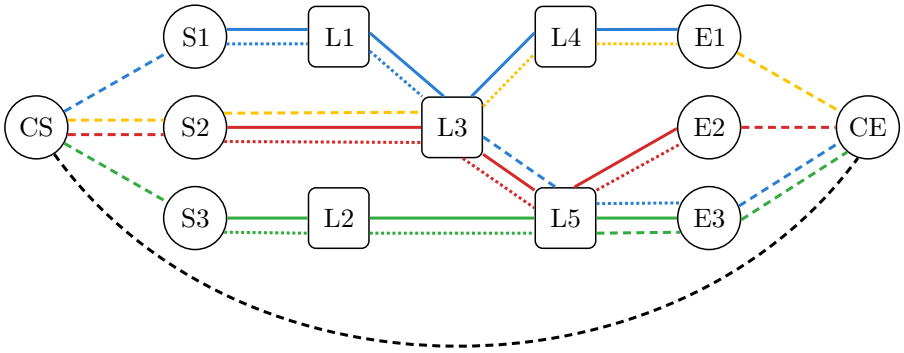


Figure 1 **Example of vehicle routes and crew routes. Vehicle routes are marked by solid lines and crew routes are marked by dashed lines and dotted lines. Crews travel as passengers on dashed lines and as drivers on dotted lines.**

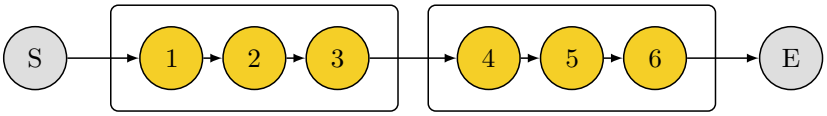


Figure 2 **Example of locations along a vehicle route.**

These requirements allow vehicles and crews to move independently within a location.

Figure 3 illustrates two crews switching vehicles at a location. The location contains requests 1 to 6. A vehicle and crew enter the location at request 1, and another vehicle and crew enter the location at request 4. While the two vehicles are servicing requests, the first crew moves to the departure request of the second vehicle (request 6), and the second crew moves to the departure request of the first vehicle (request 3). The two crews wait for their new vehicles to complete servicing the requests then leave the location. In order for this interchange to occur, both vehicles must be at the same location at the same time. Suppose requests 1 to 3 require 10 units of time for service, and requests 4 to 6 require 5 units of time for service. The first crew can move to the second vehicle because the crew arrived (at time 10) before the departure of the second vehicle (at time 75). The second crew can move to the first vehicle which waits until time 60 to depart, even though the service for request 3 is completed at time 40.

The crew constraints described above permit many symmetrical solutions due to the numerous subpaths that a crew can travel on within a location. The specific requests that a crew visits at a particular location are not important; what is significant is the two vehicles that the crew uses to arrive at and depart from the location. In other words, there are many symmetrical solutions that differ only by the path that a crew takes within a location. These symmetries are best explained using Figure 3. It is possible for the first crew to enter the location at request 1, visit requests 2 to 5 in order then exit on the second vehicle at request 6. However, this path is equivalent, for all practical purposes, to one in which the crew moves directly from request 1 to request 6. Hence, the

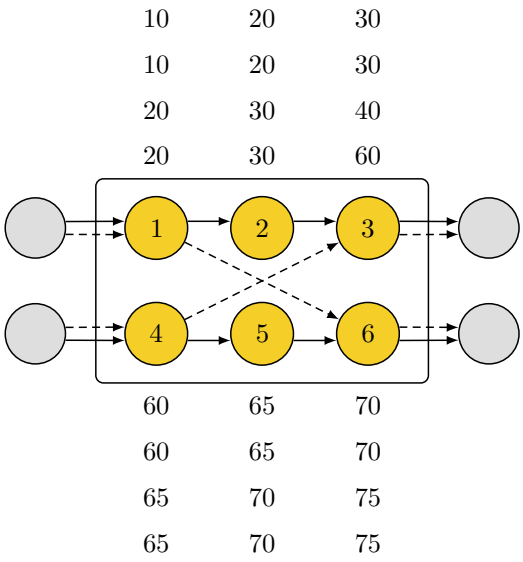
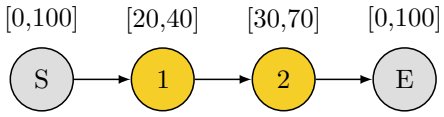


Figure 3 **Example of two crews interchanging vehicles at a location. Vehicles are marked by solid lines and crews are marked by dashed lines. Each node has its arrival time, start of service, end of service and departure time labeled from top to bottom.**

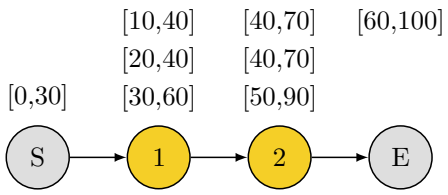
search space can be reduced by requiring crews to shortcut intermediate nodes within a location by visiting at most a single entry node and a single exit node per location. This is accomplished by ensuring that crews cannot visit a subsequence of three or more requests at the same location. This restriction allows for vehicle interchanges without considering all the symmetric solutions that have no impact on the objective.

The JVCRSP features a temporal element that emerges from the interdependency between vehicle routes and crew routes. This time complexity is not present in traditional vehicle routing problems. Consider a route in the PDPTW, as shown in Figures 4a and 4b. Figures 4c and 4d show two different schedules for the same vehicle route. Delaying the departure times along one route does not impact other routes nor invalidate the solution (provided that the delayed route satisfies the time windows). The time variables in classical vehicle routing problems are used solely to enforce the feasibility of the time windows. Solvers frequently use this knowledge to avoid branching on time variables by fixing all time variables to their earliest possible once all routes are determined. Because solvers only need to search on the arcs (i.e., space) variables and not the time variables, classical vehicle routing problems are said to possess one spatial degree of freedom and zero temporal degrees of freedom.

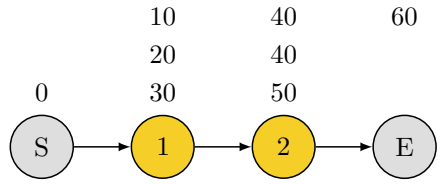
Figure 5 shows two crews that switch vehicles at a location. Delaying the top vehicle to depart at time 60, instead of departing immediately after service at time 40, allows the bottom crew to move onboard. This delay alters the departure times and can cause a cascade of events farther along this



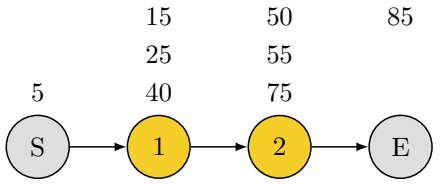
(a) A route consisting of two requests and the start and end depot nodes, and their time windows for the start of service. Every request requires 10 time units for service, and every arc requires 10 time units of travel time.



(b) Lower and upper bounds for the arrival time (top), service start time (middle) and departure time (bottom) at each request along the route can be easily computed by considering travel time and time window constraints.



(c) Since early arrival is permitted, arrival times, service times and departure times can be fixed to their earliest possible once a route is determined.



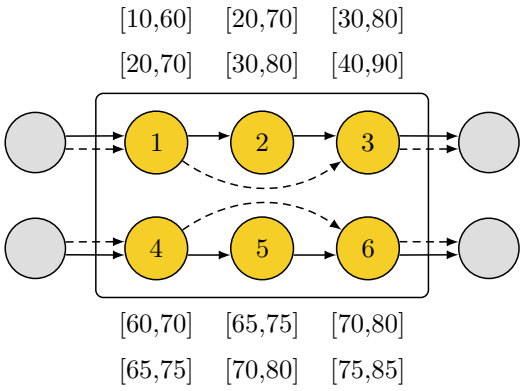
(d) Delaying service or departure along a route, although possible, is not beneficial since an amount of time is wasted, which could be better used to service other requests.

Figure 4 **Example of two different schedules for the same route in classical vehicle routing problems without time synchronization.**

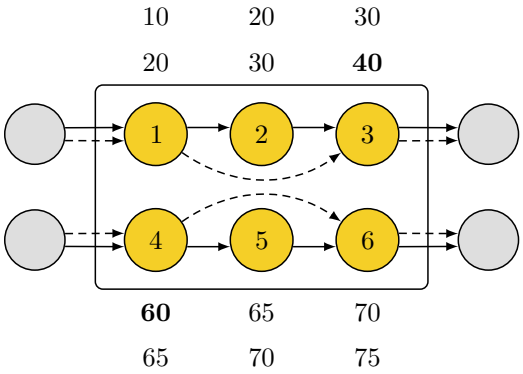
route and on other routes. For example, the delay may allow a crew to interchange vehicles at one location but prevent a crew from interchanging vehicles at another location. Hence, searching on the arrival and departure time variables is essential in the JVCRSP.

The JVCRSP has two spatial degrees of freedom because vehicles and crews can move independently in space. However, the JVCRSP has only one temporal degree of freedom because the crew time variables are tightly coupled to the vehicle time variables since the crews move with vehicles between locations and crews have no notion of time within locations. Hence, the crew time variables serve a similar purpose to the vehicle time variables in classical vehicle routing problems, such as the PDPTW, in that they exist only to ensure the feasibility of the maximum driving duration constraints.

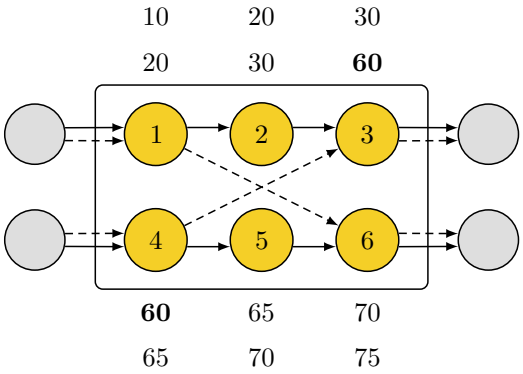
It is the temporal degree of freedom that makes the JVCRSP difficult, especially when the time horizon is large, because the solver must test many combinations of assignments to the time variables to determine if crews can interchange vehicles or if they have reached the maximum driving duration.



(a) Consider two vehicles (solid lines) and two crews (dashed lines) that arrive at a location. Next to each request at the location is the lower and upper bounds for its arrival time (top) and departure time (bottom). Requests 1 to 3 require 10 units of time for service, and requests 4 to 6 require 5 units of time for service.



(b) Setting all time variables to the earliest possible will prohibit the bottom crew from changing to the top vehicle because the top vehicle departs the location (at time 40) before the arrival of the bottom vehicle and crew (at time 60).



(c) Delaying the departure of the top vehicle to time 60 allows both the top and bottom crews to exchange vehicles.

Figure 5 Example of the significance of branching on the time variables when vehicle routes are interdependent.

5. The Mixed Integer Programming Model

This section concretizes the high-level JVCRSP model into a mixed integer program.

The inputs and decision variables for the model are listed in Table 1. The problem is defined on a time interval $\mathcal{T} = [0, T]$, where $T > 0$ is the time horizon when all $V \in \{1, \ldots, \infty\}$ vehicles and $C \in \{V, \ldots, \infty\}$ crews must have completed their routes. The vehicles and crews are represented by the sets $\mathcal{V} = \{1, \ldots, V\}$ and $\mathcal{C} = \{1, \ldots, C\}$ respectively. Each vehicle has a load capacity of $Q \geq 0$ and each crew has a maximum driving duration of $\bar{T} \in \mathcal{T}$.

