Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-1 (2022) 38–45

Route planning for capacity restricted agents over railway network, without disrupting train schedules

Somnath Buriuly * **Leena Vachhani** ** **Arpita Sinha** ***
Sivapragasam Ravitharan **** **Sunita Chauhan** †

* *Somnath Buriuly is with the Systems and Control Engineering, IIT Bombay, India, and Department of Mechanical and Aerospace, Monash University, Australia; e-mail: somnath.buriuly@monash.edu.*
** *Dr Leena Vachhani is Professor at Systems and Control Engineering, IIT Bombay, India; e-mail: leena.vachhani@iitb.ac.in.*
*** *Dr Arpita Sinha is Professor at Systems and Control Engineering, IIT Bombay, India; e-mail: arpita.sinha@iitb.ac.in.*
**** *Mr Sivapragasam Ravitharan is Director at Monash Institute of Railway Technology, Monash University, Australia; e-mail: ravi.ravitharan@monash.edu.*
† *Prof Sunita Chauhan is Professor at Department of Mechanical and Aerospace, Monash University, Australia; e-mail: sunita.chauhan@monash.edu.*

Abstract: Deploying mobile instrumentation for railway track inspection is a routing and scheduling problem, that can be benefited from plans which don't disrupt regular train schedules. In this work, we avoid disrupting train schedules by modeling unavailability of railway tracks, thereby improving reliability without sacrificing on costs. The approach involves a novel mathematical formulation for *C*apacitated *A*rc *R*outing *P*roblem with *T*emporal restrictions due to arc *U*navailabilities (CARP-TU). In addition, we discuss the challenges in CARP-TU w.r.t sibling problems, and propose a suitable column generation algorithm for improving upper and lower bounds. The proposed algorithm is validated over few benchmark datasets, and then two variants of the proposed algorithm are compared to analyse the bound improvements w.r.t problem size.

Keywords: Capacitated Arc Routing Problem, Temporal, Arc Unavailability, Railway, Robotics, Column generation, Dynamic Programming

1. INTRODUCTION AND LITERATURE SURVEY

Arc Routing Problems (ARPs) are one of the most popular class of combinatorial problems that model routing of agents to perform tasks over networks. In ARP the tasks are modeled as subset of arcs, and they are referred as service/required arcs. Capacitated Arc Routing Problem (CARP) is an ARP where tasks have resource demand, and the agents performing the tasks are resource restricted. For instance, ultrasonic inspection of railway track sections require water, and the amount of water an agent can carry is limited by its size. In certain practical ARPs, the arcs are not always available, e.g. in railways, the track sections between two signals is blocked till a locomotive leaves the 'block'. To perform inspections while ensuring least disruption to the regular train schedules, the time dictated (temporal) unavailability of tracks must be modeled. In this work, we incorporate Temporal Unavailability (TU) in the core problem statement of CARP, and propose CARP-TU. CARP is an NP-Hard problem that makes the search for an exact/optimal solution grow exponentially with size. The time-dependencies on the arcs add an additional challenge into the problem.

CARP-TW (Time Window) Dror (2000); Lannez et al. (2015) is a sibling problem with unavailability time window for service/required arcs. CARP-TU generalizes CARP-TW with multiple unavailability time windows for both service and deadhead (non-service) arcs. In this work, we will also discuss the challenges of introducing unavailability into deadhead arcs, which are addressed through the proposed column generation algorithm. Without the capacity restriction, the CARP-TU transforms into RPP-TU (Rural Postman Problem), which has been briefly described in this work. Roadway routing problems under traffic conditions are also some of the relevant studies with respect to temporal attributes. Such a challenging problem in roadway setting has been studied by Calogiuri et al. (2019), which attempts a Time-Dependent Rural Postman Problem (TDRPP) in a traffic/congestion environment. Other works on time-dependent routing involve Arc Path Arc Sequence (APAS) formulation for TDRPP Tan and Sun (2011); Sun et al. (2011); Tan et al. (2013), branch and bound algorithms for Time-dependent Travelling Salesman Problem (TDTSP) Cordeau et al. (2014), etc. We will extend the APAS formulation in this work, to ensure that order of traversal is intrinsic in the decision variables.