Assume that there are $P \in \{1, \ldots, \infty\}$ pickup-delivery pairs, and hence, $2P$ requests in total. Define $\mathcal{P} = \{1, \ldots, P\}$ and $\mathcal{D} = \{P + 1, \ldots, 2P\}$ as the set of pickup nodes and delivery nodes

Name	Description
$T > 0$	Time horizon.
$\mathcal{T} = [0, T]$	Time interval.
$V \in \{1, \ldots, \infty\}$	Number of vehicles.
$\mathcal{V} = \{1, \ldots, V\}$	Set of vehicles.
$Q \geq 0$	Vehicle capacity.
$C \in \{V, \ldots, \infty\}$	Number of crews.
$\mathcal{C} = \{1, \ldots, C\}$	Set of crews.
$\bar{T} \in \mathcal{T}$	Crew maximum driving duration.
$P \in \{1, \ldots, \infty\}$	Number of pickup-delivery pairs.
$\mathcal{P} = \{1, \ldots, P\}$	Set of pickup nodes.
$\mathcal{D} = \{P+1, \ldots, 2P\}$	Set of delivery nodes.
$\mathcal{R} = \mathcal{P} \cup \mathcal{D}$	Set of all requests.
s_v	Start node of vehicle $v \in \mathcal{V}$.
e_v	End node of vehicle $v \in \mathcal{V}$.
$\mathcal{S} = \{s_1, \ldots, s_V\}$	Set of vehicle start nodes.
$\mathcal{E} = \{e_1, \ldots, e_V\}$	Set of vehicle end nodes.
$\mathcal{N} = \mathcal{R} \cup \mathcal{S} \cup \mathcal{E}$	Set of all requests and vehicle start and end nodes.
s_{crew}	Start node of all crews.
e_{crew}	End node of all crews.
$\mathcal{A}_v$	Arcs that can be traversed by vehicle $v \in \mathcal{V}$. Defined in Equation (1).
$\mathcal{A}$	Arcs that can be traversed by crews. Defined in Equation (2).
$\mathcal{L}$	Set of locations, including one depot location.
$l_i \in \mathcal{L}$	Location of $i \in \mathcal{N}$.
$a_i \in \mathcal{T}$	Earliest start of service at $i \in \mathcal{N}$.
$b_i \in \mathcal{T}$	Latest start of service at $i \in \mathcal{N}$.
$t_i \in \mathcal{T}$	Service duration of $i \in \mathcal{N}$.
$q_i \in [-Q, Q]$	Load demand at $i \in \mathcal{N}$.
$d_{i,j} \in \mathcal{T}$	Distance and travel time along the arc $(i,j) \in \bigcup_{v \in \mathcal{V}} \mathcal{A}_v \cup \mathcal{A}$.
$w_1 > 0$	Cost of using one vehicle.
$w_2 > 0$	Cost of using one crew.
$w_3 > 0$	Cost of one unit of vehicle distance.
$w_4 > 0$	Cost of one unit of crew distance.
$\text{veh}_{v,i,j} \in \{0,1\}$	Indicates if vehicle $v \in \mathcal{V}$ traverses $(i,j) \in \mathcal{A}_v$.
$\text{arr}_{v,i} \in \mathcal{T}$	Arrival time of vehicle $v \in \mathcal{V}$ at $i \in \mathcal{N}$.
$\text{serv}_{v,i} \in [a_i, b_i]$	Start of service by vehicle $v \in \mathcal{V}$ at $i \in \mathcal{N}$.
$\text{dep}_{v,i} \in \mathcal{T}$	Departure time of vehicle $v \in \mathcal{V}$ at $i \in \mathcal{N}$.
$\text{load}_{v,i} \in [0, Q]$	Load of vehicle $v \in \mathcal{V}$ after servicing $i \in \mathcal{N}$.
$\text{crew}_{c,i,j} \in \{0,1\}$	Indicates if crew $c \in \mathcal{C}$ traverses $(i,j) \in \mathcal{A}$.
$\text{crewTime}_{c,i} \in \mathcal{T}$	Time when crew $c \in \mathcal{C}$ is at $i \in \mathcal{N}$.
$\text{driver}_{c,i,j} \in \{0,1\}$	Indicates if crew $c \in \mathcal{C}$ drives on $(i,j) \in \mathcal{A}$, $l_i \neq l_j$.
$\text{driveStart}_c \in \mathcal{T}$	Start time of driving for crew $c \in \mathcal{C}$.
$\text{driveEnd}_c \in \mathcal{T}$	End time of driving for crew $c \in \mathcal{C}$.
$\text{driveDur}_c \in [0, \bar{T}]$	Driving duration of crew $c \in \mathcal{C}$.

Table 1 The data and decision variables of the mixed integer programming model.

respectively, and let $\mathcal{R} = \mathcal{P} \cup \mathcal{D}$ be the set of all request nodes. Every vehicle $v \in \mathcal{V}$ has a unique start node s_v and end node e_v. These are grouped in the sets $\mathcal{S} = \{s_1, \ldots, s_V\}$ and $\mathcal{E} = \{e_1, \ldots, e_V\}$. Also, define $\mathcal{N} = \mathcal{R} \cup \mathcal{S} \cup \mathcal{E}$ to be all nodes that vehicles can visit. Finally, let s_{crew} and e_{crew} be the common crew start node and end node respectively.

Each vehicle can traverse arcs from its start node to any pickup, from any request to any other request, from deliveries to its end node, and if the vehicle is unused, from its start node to end node. For every vehicle $v \in \mathcal{V}$, define this set of arcs as

$$\mathcal{A}_v = \{(s_v, i) | i \in \mathcal{P}\} \cup \{(i, j) | i \in \mathcal{R}, j \in \mathcal{R}, i \neq j\} \cup \{(i, e_v) | i \in \mathcal{D}\} \cup \{(s_v, e_v)\}. \tag{1}$$

Crews can traverse all vehicle arcs except the arcs indicating that a vehicle is unused. Additionally, crews can traverse arcs from the crew start node to any vehicle start node, from any vehicle end node to the crew end node, and directly from the crew start node to the crew end node. Define the common crew arcs as

$$\mathcal{A} = \{(s_{\text{crew}}, i) | i \in \mathcal{S}\} \cup \bigcup_{v \in \mathcal{V}} \mathcal{A}_v \cup \{(i, e_{\text{crew}}) | i \in \mathcal{E}\} \cup \{(s_{\text{crew}}, e_{\text{crew}})\} \setminus \{(s_v, e_v) | v \in \mathcal{V}\}. \tag{2}$$

Define $\mathcal{L}$ as the set of locations, including one depot location. For every node $i \in \mathcal{N}$, let $l_i \in \mathcal{L}$ be its location, $a_i \in \mathcal{T}$ and $b_i \in \mathcal{T}$ be its earliest time and latest time to start service, $t_i \in \mathcal{T}$ be its service duration and $q_i \in [-Q, Q]$ be its load demand. For every arc $(i, j) \in \bigcup_{v \in \mathcal{V}} \mathcal{A}_v \cup \mathcal{A}$, define $d_{i,j} \in \mathcal{T}$ as the distance and travel time along the arc. Finally, let $w_1 > 0$ and $w_2 > 0$ be the cost of using one vehicle and one crew, and let $w_3 > 0$ and $w_4 > 0$ be the cost of one unit of distance traveled by a vehicle and by a crew.

The primary decision variables are the usual vehicle flow variables $\text{veh}_{v,i,j} \in \{0, 1\}$, which indicate whether vehicle $v \in \mathcal{V}$ traverses $(i, j) \in \mathcal{A}_v$. The variables $\text{arr}_{v,i} \in \mathcal{T}$, $\text{serv}_{v,i} \in [a_i, b_i]$ and $\text{dep}_{v,i} \in \mathcal{T}$ represent the arrival time, service start time and departure time of vehicle $v \in \mathcal{V}$ at node $i \in \mathcal{N}$. The $\text{load}_{v,i} \in [0, Q]$ variable contains the load of vehicle $v \in \mathcal{V}$ after it services node $i \in \mathcal{N}$.

The secondary decision variables are the crew flow variables $\text{crew}_{c,i,j} \in \{0, 1\}$, which indicate whether crew $c \in \mathcal{C}$ traverses $(i, j) \in \mathcal{A}$. Variable $\text{crewTime}_{c,i} \in \mathcal{T}$ stores a moment when crew $c \in \mathcal{C}$ is present at node $i \in \mathcal{N}$. In our solutions, it will be either the arrival or the departure time at the node (or both if they are equal). The $\text{driver}_{c,i,j} \in \{0, 1\}$ variable indicates whether crew $c \in \mathcal{C}$ drives the vehicle that traverses $(i, j) \in \mathcal{A}$ if $l_i \neq l_j$. The start and end time of the driving segment of crew $c \in \mathcal{C}$ is given by $\text{driveStart}_c \in \mathcal{T}$ and $\text{driveEnd}_c \in \mathcal{T}$, and the total driving duration by $\text{driveDur}_c \in [0, \bar{T}]$.

The constraints of the mixed integer programming model are separated into a vehicle component and a crew component. The vehicle component, depicted in Figure 6, is the standard three-index flow model of the PDPTW with the addition of arrival and departure time variables and the duplication of the start and end node for each vehicle. Constraints (3) to (5) are the usual flow constraints, which ensure that each vehicle follows a path from its start node to its end node. Constraint (6) is the request cover constraint, which requires every (pickup) request to be visited.

$$\sum_{j:(s_v,j)\in\mathcal{A}_v} \text{veh}_{v,s_v,j} = 1 \qquad \forall v \in \mathcal{V}, \quad (3)$$

$$\sum_{h:(h,i)\in\mathcal{A}_v} \text{veh}_{v,h,i} = \sum_{j:(i,j)\in\mathcal{A}_v} \text{veh}_{v,i,j} \qquad \forall v \in \mathcal{V}, i \in \mathcal{R}, \quad (4)$$

$$\sum_{h:(h,e_v)\in\mathcal{A}_v} \text{veh}_{v,h,e_v} = 1 \qquad \forall v \in \mathcal{V}, \quad (5)$$

$$\sum_{v\in\mathcal{V}} \sum_{h:(h,i)\in\mathcal{A}_v} \text{veh}_{v,h,i} = 1 \qquad \forall i \in \mathcal{P}, \quad (6)$$

$$\sum_{h:(h,i)\in\mathcal{A}_v} \text{veh}_{v,h,i} = \sum_{h:(h,P+i)\in\mathcal{A}_v} \text{veh}_{v,h,P+i} \qquad \forall v \in \mathcal{V}, i \in \mathcal{P}, \quad (7)$$

$$\text{dep}_{v,i} + d_{i,P+i} \le \text{arr}_{v,P+i} \qquad \forall v \in \mathcal{V}, i \in \mathcal{P}, \quad (8)$$

$$\text{arr}_{v,i} \le \text{serv}_{v,i} \qquad \forall v \in \mathcal{V}, i \in \mathcal{R}, \quad (9)$$

$$\text{serv}_{v,i} + t_i \le \text{dep}_{v,i} \qquad \forall v \in \mathcal{V}, i \in \mathcal{R}, \quad (10)$$

$$\text{arr}_{v,i} = \text{serv}_{v,i} = \text{dep}_{v,i} \qquad \forall v \in \mathcal{V}, i \in \mathcal{S} \cup \mathcal{E}, \quad (11)$$

$$\text{dep}_{v,i} + d_{i,j} - \text{arr}_{v,j} \le M_1 (1 - \text{veh}_{v,i,j}) \qquad \forall v \in \mathcal{V}, (i,j) \in \mathcal{A}_v, \quad (12)$$

$$\text{arr}_{v,j} - \text{dep}_{v,i} - d_{i,j} \le M_2 (1 - \text{veh}_{v,i,j}) \qquad \forall v \in \mathcal{V}, (i,j) \in \mathcal{A}_v, \quad (13)$$

$$\text{load}_{v,i} = 0 \qquad \forall v \in \mathcal{V}, i \in \mathcal{S} \cup \mathcal{E}, \quad (14)$$

$$q_i \le \text{load}_{v,i} \le Q \qquad \forall v \in \mathcal{V}, i \in \mathcal{P}, \quad (15)$$

$$0 \le \text{load}_{v,i} \le Q + q_i \qquad \forall v \in \mathcal{V}, i \in \mathcal{D}, \quad (16)$$

$$\text{load}_{v,i} + q_j - \text{load}_{v,j} \le M_3 (1 - \text{veh}_{v,i,j}) \qquad \forall v \in \mathcal{V}, (i,j) \in \mathcal{A}_v, \quad (17)$$

$$\sum_{h:(h,i)\in\mathcal{A}_v} \text{veh}_{v,h,i} \le 1 \qquad \forall v \in \mathcal{V}, i \in \mathcal{R}, \quad (18)$$

$$\text{veh}_{v,s_v,e_v} \le \text{veh}_{v+1,s_{v+1},e_{v+1}} \qquad \forall v \in \{1,\ldots,V-1\}. \quad (19)$$

Figure 6 The vehicle component of the mixed integer programming model.

Constraints (7) and (8) are the pickup-delivery constraints, which ensure that delivery requests are serviced by the same vehicle that serviced their associated pickup request and are serviced after the associated pickup request. Constraints (9) and (10) order the arrival, service and departure times at each request. Constraint (11) constrains each start node and end node to one common arrival/service/departure time. Constraints (12) and (13) are the travel time constraints, which linearize the constraint

$$\text{veh}_{v,i,j} = 1 \rightarrow \text{dep}_{v,i} + d_{i,j} = \text{arr}_{v,j} \qquad \forall v \in \mathcal{V}, (i,j) \in \mathcal{A}_v.$$

$$\sum_{j:(s_{\text{crew}},j)\in\mathcal{A}} \text{crew}_{c,s_{\text{crew}},j} = 1 \qquad \forall c \in \mathcal{C}, \quad (20)$$

$$\sum_{h:(h,i)\in\mathcal{A}} \text{crew}_{c,h,i} = \sum_{j:(i,j)\in\mathcal{A}} \text{crew}_{c,i,j} \qquad \forall c \in \mathcal{C}, i \in \mathcal{N}, \quad (21)$$

$$\sum_{h:(h,e_{\text{crew}})\in\mathcal{A}} \text{crew}_{c,h,e_{\text{crew}}} = 1 \qquad \forall c \in \mathcal{C}, \quad (22)$$

$$\text{crew}_{c,i,j} \leq \sum_{v\in\mathcal{V}:(i,j)\in\mathcal{A}_v} \text{veh}_{v,i,j} \qquad \forall c \in \mathcal{C}, (i,j) \in \mathcal{A}, l_i \neq l_j, \quad (23)$$

$$\sum_{v\in\mathcal{V}:(i,j)\in\mathcal{A}_v} \text{veh}_{v,i,j} = \sum_{c\in\mathcal{C}} \text{driver}_{c,i,j} \qquad \forall (i,j) \in \mathcal{A}, l_i \neq l_j, \quad (24)$$

$$\text{driver}_{c,i,j} \leq \text{crew}_{c,i,j} \qquad \forall c \in \mathcal{C}, (i,j) \in \mathcal{A}, l_i \neq l_j, \quad (25)$$

$$\text{crewTime}_{c,i} + d_{i,j} - \text{crewTime}_{c,j} \leq M_4 (1 - \text{crew}_{c,i,j}) \qquad \forall c \in \mathcal{C}, (i,j) \in \mathcal{A}, \quad (26)$$

$$\text{arr}_{v,i} \leq \text{crewTime}_{c,i} \leq \text{dep}_{v,i} \qquad \forall v \in \mathcal{V}, c \in \mathcal{C}, i \in \mathcal{N}, \quad (27)$$

$$\text{driveStart}_c - \text{crewTime}_{c,i} \leq M_5 \left(1 - \sum_{j:(i,j)\in\mathcal{A}} \text{driver}_{c,i,j}\right) \qquad \forall c \in \mathcal{C}, i \in \mathcal{R} \cup \mathcal{S}, \quad (28)$$

$$\text{crewTime}_{c,i} - \text{driveEnd}_c \leq M_6 \left(1 - \sum_{h:(h,i)\in\mathcal{A}} \text{driver}_{c,h,i}\right) \qquad \forall c \in \mathcal{C}, i \in \mathcal{R} \cup \mathcal{E}, \quad (29)$$

$$\text{driveDur}_c = \text{driveEnd}_c - \text{driveStart}_c \qquad \forall c \in \mathcal{C}, \quad (30)$$

$$\sum_{h:(h,i)\in\mathcal{A}} \text{crew}_{c,h,i} \leq 1 \qquad \forall c \in \mathcal{C}, i \in \mathcal{N}, \quad (31)$$

$$\text{crew}_{c,s_{\text{crew}},e_{\text{crew}}} \leq \text{crew}_{c+1,s_{\text{crew}},e_{\text{crew}}} \qquad \forall c \in \{1,\ldots,C-1\}, \quad (32)$$

$$\text{crew}_{c,i,j} + \text{crew}_{c,j,k} \leq 1 \qquad \forall c \in \mathcal{C}, (i,j,k) : (i,j) \in \mathcal{A}, (j,k) \in \mathcal{A}, l_i = l_j = l_k, \quad (33)$$

$$\sum_{(i,j)\in\mathcal{A}:i,j\in S} \text{crew}_{c,i,j} \leq |S| - 1 \qquad \forall c \in \mathcal{C}, S \subseteq \mathcal{N}. \quad (34)$$

Figure 7 The crew component of the mixed integer programming model.

Constraints (14) to (16) bound the vehicle load after service of a request. Vehicle loads along a route are accumulated by Constraint (17). Constraint (18) is a redundant constraint that prunes the search space by allowing each vehicle to visit a request at most once. Constraint (19) is a redundant constraint that breaks symmetry between vehicles by forcing a vehicle to be unused if a lower-numbered vehicle is unused. M_1 to M_3 are big-M constants.

The crew component, depicted in Figure 7, overlays the vehicle component with crew constraints to obtain the JVCRSP. It contains routing constraints similar to those in the vehicle component but also includes synchronization constraints to couple the vehicles and crews. Constraints (20) to (22) are the crew flow constraints, which ensure that all crews follow a path beginning at the crew start

node through to the crew end node. Constraints (23) and (24) are space synchronization constraints that require crews to move with a vehicle and vehicles to move with a driver onboard when moving from one location to another. Constraint (25) is another space synchronization constraint that restricts the driver along an arc to be one of the crews that traverses the arc. Constraint (26) is the crew travel time constraint. Constraint (27) is a time synchronization constraint that allows crews to be at a node only while a vehicle is present. Having bounds on the crew time variables, rather than strict equality, is essential to modeling vehicle interchange because it enables crews to both move off a vehicle when it arrives at a location and move on a vehicle when it departs a location. In an equality constraint, crews must either always move off a vehicle at arrival or at departure, which disallow some interchanges. Constraints (28) and (29) determine the start and end of driving of each crew. Constraint (28) is a linearization of the constraint

$$\sum_{j:(i,j)\in\mathcal{A}} \text{driver}_{c,i,j} = 1 \rightarrow \text{driveStart}_c \leq \text{crewTime}_{c,i} \qquad \forall c \in \mathcal{C}, i \in \mathcal{R} \cup \mathcal{S},$$

which imposes that, if a driver departs the node i, the driver must have already started driving before or at the departure time at i. Similarly, Constraint (29) states that if a crew drives to node i, the end of driving of the crew must be later than or at the arrival time of i. Constraint (30) calculates the driving duration of each crew. Constraints (31) and (32) are redundant constraints and are equivalent to Constraints (18) and (19). Constraint (33) prevents crews from visiting subsequences of three or more requests at the same location, as explained in Section 4. M_4 to M_6 are big-M constants.

When a crew travels between two requests i and j within a location, the distance and travel time is zero ($d_{i,j} = 0$), and hence, Constraint (26) fails to perform subtour elimination. There are two possible remedies. The first option replaces $d_{i,j}$ in Constraint (26) with a new crew travel time cost that has a positive value when traveling between two requests within a location. This value can be interpreted as the time required for a crew to switch vehicles. The alternative option is to use the subtour elimination constraints specified by Constraint (34). Our algorithm uses the second approach but adds these constraints lazily in a branch-and-cut scheme.

Objective Function (35) minimizes a weighted sum of the number of vehicles and crews used and the total vehicle and crew travel distances.

$$\begin{aligned} \min w_1 \sum_{v\in\mathcal{V}} \sum_{j\in\mathcal{P}} \text{veh}_{v,s_v,j} \quad + w_2 \sum_{c\in\mathcal{C}} \sum_{j\in\mathcal{S}} \text{crew}_{c,s_{\text{crew}},j} \; + \\ w_3 \sum_{v\in\mathcal{V}} \sum_{(i,j)\in\mathcal{A}_v} d_{i,j}\text{veh}_{v,i,j} + w_4 \sum_{c\in\mathcal{C}} \sum_{(i,j)\in\mathcal{A}} d_{i,j}\text{crew}_{c,i,j}. \end{aligned} \tag{35}$$

6. The Constraint Programming Model

This section discusses the constraint programming formulation of the high-level model of the JVCRSP.

The inputs and decision variables of the constraint programming model are listed in Table 2. The definitions of many of the variables are identical to those in the mixed integer programming model but differ in that some sets are discrete instead of continuous. The problem is defined on a discrete time interval $\mathcal{T} = \{0, \ldots, T\}$, where $T \in \{1, \ldots, \infty\}$ is the time horizon. Let $V \in \{1, \ldots, \infty\}$ be the number of vehicles, and $Q \in \{0, \ldots, \infty\}$ be the load capacity of each vehicle. Let $C \in \{V, \ldots, \infty\}$ be the number of crews, and $\bar{T} \in \mathcal{T}$ be the maximum driving duration. The vehicles and crews are represented by the sets $\mathcal{V} = \{1, \ldots, V\}$ and $\mathcal{C} = \{1, \ldots, C\}$ respectively. The set $\mathcal{C}_0 = \mathcal{C} \cup \{0\}$ extends the set of crews with a dummy value 0 that indicates no crew.

The problem has $P \in \{1, \ldots, \infty\}$ pickup-delivery pairs, giving a total of $R = 2P$ requests. Define $\mathcal{P} = \{1, \ldots, P\}$ and $\mathcal{D} = \{P+1, \ldots, R\}$ as the set of pickups and deliveries respectively, and group them in the set $\mathcal{R} = \mathcal{P} \cup \mathcal{D}$. For every vehicle $v \in \mathcal{V}$, define its unique start and end node as $s(v) = R + v$ and $e(v) = R + V + v$. The start and end nodes are grouped in $\mathcal{S} = \{R+1, \ldots, R+V\}$ and $\mathcal{E} = \{R+V+1, \ldots, R+2V\}$. Let $\mathcal{N} = \mathcal{R} \cup \mathcal{S} \cup \mathcal{E}$ be the set of all nodes, and $\mathcal{N}_0 = \mathcal{N} \cup \{0\}$ be the set of all nodes plus the combined crew start and end depot node 0.

Define $\mathcal{L}$ as the set of locations, including one depot location. For every node $i \in \mathcal{N}$, define $l(i) \in \mathcal{L}$ as its location, $a(i) \in \mathcal{T}$ and $b(i) \in \mathcal{T}$ as the opening and closing of its time window, $t(i) \in \{1, \ldots, \infty\}$ as its service duration, and $q(i) \in \{-Q, \ldots, Q\}$ as its load demand. Let $d(i, j) \in \mathcal{T}$ be the distance and travel time from $i \in \mathcal{N}$ to $j \in \mathcal{N}$.

Let $w_1 \in \{1, \ldots, \infty\}$ and $w_2 \in \{1, \ldots, \infty\}$ be the cost of using one vehicle and one crew, and let $w_3 \in \{1, \ldots, \infty\}$ and $w_4 \in \{1, \ldots, \infty\}$ be the cost of one unit of distance traveled by a vehicle and by a crew.

The primary decision variables are the vehicle successor variables. Successor variables are frequently seen in constraint programming models of vehicle routing problems (e.g., Kilby, Prosser, and Shaw 2000, Rousseau, Gendreau, and Pesant 2002) and serve the same purpose as the flow variables in the mixed integer programming model. For every node $i \in \mathcal{N}$, $\text{succ}(i) \in \mathcal{N}$ denotes the direct successor of i on its route. For example, if $\text{succ}(i) = j$, then the arc (i, j) is used. The successor of a vehicle's end node is the start node of the following vehicle, and the successor of the last vehicle's end node is the start node of the first vehicle. The time and load resources are accumulated along a route and then reset at an end node prior to the start of the next route. Under this modeling, the successor variables describe a Hamiltonian cycle. Figure 8 shows an example of the Hamiltonian cycle formed by four vehicle routes. This modeling was developed by Christofides and Eilon (1969) and subsequently called the *giant tour* representation (e.g., Irnich 2008).