10.1016/j.ifacol.2022.04.007

In railway research literature, maintenance scheduling problems are relatively emphasized over inspection scheduling. Most maintenance scheduling problems involve large locomotives, and require significant time for operations. Hence train schedule disruption is unavoidable in such planning problems Pour et al. (2018); Budai et al. (2006). These problems are therefore formulated as Vehicle Routing Problems (VRP). Maintenance planning that avoids train schedule disruptions are generally studied as scheduling problems i.e. without the component of routing Dao et al. (2018). Inspection routing and schedule using foot crew and/or multi-modal agent (agents that traverse on tracks as well as roads) are not affected by unavailability of deadhead traversals. Hence CARP-TW and VRP-TW have been suitable choice for the formulation of such inspection routing and scheduling problems Lannez et al. (2015); Peng et al. (2013). Planning of a fully track based agent requires unavailability consideration of the deadhead tracks, especially for busy networks (say sub-urban railway network). Hence we study CARP with Temporal Unavailability (TU) to enable efficient operation of railway network.

In temporal problems like CARP-TU, CARP-TW, etc, the order of traversal affects the optimality of the solution, and hence the decisions must involve the order of servicing and/or deadhead traversals. Since CARP-TU also has time-dependent/temporal attributes, a suitable choice for decision variables is very important to capture the order of arc traversal. In CARP, the state-of-the-art formulations are derived from set covering problems Pecin and Uchoa (2019); Bartolini et al. (2013), where the decisions involve selection from a list of *routes* (sequence of required arcs). The use of column generation algorithm is the key motivating factor for a set covering based formulation for CARP. Observe that the routes are sequences and therefore they have intrinsic ordering, this makes set covering approach a suitable choice for representing CARP-TW also. However for CARP-TU, the deadhead traversal order is also important, and cannot be overlooked. This requirement will be discussed in detail in this work. As a consequence of this dependence on deadhead traversal history, and its application in large scale industries like railway, thorough research on CARP-TU is requisite. Formulations, exact/optimal solutions, lower bounds, heuristics, etc are some of the key studies essential for exploring CARP-TU.

The main contributions of this work are:

- A three-index formulation for CARP-TU, as well as a set covering formulation for the spatial part of the problem. In addition, conversion of RPP-TU to CARP-TU is also presented.
- Three approaches for determining lower bounds for CARP-TU using column generation. The propositions are supported with comparison study based on the resulting lower bounds, to qualitatively determine the problem scale suitable for each approach.

This work is structured as follows: Section 3 refreshes some of the key concepts adapted from literature. Next, Section 4 proposes a three-index formulation for CARP-TU that has intrinsic ordering. A route based Set Covering formulation for the spatial sub-problem is then proposed for applying column generation on CARP-TU. In Section 5, modifications are proposed for the column generation algorithm, to improve the lower bounds and dual solutions of CARP-TU. Various comparisons have been made in Section 6 to validate the results and draw qualitative analysis against similar problems from literature. The results also compare two variants of column generation implementations suitable for CARP-TU. Lastly, Section 7 concludes with a discussion on the problem and algorithm, followed by insight into the future research prospects.

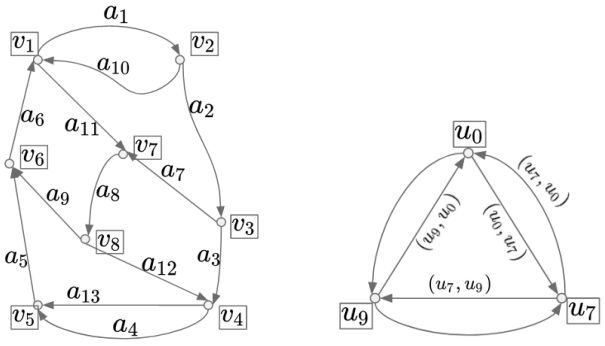
3. PRELIMINARIES

In this section, we describe some key terminologies and concepts used in this work.

3.1 Few definitions

Directed multi-graph: A directed multi-graph is an ordered 4-tuple $G = (V, A, F^+, F^-)$; where V is a vertex set, A is an arc set, $F^+ : A \to V$ is a map that assigns a head vertex to each arc, and $F^- : A \to V$ is a map that assigns a tail vertex to each arc. Every arc in the arc set A is directed from its tail vertex to its head vertex. Such graphs are useful for representing railway and roadway networks with multiple paths between same locations.

Figure 1(a) shows a directed multi-graph with vertex set $V = \{v_1, \ldots, v_8\}$ and arc set $A = \{a_1, \ldots, a_{13}\}$, where a_4 and a_{13} are parallel arcs. Both arcs a_4 and a_{13} are directed from their tail vertex $F^-(a_4) = F^-(a_{13}) = v_4$ to their head vertex $F^+(a_4) = F^+(a_{13}) = v_5$.