Name	Description
$T \in \{1, \ldots, \infty\}$	Time horizon.
$\mathcal{T} = \{0, \ldots, T\}$	Time interval.
$V \in \{1, \ldots, \infty\}$	Number of vehicles.
$\mathcal{V} = \{1, \ldots, V\}$	Set of vehicles.
$Q \in \{0, \ldots, \infty\}$	Vehicle capacity.
$C \in \{V, \ldots, \infty\}$	Number of crews.
$\mathcal{C} = \{1, \ldots, C\}$	Set of crews.
$\mathcal{C}_0 = \mathcal{C} \cup \{0\}$	Set of crews, including a 0 value indicating no crew.
$\bar{T} \in \mathcal{T}$	Maximum driving duration of a crew.
$P \in \{1, \ldots, \infty\}$	Number of pickup-delivery pairs.
$R = 2P$	Number of requests.
$\mathcal{P} = \{1, \ldots, P\}$	Set of pickup nodes.
$\mathcal{D} = \{P+1, \ldots, R\}$	Set of delivery nodes.
$\mathcal{R} = \mathcal{P} \cup \mathcal{D}$	Set of all requests.
$s(v) = R + v$	Start node of vehicle $v \in \mathcal{V}$.
$e(v) = R + V + v$	End node of vehicle $v \in \mathcal{V}$.
$\mathcal{S} = \{R+1, \ldots, R+V\}$	Set of vehicle start nodes.
$\mathcal{E} = \{R+V+1, \ldots, R+2V\}$	Set of vehicle end nodes.
$\mathcal{N} = \mathcal{R} \cup \mathcal{S} \cup \mathcal{E}$	Set of all requests and vehicle start and end nodes.
$\mathcal{N}_0 = \mathcal{N} \cup \{0\}$	Set of all nodes, including the crew depot node 0.
$\mathcal{L}$	Set of locations, including one depot location.
$l(i) \in \mathcal{L}$	Location of $i \in \mathcal{N}$.
$a(i) \in \mathcal{T}$	Earliest start of service at $i \in \mathcal{N}$.
$b(i) \in \mathcal{T}$	Latest start of service at $i \in \mathcal{N}$.
$t(i) \in \{1, \ldots, \infty\}$	Service duration of $i \in \mathcal{N}$.
$q(i) \in \{-Q, \ldots, Q\}$	Load demand at $i \in \mathcal{N}$.
$d(i,j) \in \mathcal{T}$	Distance and travel time from $i \in \mathcal{N}$ to $j \in \mathcal{N}$.
$w_1 \in \{1, \ldots, \infty\}$	Cost of using one vehicle.
$w_2 \in \{1, \ldots, \infty\}$	Cost of using one crew.
$w_3 \in \{1, \ldots, \infty\}$	Cost of one unit of vehicle distance.
$w_4 \in \{1, \ldots, \infty\}$	Cost of one unit of crew distance.
$\text{succ}(i) \in \mathcal{N}$	Successor of $i \in \mathcal{N}$.
$\text{veh}(i) \in \mathcal{V}$	Vehicle that visits $i \in \mathcal{N}$.
$\text{arr}(i) \in \mathcal{T}$	Arrival time at $i \in \mathcal{N}$.
$\text{serv}(i) \in \{a(i), \ldots, b(i)\}$	Start of service at $i \in \mathcal{N}$.
$\text{dep}(i) \in \mathcal{T}$	Departure time at $i \in \mathcal{N}$.
$\text{load}(i) \in \{0, \ldots, Q\}$	Load of vehicle $\text{veh}(i)$ after servicing $i \in \mathcal{N}$.
$\text{vehUsed}(v) \in \{0, 1\}$	Indicates if vehicle $v \in \mathcal{V}$ is used.
$\text{crewSucc}(c, i) \in \mathcal{N}_0$	Successor of $i \in \mathcal{N}_0$ for crew $c \in \mathcal{C}_0$.
$\text{crewTime}(i) \in \mathcal{T}$	Time when all crews that visit $i \in \mathcal{N}$ is present at i.
$\text{crewUsed}(c) \in \{0, 1\}$	Indicates if crew $c \in \mathcal{C}$ is used.
$\text{crewDist}(c) \in \{0, \ldots, \infty\}$	Distance traveled by crew $c \in \mathcal{C}$.
$\text{driver}(i) \in \mathcal{C}_0$	Driver of vehicle $\text{veh}(i)$ from $i \in \mathcal{R} \cup \mathcal{S}$ to $\text{succ}(i)$, with the value 0 indicating no driver.
$\text{driveStart}(c) \in \mathcal{T}$	Start time of driving for crew $c \in \mathcal{C}_0$.
$\text{driveEnd}(c) \in \mathcal{T}$	End time of driving for crew $c \in \mathcal{C}_0$.
$\text{driveDur}(c) \in \{0, \ldots, \bar{T}\}$	Driving duration of crew $c \in \mathcal{C}$.

Table 2 The data and decision variables of the constraint programming model.

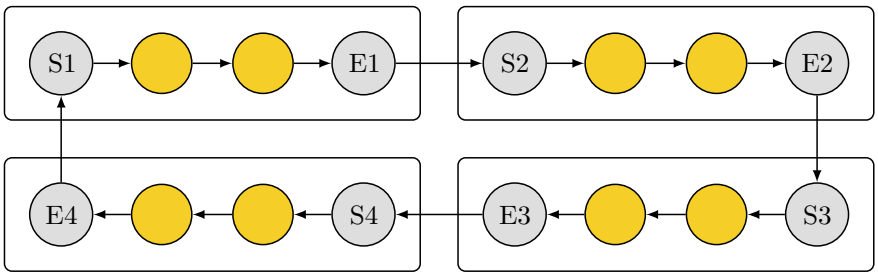


Figure 8 **Example of four vehicle routes as modeled by successor variables. The S nodes and E nodes respectively are the start nodes and end nodes of the four vehicles.**

Since the successor variables are not indexed by vehicle, the model uses the $\text{veh}(i) \in \mathcal{V}$ variable to store the vehicle that visits node $i \in \mathcal{N}$. This modeling, which uses only two vectors, is more succinct than the three-dimensional flow variables found in the mixed integer programming model.

Variables $\text{arr}(i) \in \mathcal{T}$, $\text{serv}(i) \in \{a(i), \ldots, b(i)\}$ and $\text{dep}(i) \in \mathcal{T}$ represent the arrival time, service start time and departure time at node $i \in \mathcal{N}$. The load after servicing node $i \in \mathcal{N}$ is stored in $\text{load}(i) \in \{0, \ldots, Q\}$. Variable $\text{vehUsed}(v) \in \{0, 1\}$ indicates whether vehicle $v \in \mathcal{V}$ is used.

There is a significant difference between vehicles and crews in terms of routing. For vehicles, every request is visited exactly once, which allows the use of a single set of successor variables. In contrast, multiple crews can be on a vehicle when it visits a node, and hence, it is not possible to associate a single crew successor variable with every node. Instead, the model needs a crew successor variable for every crew and every node. However, for any given crew, its successor variables do not need to cover all requests. The crew successor variables are a matrix $\text{crewSucc}(c, i) \in \mathcal{N}_0$ that stores the immediate successor of node $i \in \mathcal{N}_0$ for crew $c \in \mathcal{C}_0$. If crew c does not visit a node i, then $\text{crewSucc}(c, i) = i$. The $\text{crewTime}(i) \in \mathcal{T}$ variable stores a moment when every crew that visits $i \in \mathcal{N}$ is present at i. For every crew $c \in \mathcal{C}$, variable $\text{crewUsed}(c) \in \{0, 1\}$ indicates whether the crew is used, and $\text{crewDist}(c) \in \{0, \ldots, \infty\}$ stores the distance traveled by the crew.

Every node $i \in \mathcal{R} \cup \mathcal{S}$ has an associated $\text{driver}(i) \in \mathcal{C}_0$ variable that either denotes the driver of vehicle $\text{veh}(i)$ if the vehicle travels from i to its successor $\text{succ}(i)$ at a different location, or takes the value 0, indicating that no driver is necessary, if the successor is at the same location. The variables $\text{driveStart}(c) \in \mathcal{T}$ and $\text{driveEnd}(c) \in \mathcal{T}$ store the start and end time of driving of crew $c \in \mathcal{C}_0$, and $\text{driveDur}(c) \in \{0, \ldots, \bar{T}\}$ stores the total driving duration of crew $c \in \mathcal{C}$.

Like the mixed integer programming model, the constraints of the constraint programming model are also divided into a vehicle component and a crew component. The vehicle component, depicted in Figure 9, models a PDPTW. Constraints (36) to (38) restrict the possible values of the successor variables. Constraint (36) states that a vehicle can only move from its start node to any pickup node or its end node. Constraint (37) states that a vehicle can move from a pickup node to any pickup or delivery node, and Constraint (38) states that a vehicle can move from a delivery node

to any pickup, delivery or end node. Constraints (39) and (40) join the end nodes to the start nodes. Using the giant tour modeling, the Circuit global constraint from Constraint (41) performs subtour elimination by imposing a Hamiltonian cycle through the succ$(\cdot)$ variables. Constraints (42) and (43) allow only the associated vehicle to visit the start nodes and end nodes. Constraint (43) is needed to prevent vehicles from visiting the end node of other vehicles, which is permitted by Constraint (38). Constraint (44) tracks vehicles along their routes. Constraints (45) and (46) are the pickup and delivery constraints. Constraints (47) and (48) order the arrival, service and departure times at each request. Constraint (49) restricts each start and end node to one common arrival/service/departure time. Constraint (50) enforces travel times. Constraints (51) to (53) bound the vehicle loads. Constraint (54) is the load constraint. Constraints (55) and (56) state that a vehicle is used if and only if it does not travel from its start node to its end node or if it visits any request. Since they are equivalences, only one of these constraints is necessary but, in practice, stating the two constraints achieves stronger propagation. Constraint (57) breaks vehicle symmetry in a manner similar to Constraint (19).

The crew component, depicted in Figure 10, overlays the vehicle component with crew decisions. Constraints (58) to (62) are the domain restrictions. Constraint (58) requires crews to either leave the crew depot node for a vehicle start node or remain at the crew depot node. Constraints (59) to (61) are similar to Constraints (36) to (38) with the exception that the successor of a node is itself if it is not visited. Constraint (62) states that a crew either moves from a vehicle end node to the crew depot node or the crew does not visit the end node. The Subcircuit global constraint of Constraint (63) enforces connectivity and eliminates subtours (Francis and Stuckey 2014). It differs from the Circuit constraint seen in the vehicle component by allowing some nodes to be excluded from the Hamiltonian cycle.

Constraint (64) states that vehicles have no driver when moving within a location. Constraint (65) requires drivers to move with their vehicles. Constraint (66) requires crews to either move to another node at their current location or move with a vehicle (to a different location). The CrewShortcut global constraint in Constraint (67) removes crew subsequence symmetries within locations, and is detailed in Section 6.1. Constraint (68) allows crews to be at a node only while a vehicle is present. Constraint (69) forces crews to move forward in time only. Constraint (70) states that the driver on the arc $(i, \text{succ}(i))$ starts driving at or before departing i. Similarly, Constraint (71) states that the driver on the arc $(i, \text{succ}(i))$ ends driving at or after arriving at succ(i). Constraint (72) calculates the total driving duration of each crew. Constraints (73) and (74) state that a crew is used if and only if it visits at least one node or if it drives along any arc. Constraint (75) is a symmetry-breaking constraint. Constraint (76) is a global optimization constraint, described in Section 6.2, that bounds crew distances and checks whether crews can return to the depot.

$$\text{succ}(s(v)) \in \mathcal{P} \cup \{e(v)\} \qquad \forall v \in \mathcal{V}, \quad (36)$$

$$\text{succ}(i) \in \mathcal{P} \cup \mathcal{D} \qquad \forall i \in \mathcal{P}, \quad (37)$$

$$\text{succ}(i) \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{E} \qquad \forall i \in \mathcal{D}, \quad (38)$$

$$\text{succ}(e(v)) = s(v+1) \qquad \forall v \in \{1, \ldots, V-1\}, \quad (39)$$

$$\text{succ}(e(V)) = s(1), \quad (40)$$

$$\textsc{Circuit}(\text{succ}(\cdot)), \quad (41)$$

$$\text{veh}(s(v)) = v \qquad \forall v \in \mathcal{V}, \quad (42)$$

$$\text{veh}(e(v)) = v \qquad \forall v \in \mathcal{V}, \quad (43)$$

$$\text{veh}(\text{succ}(i)) = \text{veh}(i) \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \quad (44)$$

$$\text{veh}(i) = \text{veh}(P+i) \qquad \forall i \in \mathcal{P}, \quad (45)$$

$$\text{dep}(i) + d(i, P+i) \leq \text{arr}(P+i) \qquad \forall i \in \mathcal{P}, \quad (46)$$

$$\text{arr}(i) \leq \text{serv}(i) \qquad \forall i \in \mathcal{R}, \quad (47)$$

$$\text{serv}(i) + t(i) \leq \text{dep}(i) \qquad \forall i \in \mathcal{R}, \quad (48)$$

$$\text{arr}(i) = \text{serv}(i) = \text{dep}(i) \qquad \forall i \in \mathcal{S} \cup \mathcal{E}, \quad (49)$$

$$\text{dep}(i) + d(i, \text{succ}(i)) = \text{arr}(\text{succ}(i)) \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \quad (50)$$

$$\text{load}(i) = 0 \qquad \forall i \in \mathcal{S} \cup \mathcal{E}, \quad (51)$$

$$q(i) \leq \text{load}(i) \leq Q \qquad \forall i \in \mathcal{P}, \quad (52)$$

$$0 \leq \text{load}(i) \leq Q + q(i) \qquad \forall i \in \mathcal{D}, \quad (53)$$

$$\text{load}(i) + q(\text{succ}(i)) = \text{load}(\text{succ}(i)) \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \quad (54)$$

$$\text{vehUsed}(v) \leftrightarrow \text{succ}(s(v)) \neq e(v) \qquad \forall v \in \mathcal{V}, \quad (55)$$

$$\text{vehUsed}(v) \leftrightarrow \bigvee_{i \in \mathcal{R}} \text{veh}(i) = v \qquad \forall v \in \mathcal{V}, \quad (56)$$

$$\text{vehUsed}(v) \geq \text{vehUsed}(v+1) \qquad \forall v \in \{1, \ldots, V-1\}. \quad (57)$$

Figure 9 The vehicle component of the constraint programming model.

Objective Function (77) minimizes a weighted sum of the vehicle and crew counts and the total vehicle and crew travel distances.

$$\min w_1 \sum_{v \in \mathcal{V}} \text{vehUsed}(v) + w_2 \sum_{c \in \mathcal{C}} \text{crewUsed}(c) + w_3 \sum_{i \in \mathcal{R} \cup \mathcal{S}} d(i, \text{succ}(i)) + w_4 \sum_{c \in \mathcal{C}} \text{crewDist}(c). \quad (77)$$

$$\text{crewSucc}(c,0) \in \mathcal{S} \cup \{0\} \qquad \forall c \in \mathcal{C}, \quad (58)$$

$$\text{crewSucc}(c,i) \in \mathcal{P} \cup \{i\} \qquad \forall c \in \mathcal{C}, i \in \mathcal{S}, \quad (59)$$

$$\text{crewSucc}(c,i) \in \mathcal{P} \cup \mathcal{D} \qquad \forall c \in \mathcal{C}, i \in \mathcal{P}, \quad (60)$$

$$\text{crewSucc}(c,i) \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{E} \qquad \forall c \in \mathcal{C}, i \in \mathcal{D}, \quad (61)$$

$$\text{crewSucc}(c,i) \in \{0,i\} \qquad \forall c \in \mathcal{C}, i \in \mathcal{E}, \quad (62)$$

$$\textsc{Subcircuit}(\text{crewSucc}(c,\cdot)) \qquad \forall c \in \mathcal{C}, \quad (63)$$

$$l(\text{succ}(i)) = l(i) \leftrightarrow \text{driver}(i) = 0 \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \quad (64)$$

$$\text{crewSucc}(\text{driver}(i), i) = \text{succ}(i) \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \quad (65)$$

$$l(\text{crewSucc}(c,i)) = l(i) \vee \text{crewSucc}(c,i) = \text{succ}(i) \qquad \forall c \in \mathcal{C}, i \in \mathcal{R} \cup \mathcal{S}, \quad (66)$$

$$\textsc{CrewShortcut}(\text{succ}(\cdot), \text{crewSucc}(\cdot,\cdot), \text{arr}(\cdot), \text{dep}(\cdot), \mathcal{C}, \mathcal{R}, \mathcal{S}, \mathcal{E}, l(\cdot)), \quad (67)$$

$$\text{arr}(i) \le \text{crewTime}(i) \le \text{dep}(i) \qquad \forall i \in \mathcal{N}, \quad (68)$$

$$\text{crewTime}(i) \le \text{crewTime}(\text{crewSucc}(c,i)) \qquad \forall c \in \mathcal{C}, i \in \mathcal{R} \cup \mathcal{S}, \quad (69)$$

$$\text{driveStart}(\text{driver}(i)) \le \text{dep}(i) \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \quad (70)$$

$$\text{driveEnd}(\text{driver}(i)) \ge \text{arr}(\text{succ}(i)) \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \quad (71)$$

$$\text{driveDur}(c) = \text{driveEnd}(c) - \text{driveStart}(c) \qquad \forall c \in \mathcal{C}, \quad (72)$$

$$\text{crewUsed}(c) \leftrightarrow \bigvee_{i \in \mathcal{N}_0} \text{crewSucc}(c,i) \neq i \qquad \forall c \in \mathcal{C}, \quad (73)$$

$$\text{crewUsed}(c) \leftrightarrow \bigvee_{i \in \mathcal{R} \cup \mathcal{S}} \text{driver}(i) = c \qquad \forall c \in \mathcal{C}, \quad (74)$$

$$\text{crewUsed}(c) \ge \text{crewUsed}(c+1) \qquad \forall c \in \{1, \ldots, C-1\}, \quad (75)$$

$$\textsc{CrewBound}(\text{crewDist}(c), \text{crewSucc}(c,\cdot), \text{crewTime}(\cdot), \mathcal{R}, \mathcal{S}, \mathcal{E}, d(\cdot,\cdot)) \qquad \forall c \in \mathcal{C}. \quad (76)$$

Figure 10 The crew component of the constraint programming model.

6.1. Breaking Crew Subpath Symmetries within Locations

The CrewShortcut global constraint of Constraint (67) removes symmetric subsequences of requests within locations, which are previously explained in Section 4. Define $\text{pred}(i) \in \mathcal{N}$ as the vehicle predecessor of the node $i \in \mathcal{N}$, and $\text{crewPred}(c,i) \in \mathcal{N}_0$ as the predecessor of $i \in \mathcal{N}_0$ for crew $c \in \mathcal{C}_0$. CrewShortcut implements the following two propagation rules:

$$l(\text{pred}(i)) = l(i) = l(\text{succ}(i)) \leftrightarrow \bigwedge_{c \in \mathcal{C}} \text{crewSucc}(c,i) = i \qquad \forall i \in \mathcal{R}, \quad (78)$$

$$l(i) = l(\text{crewSucc}(c,i)) \wedge \text{crewSucc}(c,i) \neq i \rightarrow$$

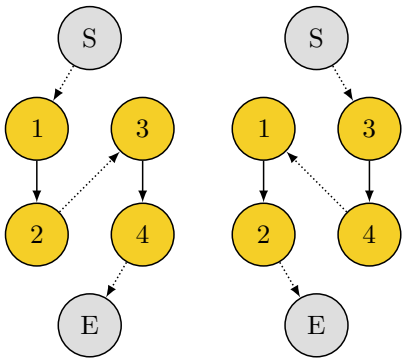


Figure 11 **Example of a partial crew route obtained at an early stage of the search. The S and E nodes are the start node and end node respectively. Solid arrows represent crew assignments and dotted arrows represent possible crew assignments.**

$$l(i) \neq l(\text{crewSucc}(c, \text{crewSucc}(c, i))) \wedge l(i) \neq l(\text{crewPred}(c, i)) \qquad \forall c \in \mathcal{C}, i \in \mathcal{R}. \quad (79)$$

Rule (78) states that if a vehicle visits a sequence of three requests at the same location, then crews cannot visit the second request of the sequence. Rule (79) states that if a crew visits a node i and then another node at the same location, then the crew cannot visit a third node at the same location and the crew must reach i from a different location.

6.2. Feasibility and Bounding of Crew Routes

The constraint programming model presented so far has fewer variables than the mixed integer programming model and prunes infeasible solutions effectively. However, removing infeasible arcs by pruning values from the domains of successor variables may have little or no impact on the lower bound of the objective value. This limitation of constraint programming can be addressed using global optimization constraints (Focacci, Lodi, and Milano 1999, 2000, 2002, 2004). A global optimization constraint wraps a relaxation within a global constraint, giving a constraint programming model tighter lower bounds. This section presents a global optimization constraint that checks the feasibility of the crew partial routes and computes lower bounds to the crew objectives.

The search procedure of the constraint programming model branches on the vehicle successor variables before the crew successor variables. This means that, while searching for vehicle routes, the domains of the crew successor variables are large, leading to weak lower bounds to the crew distance terms in the objective function. For example, Figure 11 shows a partial crew route found in an early stage of the search when the focus is on vehicle routes. The crew is allocated to traverse the arcs $(1, 2)$ and $(3, 4)$, but it is not yet known if a path from the start node S to the end node E via these two arcs exists, and how long this path is if it exists.

A global optimization constraint is used to determine whether each crew has a feasible route and to compute lower bounds to the crew distance. Such a constraint needs to find, for every crew, a

$$\min \sum_{(i,j)\in\mathcal{B}} d(i,j)x_{i,j} \tag{80}$$

subject to

$$\min(\text{cDist}) \leq \sum_{(i,j)\in\mathcal{B}} d(i,j)x_{i,j} \leq \max(\text{cDist}), \tag{81}$$

$$\sum_{(i,j)\in\mathcal{B}:i\in\mathcal{S}} x_{i,j} = 1, \tag{82}$$

$$\sum_{h:(h,i)\in\mathcal{B}} x_{h,i} = \sum_{j:(i,j)\in\mathcal{B}} x_{i,j} \qquad \forall i \in \mathcal{R}, \tag{83}$$

$$\sum_{(h,i)\in\mathcal{B}:i\in\mathcal{E}} x_{h,i} = 1, \tag{84}$$

$$\sum_{j:(i,j)\in\mathcal{B}} x_{i,j} \leq 1 \qquad \forall i \in \mathcal{R}\cup\mathcal{S}, \tag{85}$$

$$t_i + d(i,j) - t_j \leq M(1-x_{i,j}) \qquad \forall (i,j) \in \mathcal{B}, \tag{86}$$

$$x_{i,j} \in [0,1] \qquad \forall (i,j) \in \mathcal{B}, \tag{87}$$

$$t_i \in [\min(\text{cTime}(i)), \max(\text{cTime}(i))] \qquad \forall i \in \mathcal{N}. \tag{88}$$

Figure 12 The linear relaxation of the shortest path problem in the CrewBound optimization constraint.

shortest path from a start node to an end node that includes all arcs known to be traversed by the crew. Since this problem is NP-hard (Laporte, Mercure, and Norbert 1984, Ibaraki 1973, Dreyfus 1969, Volgenant and Jonker 1987), the CrewBound constraint, presented below, uses its linear relaxation.

Whenever a crewSucc$(c,\cdot)$ variable is fixed, the CrewBound constraint solves the linear program defined in Figure 12 for the crew $c \in \mathcal{C}$. The inputs to this linear program include four sets extracted from the current domains of the variables in the main constraint programming model. Let the $D(\cdot)$ function denote the current domain of a variable, then the four input sets are

- cSucc$(i) = D(\text{crewSucc}(c,i))$ for $i \in \mathcal{R}\cup\mathcal{S}$,
- cTime$(i) = D(\text{crewTime}(i))$ for $i \in \mathcal{N}$,
- cDist $= D(\text{crewDist}(c))$, and
- $\mathcal{B} = \{(i,j) | i \in \mathcal{R}\cup\mathcal{S}, j \in \text{cSucc}(i), i \neq j\}$, which represents the current set of arcs that can be traversed by the crew.

The $x_{i,j}$ variable indicates whether the crew traverses arc $(i,j) \in \mathcal{B}$, and the t_i variable stores the arrival time to node i.

Objective Function (80) minimizes the total distance. Constraint (81) bounds the objective value using information from the current domains of the variables in the main constraint programming

Algorithm 1: Sketch of the procedure that assigns vehicle routes.

```
1 for all v ∈ 𝒱
2     i ← s(v)
3     while i ≠ e(v)
4         try all j ∈ D(succ(i)) ordered by min(l(i) ≠ l(j), min(D(ser(j))))
5             succ(i) ← j
6         i ← succ(i)
```

model. Constraints (82) to (84) are the flow constraints, which ensure the existence of a path from a start node to an end node. Constraint (85) is a redundant constraint that strengthens the linear program. Constraint (86) is the time-based subtour elimination constraint, where M is a big-M constant. When $d(i,j)=0$, this constraint fails to perform subtour elimination; however, the linear program remains a valid relaxation of the shortest path problem. Constraints (87) and (88) restrict the domains of the $x_{i,j}$ and t_i variables respectively.

The CrewBound constraint performs three tasks. First, it checks feasibility, i.e., the existence of a route for the crew that satisfies all constraints. Second, it constrains the lower bound of crewDist(c) to be greater than the objective value of the linear program. Third, it prunes the search space using the reduced costs at optimality. For every arc $(i,j) \in \mathcal{B}$, it prunes j from the domain of crewSucc(c,i) if

$$\min(\text{cDist}) + \bar{x}_{i,j} > \max(\text{cDist}),$$

where $\bar{x}_{i,j}$ is the reduced cost of $x_{i,j}$.

6.3. The Search Procedures

The search procedure first assigns vehicle routes and then crew routes. Once all routes are assigned, it assigns values to the time variables.

Vehicle routes are assigned according to Algorithm 1. The procedure considers each vehicle in turn (Line 1) and labels the successor variables from its start node (Line 2) to its end node (Line 3). To choose successors, the algorithm assigns nodes to successor variables using a heuristic that lexicographically prefers nodes at the same location and then nodes with earlier service start times (Line 4). Line 5 makes the assignment, and Line 6 advances to the chosen successor node. The **try all** instruction defines a branching decision.