(a) An illustrative multi-graph $G = (V, A, F^+, F^-)$, where $V = \{v_1, \ldots, v_8\}$ and $A = \{a_1, \ldots, a_{13}\}$, F^+ and F^- are suitable maps.

(b) A simple graph $\mathbb{G} = (\mathbb{V}, \mathbb{A})$, with vertex set $\mathbb{V} = \{u_0, u_7, u_9\}$ and arc set $\mathbb{A} = \{(u_0, u_7), \ldots, (u_9, u_0)\}$.

Fig. 1. If $A_R = \{a_7, a_9\} \subset A$ represent a set of required/service arcs, then $\mathbb{G}$ is an equivalent VRP simple graph representation for ARP multi-graph G; see Section3.2.1.

Directed simple-graph: A directed simple-graph is denoted as $\mathbb{G} = (\mathbb{V}, \mathbb{A})$, where $\mathbb{V}$ is a vertex set, and $\mathbb{A}$ is an arc set whose elements are ordered 2-tuples of the elements of vertex set. It is a special case of directed multi-graph with no parallel arcs, and hence a 2-tuple notation is sufficient to describe the arcs of this graph; see Figure 1(b).

3.2 Basics on CARP

Arc Routing Problem (ARP) to Vehicle Routing Problem(VRP) The conversion process compresses the deadheading information in the ARP setting. Hence it is useful for formulating and solving CARP instances Blais and Laporte (2003), Ghiani and Improta (2000), Bartolini et al. (2013). However, in CARP-TU, the temporal attributes cause the deadheading decisions to affect the overall cost of traversal. Hence, we discuss the conversion process here, and show dependence on deadhead traversal history in case of CARP-TU.

Let the following information be available for an Arc Routing Problem (ARP): a *directed multi-graph* $G = (V, A, F^+, F^-)$; costs associated with the arcs $c_m, \forall a_m \in A$; time traversal associated with the arcs $t_m, \forall a_m \in A$; a depot vertex $v_1 \in V$; and a subset of the arc set requiring servicing called service/required arc subset $A_R \subseteq A$. To convert this problem setting to a Vehicle Routing Problem (VRP) setting, the depot and all required arcs are represented as a vertex of a new fully-connected *directed simple-graph* $\mathbb{G} = (\mathbb{V}, \mathbb{A})$, where $|\mathbb{V}| = |A_R| + 1$. In particular, required arc $a_m \in A_R \subseteq A$ in multi-graph G and vertex $u_m \in \mathbb{V}$ in simple-graph $\mathbb{G}$ represent the same entity; similarly the depot location is represented by vertex v_1 in graph G and vertex u_0 in graph $\mathbb{G}$. Figure 1 shows a VRP simple-graph $\mathbb{G}$ and an equivalent ARP multi-graph G; provided the depot in graph G is $v_1 \in V$ (equivalently $u_0 \in \mathbb{V}$ in graph $\mathbb{G}$), and the required arcs in G are $A_R = \{a_7, a_9\} \subseteq A$ (equivalently the corresponding required vertices of $\mathbb{G}$ are $\mathbb{V}_R = \{u_7, u_9\} = \mathbb{V} \backslash \{u_0\}$).

The VRP arc set $\mathbb{A}$ is represents specifically selected paths between the required arcs and depot in the ARP graph. The selection of an ARP path to represent as VRP arc is based on least cost of traversal between required arcs and depot. In particular, the VRP arc cost is measured based on least cost of traversal from center of one required arc/depot to another required arc/depot in the ARP graph. For CARP-TU, selection based on least cost paths will be replaced by a different rule; see Section 5.

For VRP arc denoted using ordered 2-tuple $(u_7, u_0) \in \mathbb{A}$ in Figure 1(b), the VRP arc cost (denoted by $c_{7,0}$) is equal to the least cost path in Figure 1(a) from the center of arc a_7 to vertex v_1. Travel time associated with these least cost paths are also computed in the process. Equation (1) and Equation (2) represent the cost $c_{m,n}$ and travel time $t_{m,n}$ associated with an arc (pair of vertices u_m and u_n) of the converted graph $\mathbb{G}$ respectively.

$$c_{m,n} = \frac{1}{2}c_m + lc(a_m, a_n) + \frac{1}{2}c_n; \quad \forall a_m, a_n \in A_R \tag{1}$$

$$t_{m,n}(t^o_m) = \frac{1}{2}t_m + tt(a_m, a_n, t^o_m) + \frac{1}{2}t_n; \quad \forall a_m, a_n \in A_R \tag{2}$$

Here, $lc(a_m, a_n)$ is the least cost path, and $tt(a_m, a_n, t^o_m)$ is the corresponding travel time from the tail of arc $a_m \in A_R$ to the head of arc $a_n \in A_R$ with start time t^o_m. The equations are easily extended for cost and time representations involving depot: $c_{0,m}$, $c_{m,0}$, $t_{0,m}$, and $t_{m,0}$; where $a_m \in A_R$.