Crew routes assigned using Algorithm 2. The intuition behind the algorithm is to start with a node i that has no assigned driver, assign a driver c to this node, and then complete the route of this crew. In assigning the route of crew c, preference is given to assigning the crew to drive as much as possible. More precisely, the search procedure begins by ordering all nodes $i \in \mathcal{R} \cup \mathcal{S}$ that have an unassigned driver(i) variable by earliest departure time (Line 2). The search procedure

Algorithm 2: Sketch of the procedure that assigns drivers and crew routes.

```
1  initialize list startDrive of tuples
2  for all i ∈ R ∪ S such that |D(driver(i))| > 1 ordered by min(D(dep(i)))
3      c ← min(D(driver(i)))
4      driver(i) ← c
5      startDrive.append(c, i)
6      while i ∉ E
7          try all j ∈ D(crewSucc(c, i)) ordered by max(c ∈ D(driver(i)) ∧ j ∈ D(succ(i)), rand())
8              crewSucc(c, i) ← j
9              try driver(i) ← c
10         i ← crewSucc(c, i)
11     for all i ∈ R ∪ S such that c ∈ D(driver(i)) ∧ |D(driver(i))| > 1
12         driver(i) ≠ c
13 for all (c, i) ∈ startDrive
14     while i ∉ S
15         try all h ∈ N such that i ∈ D(crewSucc(c, h))
16             crewSucc(c, h) ← i
17         i ← h
```

then selects the first of these nodes and assigns it a driver (Line 4). Only a single driver (here, the one with smallest index) must be considered at this point because all crews are identical (Line 3). For this driving crew, the search procedure constructs a path covering the vehicle routes from the node to the depot (Line 6). It simultaneously labels the crew successor variables from the node to any vehicle end node (Line 8) and the driver variables until the crew exceeds its maximum driving duration (Line 9). Note that the **try** instruction in Line 9 tries to assign crew c as the driver at node i. If this is not possible, the instruction has no effect. The value selection heuristic for crew successor variables favors the node that is visited next by the current vehicle of the crew, provided that the crew can drive from this node; otherwise, the successor is chosen randomly. Once the crew reaches an end node, it is disallowed from driving at any other node (Lines 11 and 12). This process is repeated until all driver$(\cdot)$ variables are fixed. The search procedure then completes the crew routes by labeling the crew successor variables from the first drive node of each crew back to any vehicle start node (Lines 13 to 17).

Algorithm 3 shows the procedure that assigns vehicle schedules and crew schedules. This procedure is best explained using Figure 2. The search procedure selects a vehicle route and divides it into segments consisting of requests at the same location. These segments correspond to requests 1 to 3 and requests 4 to 6 in the example. The search procedure then labels the departure time variable at the exit request in each segment (i.e., requests 3 and 6). It only needs to branch on the departure time variables at these requests because the arrival time at the entry requests (i.e., requests 1 and

Algorithm 3: Sketch of the procedure that assigns vehicle and crew schedules.

```
1  for all v ∈ 𝒱 ordered by min(D(dep(s(v))))
2      i ← s(v)
3      while i ∉ ℰ
4          while l(i) = l(succ(i))
5              dep(i) ← min(D(dep(i)))
6              ser(i) ← min(D(ser(i)))
7              i ← succ(i)
8          try all t ∈ D(dep(i))
9              dep(i) ← t
10         ser(i) ← min(D(ser(i)))
11         i ← succ(i)
12 for all c ∈ 𝒞
13     i ← crewSucc(c, 0)
14     while i ≠ 0
15         crewTime(i) ← min(D(crewTime(i)))
16         i ← crewSucc(c, i)
```

4) are specified by the travel time constraints, and all three time variables at the intermediate nodes (i.e., requests 2 and 5) are bounded by the arrival time at entry requests and the departure time at the exit requests. Once the arrival and departure time of each segment is known, the search procedure simply fixes the time variables at intermediate nodes to their earliest possible. This is always feasible since the travel time and service time constraints already bound the time variables at intermediate requests during the construction of the routes. Crew schedules are simply fixed to their earliest possible since they are tied to the vehicle schedules.

Note that branching on the departure times is essential because of the constraint that limits the maximum driving duration. This constraint is a simple subtraction and propagates extremely weakly unless its variables are fixed.

7. The Large Neighborhood Search

A large neighborhood search is also applied to the mixed integer programming and constraint programming models. Large neighborhood search aims to iteratively find a sequence of improving solutions by destroying parts of a solution and reconstructing it using an underlying solution method. In particular, large neighborhood search uses a neighborhood to fix a number of variables to their values in the incumbent solution and then calls the underlying solver to determine values for the remaining relaxed variables. The solver is given a limited run time since large neighborhood search is an incomplete method and no proof of optimality is available. The large neighborhood search

is applied to both the mixed integer programming and constraint programming models with the following four neighborhoods:

- *Vehicle Route Neighborhood:* This neighborhood fixes a number of vehicle routes and destroys all other vehicle routes and all existing crew routes. This neighborhood destroys large portions of an existing solution, and hence, has the potential to explore diverse parts of the search space. However, the neighborhood is difficult to explore exhaustively because of its size. Because of this, this neighborhood performs better early in optimization and has difficulty improving the solutions near optimality.
- *Request Neighborhood:* This neighborhood relaxes several pickup-delivery pairs and all crew routes. The relaxed requests are then inserted into existing routes. This neighborhood complements the vehicle route neighborhood since it attempts to obtain incremental improvement to an existing solution by destroying small portions of the solution. This neighborhood performs better than the vehicle routing neighborhood later in the solution process because it is significantly smaller and can be explored more exhaustively.
- *Crew Route Neighborhood:* This neighborhood fixes all vehicle routes and relaxes a number of crew routes. It aims to improve the crew objectives using the same reasoning as for the Vehicle Route Neighborhood.
- *Crew Passenger Neighborhood:* This neighborhood fixes all vehicle routes and the driving segments of all crews, and relaxes the passenger segments before and after the driving segment of each crew. This neighborhood attempts to obtain minor improvement to an existing solution by optimizing the pre-driving and post-driving passenger segments of crews. These two segments are only loosely coupled to the vehicles, whereas the driving segment of each crew is tightly coupled to the vehicles. This means that there is more opportunity to optimize these two segments.

8. Experimental Results

This section describes the experiments and analyzes the results. The full tables are given in the online appendix.

8.1. The Instances

The instances are generated to capture the essence of applications in humanitarian and military logistics. These problems typically feature fewer locations than traditional vehicle routing applications but comprise multiple requests at each location. First, five sets of seven locations and five sets of eleven locations are generated on a 50×50 Euclidean grid. Next, 5, 10, 20, 30, 40 and 50 pickup-delivery pairs are generated and assigned to the locations. The instances are then duplicated using three different cost functions:

- $w_1 = 1000$, $w_2 = 1000$, $w_3 = 1$, $w_4 = 1$,

- $w_1 = 1000$, $w_2 = 5000$, $w_3 = 1$, $w_4 = 1$, and
- $w_1 = 5000$, $w_2 = 1000$, $w_3 = 1$, $w_4 = 1$.

In total, there are $10 \times 6 \times 3 = 180$ instances. Service durations vary between 1 and 20, and load demands vary between 1 and 15. The time windows of requests are randomly chosen.

8.2. The Methods

Both the mixed integer programming and constraint programming models are solved in two stages. First, the vehicle component is solved to produce an initial set of feasible vehicle routes. Then the full model, consisting of the vehicle and crew components, is started using these vehicle routes. If the vehicle routing stage is unable to find a feasible solution, the main model is started with a solution consisting of one pickup-delivery pair per vehicle.

Additionally, the full model is solved with (1) the vehicle route variables fixed according to the initial solution, and (2) the vehicle route variables fixed according to the initial solution and the vehicle time variables fixed to their earliest possible. These two models are respectively named Fixed and Semi-flexible. The original model with both variable vehicle routes and schedules is named Flexible.

All three models are started using the same vehicle routes, enabling any improvement in the objective value to be attributed to the joint vehicle and crew optimization rather than to other causes, such as the branching decisions used to obtain an initial solution. Because the vehicle routes in Fixed and Semi-flexible are fixed to the initial vehicle routing solution but Semi-flexible also searches over vehicle schedules, any improvement in crew routing objectives seen in Semi-flexible can be attributed to better vehicle scheduling. Similarly, the impact of rerouting vehicles according to the crew objectives can be examined by comparing Flexible against Semi-flexible since Semi-flexible has fixed vehicle routes but Flexible does not.

The mixed integer programming models and constraint programming models are respectively implemented in Gurobi and Objective-CP (Van Hentenryck and Michel 2013). All six models are solved using branch-and-bound and the large neighborhood search procedures detailed in Section 7. The two search techniques are respectively named BB and LNS.

The following parameters are used for the large neighborhood search:

- The four neighborhoods are selected with equal probability. For the first stage of the sequential methods, there is an equal probability of selecting any of the two vehicle neighborhoods.
- For the vehicle route neighborhood, between two and five vehicle routes are destroyed. If the current solution has fewer than the chosen number of vehicle routes to destroy, one route will be retained and the rest destroyed.
- For the request neighborhood, between two and seven pickup-delivery pairs are destroyed but at least one is retained.

	MIP-BB	CP-BB	MIP-LNS	CP-LNS
Fixed	118	180	151	180
Semi-flexible	121	180	144	180
Flexible	94	180	143	180

Table 3 Number of instances with feasible solutions for each method.

- For the crew route neighborhood, between two and seven crew routes are destroyed but at least one is retained.
- For the crew passenger neighborhood, the passenger segments of all crews are destroyed but the driving segments are retained.
- For MIP-LNS, Gurobi is called to determine values for the relaxed variables until it reaches a time limit of one minute.
- For CP-LNS, Objective-CP is used to fix values for the relaxed variables until it reaches a failure limit of 800.

The vehicle routing initialization is solved for one hour and the main model is solved for three hours on an Intel Xeon E5-2660 V3 CPU at 2.6 GHz.

8.3. Feasible Solutions

Table 3 shows the number of instances for which the methods find at least one feasible solution. The CP-based methods find feasible solutions to all 180 instances but the MIP-based methods face significant difficulties in even finding feasible solutions.

8.4. The Impacts of Rescheduling Vehicles

Table 4 compares the four Semi-flexible models against the Fixed models. The advantages seen in Semi-flexible in comparison to Fixed can be attributed to the ability to co-optimize vehicle schedules and crews. The results show that the four Semi-flexible approaches perform substantially better than their Fixed counterparts, achieving overall cost reductons of between 4.40% and 12.82%. The difference is most apparent in MIP-LNS, which performs up to 50.32% better than Fixed MIP-LNS. However, it also performs up to 129.03% worse. Semi-flexible MIP-BB performs up to 1.22% worse than Fixed MIP-BB, while Semi-flexible CP-BB and CP-LNS perform no worse than their Fixed variants.

A comparison of the four Semi-flexible methods against the a posteriori best Fixed method of each instance is also presented in the table. The results indicate that Semi-flexible MIP-BB (98.59%), CP-BB (12.19%) and MIP-LNS (28.69%) all perform worse than the best Fixed method on average. Only Semi-flexible CP-LNS improves on the best Fixed approach on average (9.28%). In fact, CP-LNS only performs up to 0.06% worse than the best Fixed approach on difficult instances, whereas the other three methods perform from five to over nine times worse. However, all four

		MIP-BB	CP-BB	MIP-LNS	CP-LNS
Fixed	Average	−12.82%	−9.64%	−4.40%	−9.52%
	Minimum	−45.36%	−45.68%	−50.32%	−31.12%
	Maximum	1.22%	0.00%	129.03%	0.00%
	Standard deviation	10.96%	8.11%	22.38%	7.83%
Best Fixed	Average	98.59%	12.19%	28.69%	−9.28%
	Minimum	−31.02%	−31.02%	−31.02%	−31.05%
	Maximum	918.69%	754.83%	560.02%	0.06%
	Standard deviation	173.77%	71.24%	95.37%	7.60%

Table 4 Comparison of the four Semi-flexible methods against their Fixed counterparts and the best Fixed method of each instance.

Semi-flexible approaches can perform up to 31% better. The standard deviations of comparing MIP-BB, CP-BB and MIP-LNS against the best Fixed approach is particularly large, indicating that their performance is highly dependent on the instances. The standard deviation of CP-LNS is much smaller, suggesting that this method is more consistent in its performance.

The excellent behavior of Semi-flexible CP-LNS indicates that there is significant value in jointly optimizing vehicle schedules and crew routes in a two-stage approach consisting of an initial vehicle routing phase and a vehicle scheduling and crew routing and scheduling step.

8.5. The Impacts of Rerouting Vehicles

The effects of rerouting vehicles according to crew objectives can be observed by comparing Flexible to Semi-flexible. Table 5 provides statistics comparing the four Flexible models against the Semi-flexible models. On average, Flexible MIP-BB (0.49%), CP-BB (0.30%) and CP-LNS (1.61%) perform better than their Semi-flexible counterparts. However, allowing variable vehicle routes is detrimental to Flexible MIP-LNS. It displays significant difficulties in finding good feasible solutions, performing 3.67% worse than Semi-flexible MIP-LNS overall. Flexible MIP-BB, CP-BB and MIP-LNS can all perform up to 22.52% better than their Semi-flexible variants, while CP-LNS can reach 30.76% better. Due to the short time limit and the larger search space, all four Flexible methods can perform worse than their Semi-flexible counterparts. In particular, Flexible MIP-LNS can be up to 58.89% worse off than Semi-flexible MIP-LNS.

The trends seen in comparing Flexible against the best Semi-flexible model is similar to comparing Semi-flexible against the best Fixed method in the previous subsection. On average, only Flexible CP-LNS performs better than the best Semi-flexible model (1.45%), and the other three approaches perform significantly worse (24.45% to 103.08%). Even on difficult instances, Flexible CP-LNS only performs up to 9.02% worse than the best Semi-flexible model; the other three models perform from seven to over ten times worse. The standard deviation is again small for CP-LNS, indicating that its performance is much more consistent than the other three methods.

		MIP-BB	CP-BB	MIP-LNS	CP-LNS
Semi-flexible	Average	−0.49%	−0.30%	3.67%	−1.61%
	Minimum	−22.52%	−22.52%	−22.52%	−30.76%
	Maximum	5.73%	1.55%	58.89%	9.02%
	Standard deviation	3.34%	2.14%	10.73%	4.80%
Best Semi-flexible	Average	103.08%	24.45%	48.48%	−1.45%
	Minimum	−22.52%	−22.52%	−22.52%	−30.66%
	Maximum	1054.08%	867.52%	739.96%	9.02%
	Standard deviation	215.79%	83.86%	117.43%	4.72%

Table 5 Comparison of the four Flexible methods against their Semi-flexible counterparts and the best Semi-flexible method of each instance.

It appears that the concepts of rerouting vehicles for crew optimization are only realized by Flexible CP-LNS, which improves upon the best Semi-flexible method with cost reductions of up to 30.66% and 1.45% on average. These results indicate that improvements can be found by rerouting vehicles but are computationally demanding, warranting further investigation on the consequences of vehicle rerouting for crew routing and scheduling.

8.6. The Impacts of Rerouting and Rescheduling Vehicles

The combined effects of both rerouting and rescheduling vehicles can be analyzed in a comparison between the Flexible and Fixed methods. Table 6 shows summary statistics of this comparison. Note that there is some inconsistency with the MIP results in Tables 4 and 5 due to the different instances for which feasible solutions are available. The results demonstrate that Flexible MIP-BB and MIP-LNS are inferior to Semi-flexible MIP-BB and MIP-LNS on average. Flexible MIP-BB and MIP-LNS find fewer feasible solutions and improve less on Fixed MIP-BB and MIP-LNS. Contrastingly, Flexible CP-BB and CP-LNS perform 9.90% and 11.01% better than Fixed CP-BB and CP-LNS, which are more than the 9.64% and 9.52% improvements found by Semi-flexible CP-BB and CP-LNS. MIP-BB, CP-BB and CP-LNS achieve benefits of up to 45.36%, 45.68% and 47.04% at their best, and never perform worse than their Fixed counterparts. Flexible MIP-LNS improves on its own Fixed variant the most (50.32%), probably because Fixed MIP-LNS performed poorly. Flexible MIP-LNS is also the only method not dominating its Fixed counterpart, finding solutions up to 131.06% worse.

A comparison of Flexible against the a posteriori best Fixed results is also almost identical to the previous discussions. Flexible MIP-BB, CP-BB and MIP-LNS average 78.56%, 11.94% and 33.25% worse than the best Fixed result, and can exceed nine times worse. Only Flexible CP-LNS improves on its Fixed variant on average (10.77%) and never performs any worse.

These results indicate that given an appropriate formulation and search technique, it is possible to find improved solutions by rerouting and rescheduling vehicles. Considering that Flexible CP-LNS

		MIP-BB	CP-BB	MIP-LNS	CP-LNS
Fixed	Average	−11.13%	−9.90%	−0.86%	−11.01%
	Minimum	−45.36%	−45.68%	−50.32%	−47.04%
	Maximum	0.00%	0.00%	131.06%	0.00%
	Standard deviation	10.89%	8.49%	26.09%	8.34%
Best Fixed	Average	78.56%	11.94%	33.25%	−10.77%
	Minimum	−36.82%	−36.82%	−36.82%	−46.99%
	Maximum	918.69%	754.83%	560.02%	0.00%
	Standard deviation	182.18%	71.37%	97.38%	8.16%

Table 6 Comparison of the four Flexible methods against their Fixed counterparts and the best Fixed method of each instance.

dominates the best Fixed and improves on Semi-flexible as discussed previously, it is easy to conclude that Flexible CP-LNS is the best method of those evalulated.

8.7. Detailed Analysis

This section presents the main discoveries for each method. All solutions from MIP-BB, CP-BB, MIP-LNS and CP-LNS are reported in the accompanying online appendix.

Analysis of MIP-BB A number of findings related to MIP-BB are listed below:

- Fixed is only able to find feasible solutions to 118 of the 180 instances despite being initialized with feasible vehicle routes. Semi-flexible and Flexible respectively find feasible solutions to 121 and 94 instances. MIP-BB performs particularly poorly on the instances with 40 and 50 pickup-delivery pairs. Fixed, Semi-flexible and Flexible are only able to find feasible solutions to 7, 10 and 9 of the 60 largest instances respectively.

- Semi-flexible does not dominate Fixed; Semi-flexible performs worse on two instances (1.22% and 0.42%). On average, Semi-flexible performs 12.82% better than Fixed on the 106 instances for which they both find feasible solutions. For the cost functions $w_1 = 1000$ and $w_2 = 1000$, $w_1 = 1000$ and $w_2 = 5000$, and $w_1 = 5000$ and $w_2 = 1000$, Semi-flexible respectively averages 14.48%, 17.48% and 6.76% better than Fixed on the 36, 34 and 36 instances with feasible solutions from both models. As discussed previously, these improvements are likely a consequence of allowing variable vehicle schedules.

- There are 82 instances for which Semi-flexible reschedules the vehicles for an improved overall cost and a reduction in the number of crews. For these instances, 7.52 fewer crews are needed on average, resulting in cost savings of 16.55%.

- Flexible dominates Fixed. It finds savings of 11.13% better overall, at least 20% savings on 16 instances, and 45.36% savings on one instance. For the three cost functions, Flexible performs 12.46%, 15.41% and 5.94% better than Fixed on average on the 29, 25 and 28 instances for which both models find feasible solutions.

• There are 25 instances for which Flexible trades crew costs for vehicle costs in order to find overall better solutions. On these instances, overall costs reduced by 12.97%, vehicle costs increased by 4.20% and crew costs reduced by 19.18%. Two of these 25 instances require one more vehicle but one fewer crew, resulting in 0.50% and 10.49% cost savings overall, which comprises of an increase in vehicle costs of 44.93% and 48.01%, and a decrease in crew costs of 13.98% and 14.13%.

• Flexible performs 0.49% better on average than Semi-flexible. There are 12 instances for which Flexible performs worse than Semi-flexible, and 28 instances for which it performs better. For the three cost functions, Flexible averages 0.49%, 0.70% and 0.26% better than Semi-flexible on the 31, 31 and 30 instances for which both approaches find feasible solutions.

• There are 80 instances for which all three variants find feasible solutions. On these instances, Semi-flexible and Flexible perform 10.77% and 11.41% better than Fixed respectively, and Flexible improves upon Semi-flexible by 0.72%. These results suggest that Flexible MIP-BB is superior to Semi-flexible; although it is difficult to definitively argue this case considering that Flexible only finds feasible solutions to 94 instances whereas Semi-flexible finds feasible solutions to 121 instances.

Analysis of CP-BB Several findings focusing on CP-BB are discussed below:

• Fixed, Semi-flexible and Flexible find feasible solutions to all 180 instances.

• CP-BB Fixed, Semi-flexible and Flexible averages 10.81%, 8.72% and 8.62% better than MIP-BB Fixed, Semi-flexible and Flexible on the 80 instances where the three variants of MIP-BB find solutions. CP-BB Fixed performs better than MIP-BB Fixed on 20 instances and worse on 15 instances. CP-BB Semi-flexible performs better than MIP-BB Semi-flexible on 20 instances and worse on 15. CP-BB Flexible performs better than MIP-BB Flexible on 21 instances and worse on 16 instances.

• Semi-flexible performs 9.64% better overall than Fixed. For the three cost functions, it averages 10.39%, 12.96% and 5.59% better. These numbers are not as pronounced as MIP-BB because Fixed CP-BB already performs better than Fixed MIP-BB.

• Semi-flexible dominates Fixed, which shows that CP-BB can improve crew optimization by searching over vehicle schedules. On 144 instances, Semi-flexible finds solutions with 4.47 fewer crews on average and reductions of 12.03% in cost compared to Fixed.

• Flexible dominates Fixed and averages 9.90% better overall, and 10.70%, 13.28% and 5.71% better for the three cost functions.

• Flexible performs almost identically to Semi-flexible, achieving 0.30% better solutions in general. It performs better than Semi-flexible on 28 instances, achieving reductions of 14.20% in cost. It also performs worse on three instances, with increased costs of 0.54% on average.

• There are 24 instances on which Flexible finds improved solutions compared to Fixed by increasing vehicle costs. Increasing vehicle costs by 0.25% allows crew costs to be decreased by

21.20%, achieving an overall reduction of 14.65%. On the same instances, relaxing the vehicle routes allows Flexible to improve upon Semi-flexible by decreasing crew costs by 3.81%, resulting in an overall decrease of 2.26% in costs.

Analysis of MIP-LNS An analysis of MIP-LNS is presented below:

- Fixed, Semi-flexible and Flexible find feasible solutions to 151, 144 and 143 instances respectively; that is, 33, 23 and 49 more than MIP-BB.
- Semi-flexible sees benefits of 4.40% compared to Fixed. It performs worse (23.18%) on five instances and better (11.62%) on 115 instances. For the three cost functions, Semi-flexible finds cost savings of 5.27%, 5.05% and 2.88%.
- Semi-flexible is able to reschedule the vehicles to reduce the total cost and the total number of crews compared to Fixed on 99 instances. On these instances, 2.97 fewer crews are required and 13.11% cost savings are available.
- Flexible improves upon Fixed by 0.86% overall. The cost reductions on 13 instances exceed 20%, and reaches 50.32% on one instance. For the three cost functions, it performs 2.59% better, 1.24% worse and 1.31% better.
- Flexible performs 3.67% worse than Semi-flexible overall. The solutions are 2.77%, 6.29% and 1.83% worse when separated into the three cost functions.
- Flexible reroutes vehicles to increase vehicle costs but decrease crew costs on 36 instances. Vehicle costs are increased by 0.28% on average when compared to Fixed. In return, crew costs are decreased by up to 39.97% but averages 18.20%. On these 36 instances, overall costs are reduced by 13.05%.
- MIP-LNS Fixed, Semi-flexible and Flexible averages 11.06%, 5.94% and 4.84% better than MIP-BB on the 80 instances that all six variants find feasible solutions, and 8.58%, 16.40% and 21.23% worse than CP-BB on the 140 instances with feasible solutions.

Analysis of CP-LNS A number of key results are highlighted below:

- Fixed, Semi-flexible and Flexible find feasible solutions to all 180 instances.
- Semi-flexible dominates Fixed, performing 9.52% better on average, and 10.29%, 12.79% and 5.49% better for the three cost functions.
- Semi-flexible reschedules the vehicles on 141 instances to find lower total costs, which are reduced by 12.13%. On average, 2.98 fewer crews are necessary.
- Flexible dominates Fixed, improving upon Fixed by 11.01% overall, and 11.70%, 14.72% and 6.62% for the three cost functions.
- There are 30 instances for which Flexible performs worse (1.84%) and 98 instances for which it performs better (3.51%) than Semi-flexible. Generally, Flexible improves on Semi-flexible by 1.61%, unlike the other three search methods, for which Flexible performs nearly identically or worse than Semi-flexible.

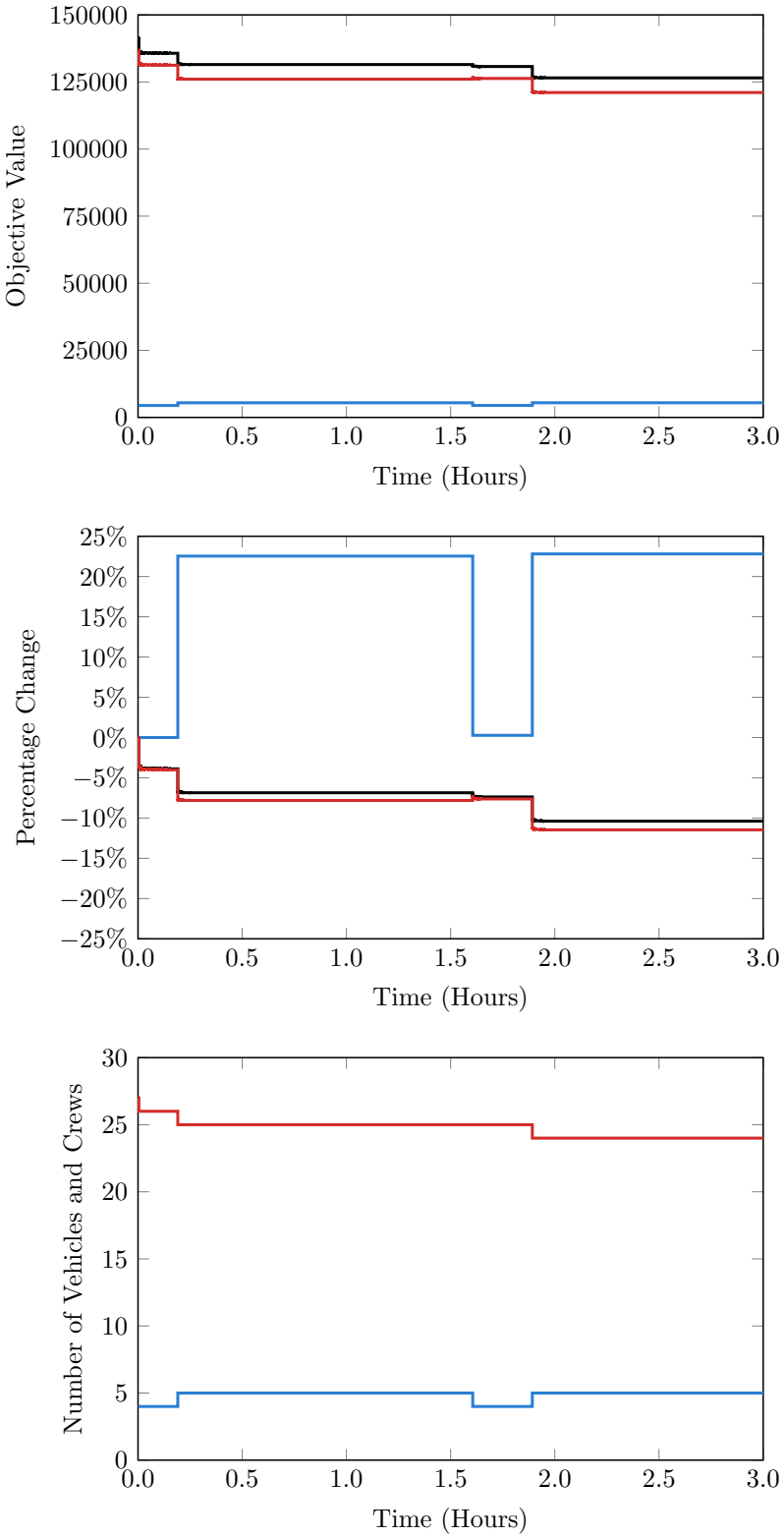


Figure 13 **Plots of the objective value, the percentage change and the number of vehicles and crews over time for Flexible CP-LNS on an instance with $|\mathcal{L}| = 11$, $|\mathcal{P}| = 30$, $w_1 = 1000$ and $w_2 = 5000$. Lines colored blue represent vehicles, red represent crews and black represent the total objective.**

- Flexible often finds solutions in which vehicle costs are higher than those from Fixed. On 75 instances, it increases vehicle costs by 1.48% and decreases crew costs by 17.83%, resulting in savings of 12.82% on average.
- Figure 13 shows three plots related to Flexible CP-LNS on an instance with $|\mathcal{L}| = 11$, $|\mathcal{P}| = 30$, $w_1 = 1000$ and $w_2 = 5000$. Notice that the vehicle costs and crew costs do not monotonically decrease. Furthermore, the plots show that the solver twice finds a solution that uses one additional vehicle to decrease overall costs. The final solutions from Fixed and Semi-flexible require 4 vehicles and 29 and 26 crews respectively. The final solution of Flexible requires 5 vehicles but only 24 crews. This solution is 16.09% better than Fixed and 6.76% better than Semi-flexible. Vehicle costs are increased by 22.81% but crew costs are decreased by 17.27% compared to Fixed and 7.76% compared to Semi-flexible.
- On average, CP-LNS Fixed, Semi-flexible and Flexible respectively perform 12.35%, 11.37% and 13.07% better than MIP-BB, 12.17%, 12.32% and 13.62% better than CP-BB and 11.93%, 13.41% and 17.41% better than MIP-LNS on the instances for which the six variants in each comparison find feasible solutions.

9. Conclusion

This paper presents the Joint Vehicle and Crew Routing and Scheduling Problem, which is motivated by applications arising in humanitarian and military logistics. The problem routes and schedules vehicles and crews to pick up and deliver requests. Because crews are able to interchange vehicles, both vehicle routes and crew routes become highly interdependent in space and time.

This paper develops a high-level model of the problem, which is formulated as a mixed integer programming model and a constraint programming model. The two models overlay crew routing constraints over the Pickup and Delivery Problem with Time Windows. The constraint programming model features a symmetry-breaking global constraint and a global optimization constraint to detect infeasibility and to bound crew objectives. Both the mixed integer programming and constraint programming models are solved using a regular branch-and-bound search and a large neighborhood search. In order to compare the effects of rerouting and rescheduling vehicles on crew optimization, both models are extended with constraints that (1) fix the vehicle routes and (2) fix the vehicle routes and vehicle schedules. Comparing the full model, called Flexible, against one with fixed vehicle routes, called Semi-flexible, allows the impacts of rerouting vehicles for crew optimization to be quantified. Similarly, comparing Semi-flexible to the model with fixed vehicle routes and schedules, named Fixed, allows the benefits of rescheduling vehicles to be observed.

Experimental results indicate that jointly optimizing vehicle and crew routing and scheduling achieves significant benefits, and that the constraint programming model coupled with large

neighborhood search finds the greatest reductions in costs compared to the other approaches. In particular, it produces average improvements of 10.77% compared to the a posteriori best Fixed sequential method and 1.45% compared to the a posteriori best partially sequential Semi-flexible method.

All models and search methods found solutions that trade crew costs for vehicle costs in order to produce an overall improved solution. The ability to use fewer crews at the expense of more vehicles is a key feature of integrated models and is highly difficult, if not impossible, to replicate in sequential methods.

These results highlight the benefits of jointly optimizing vehicles and crew, and in particular, suggest that many of the benefits of jointly optimizing vehicle and crew routing originate in the ability to reschedule vehicles according to crew objectives. There are also benefits in rerouting vehicles for crew optimization but, at this stage, they seem much more computationally demanding.

There are several interesting research avenues going forward. Given the strong performance of the Semi-flexible methods, it would be interesting to study whether vehicle scheduling and crew routing can be solved to optimality using combinations of constraint programming and mathematical programming. Another direction for future research is to evaluate a three-stage optimization process that runs Fixed, Semi-flexible and Flexible in sequence with each stage initialized using the solution from the previous stage. Furthermore, another promising direction for future research is to develop neighborhoods that span vehicle and crew routes and/or schedules because the existing neighborhoods only consider vehicle routes or crew routes independently. Finally, it will also be interesting to compare the existing routing models against state-of-the-art scheduling models based on constraint programming.

References

Barnhart C, Lu F, Shenoi R, 1998 *Integrated Airline Schedule Planning*, chapter 13, 384–403. Operations Research in the Airline Industry (Springer US).

Christofides N, Eilon S, 1969 *An algorithm for the vehicle-dispatching problem. Journal of the Operational Research Society* 20(3):309–318.

Cordeau JF, Stojković G, Soumis F, Desrosiers J, 2001 *Benders decomposition for simultaneous aircraft routing and crew scheduling. Transportation Science* 35(4):375–388.

Drexl M, 2007 *On some generalized routing problems*. Ph.D. thesis, RWTH Aachen University.

Drexl M, 2012 *Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. Transportation Science* 46(3):297–316.

Drexl M, 2013 *Applications of the vehicle routing problem with trailers and transshipments. European Journal of Operational Research* 227(2):275–283.

Drexl M, 2014 *Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. Networks* 63(1):119–133.

Drexl M, Rieck J, Sigl T, Press B, 2013 *Simultaneous vehicle and crew routing and scheduling for partial- and full-load long-distance road transport. BuR - Business Research* 6(2):242–264.

Dreyfus SE, 1969 *An appraisal of some shortest-path algorithms. Operations Research* 17(3):395–412.

Focacci F, Lodi A, Milano M, 1999 *Cost-based domain filtering.* Jaffar J, ed., *Principles and Practice of Constraint Programming – CP'99: 5th International Conference, CP'99, Alexandria, VA, USA, October 11-14, 1999. Proceedings*, 189–203 (Berlin, Heidelberg: Springer Berlin Heidelberg).

Focacci F, Lodi A, Milano M, 2000 *Cutting planes in constraint programming: An hybrid approach.* Dechter R, ed., *Principles and Practice of Constraint Programming – CP 2000*, volume 1894 of *Lecture Notes in Computer Science*, 187–201 (Springer Berlin Heidelberg).

Focacci F, Lodi A, Milano M, 2002 *Optimization-oriented global constraints. Constraints* 7(3-4):351–365.

Focacci F, Lodi A, Milano M, 2004 *Exploiting relaxations in CP.* Milano M, ed., *Constraint and Integer Programming*, volume 27 of *Operations Research/Computer Science Interfaces Series*, 137–167 (Springer US).

Francis KG, Stuckey PJ, 2014 *Explaining circuit propagation. Constraints* 19(1):1–29.

Freling R, Huisman D, Wagelmans AP, 2001 *Applying an integrated approach to vehicle and crew scheduling in practice.* Voß S, Daduna JR, eds., *Computer-Aided Scheduling of Public Transport*, volume 505 of *Lecture Notes in Economics and Mathematical Systems*, 73–90 (Springer Berlin Heidelberg).

Freling R, Huisman D, Wagelmans AP, 2003 *Models and algorithms for integration of vehicle and crew scheduling. Journal of Scheduling* 6(1):63–85.

Freling R, Wagelmans AP, Paixão JMP, 1999 *An overview of models and techniques for integrating vehicle and crew scheduling.* Wilson NH, ed., *Computer-Aided Transit Scheduling*, volume 471 of *Lecture Notes in Economics and Mathematical Systems*, 441–460 (Springer Berlin Heidelberg).

Haase K, Desaulniers G, Desrosiers J, 2001 *Simultaneous vehicle and crew scheduling in urban mass transit systems. Transportation Science* 35(3):286–303.

Hollis B, Forbes M, Douglas B, 2006 *Vehicle routing and crew scheduling for metropolitan mail distribution at Australia Post. European Journal of Operational Research* 173(1):133–150.

Ibaraki T, 1973 *Algorithms for obtaining shortest paths visiting specified nodes. SIAM Review* 15(2):309–317.

Irnich S, 2008 *A unified modeling and solution framework for vehicle routing and local search-based meta-heuristics. INFORMS Journal on Computing* 20(2):270–287.

Kilby P, Prosser P, Shaw P, 2000 *A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints. Constraints* 5(4):389–414.

Kim BI, Koo J, Park J, 2010 *The combined manpower-vehicle routing problem for multi-staged services. Expert Systems with Applications* 37(12):8424–8431.

Lam E, Van Hentenryck P, 2016 *A branch-and-price-and-check model for the vehicle routing problem with location congestion. Constraints* 21(3):394–412.

Lam E, Van Hentenryck P, Kilby P, 2015 *Joint vehicle and crew routing and scheduling.* Pesant G, ed., *Principles and Practice of Constraint Programming: 21st International Conference, CP 2015, Cork, Ireland, August 31 – September 4, 2015, Proceedings*, 654–670 (Springer, Cham).

Laporte G, Mercure H, Norbert Y, 1984 *Optimal tour planning with specified nodes. RAIRO-Operations Research* 18(3):203–210.

Mercier A, Cordeau JF, Soumis F, 2005 *A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. Computers & Operations Research* 32(6):1451 – 1476.

Mercier A, Soumis F, 2007 *An integrated aircraft routing, crew scheduling and flight retiming model. Computers & Operations Research* 34(8):2251–2265.

Mesquita M, Paias A, 2008 *Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. Computers & Operations Research* 35(5):1562–1575.

Rousseau LM, Gendreau M, Pesant G, 2002 *Using constraint-based operators to solve the vehicle routing problem with time windows. Journal of Heuristics* 8(1):43–58.

Van Hentenryck P, Michel L, 2013 *The Objective-CP optimization system.* Schulte C, ed., *Principles and Practice of Constraint Programming: 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, 8–29 (Springer Berlin Heidelberg).

Vigo D, Toth P, 2014 *Vehicle Routing: Problems, Methods, and Applications* (Society for Industrial and Applied Mathematics), second edition.

Volgenant T, Jonker R, 1987 *On some generalizations of the travelling-salesman problem. The Journal of the Operational Research Society* 38(11):1073–1079.