Note that, the travel time function $tt(a_m, a_n, t^o_m)$ depends on the time taken to reach the center of arc a_m in temporal problems like CARP-TU. This start time is denoted by t^o_m, and it is given as the sum of departure time [1] at tail vertex $F^-(a_m)$ and half the time value of the arc a_m. As a consequence, the start time computation is not only dependent on servicing history but also on the delays occurred due to unavailability of deadhead arcs before arriving at the center of arc a_m. For VRP traversal time computations $t_{m,n}$, the dependence on start time is ignored by assuming no waiting/delay at the intermediate vertices and arcs. Algorithm 1 shows the conversion of ARP problem setting to VRP problem setting.

Observe that, the indices m and n are reserved for arcs of ARP and corresponding vertices of VRP, i.e. $a_m \in A_R$ and $u_m \in \mathbb{V}_R$ represent the same physical entity (say, a railway-track section). In particular, usage of the notation $a_m \in A_R$ implies the following relations: a_m is a required arc [2] in ARP graph G; $u_m \in \mathbb{V}_R$ is the corresponding required vertex in VRP graph $\mathbb{G}$; $(u_m, u_n) \in A$ is an arc of VRP graph $\mathbb{G}$ that represents a least cost path originating at the center of the required arc a_m and terminates at the center of some other required arc a_n; etc.

Algorithm 1 ARP to VRP

Input: ARP setting: $G = (V, A, F^+, F^-), A_R, c_m, t_m$
Output: VRP setting: $\mathbb{G} = (\mathbb{V}, \mathbb{A}), \mathbb{V}_R, c_{m,n}, t_{m,n}$
Initialisation (line 1) : Add a vertex u_0 in $\mathbb{V}$
Initialisation (line 2) : Add vertex u_m in $\mathbb{V}$ and $\mathbb{V}_R$, for each arc a_m in A_R

```
for u_0 or u_m in V do
    for u_0 or u_n in V do
        if (m == n) then
            skip (no self loop)
        end
        Add an arc (u_m, u_n) or (u_0, u_n) or (u_m, u_0) in A
        Calculate c_{m,n} or c_{0,n} or c_{m,0}; Equation (1)
        Calculate t_{m,n} or t_{0,n} or t_{m,0}; Equation (2)
    end for
end for
```

Route set Any tour in the VRP graph $\mathbb{G}$ that starts and ends at the depot vertex is called a **route**. The routes are described using the graph $\mathbb{G} = (\mathbb{V}, \mathbb{A})$ obtained from ARP to VRP conversion. Index of the set of all such routes is called a *route set*, denoted by $\mathscr{R}$, and the route (tour that starts and ends at the depot vertex) itself is denoted by $R_r, r \in \mathscr{R}$. The set $\mathscr{R}$ is an integer subset of the form $\{1, 2, \ldots, r, \ldots\}$; implying R_r is the route listed as r^{th} entry. The sequence of vertices $R_1 = \{u_0, u_7, u_9, u_7, u_0\}$ is an example for a route in VRP graph $\mathbb{G}$ (see Figure 1(b)) listed as 1^{st} entry ($r = 1$), where u_0 is the depot vertex.

[1] Departure time captures the traversal history data, that includes both running time and waiting. Using path data and starting time, the time of arrival at any vertex can be computed; and then the addition of the waiting time at that vertex yields the departure time at the vertex.

[2] For arc $a_m \in A_d := A \backslash A_R$, there is no corresponding VRP vertex. Hence it is important to specify that arc a_m is a required arc from set A_R wherever its VRP equivalent has to be exploited.

The cost of a route is constructed using the VRP costs $c_{m,n}$, as shown in Equation (3). For every consecutive pair of vertices in a route $R_r, r \in \mathscr{R}$, the VRP costs $c_{0,\iota(r,2)}, c_{\iota(r,m-1),\iota(r,m)}$ and $c_{\iota(r,|R_r|-1),0}$ are computed from Equation (1), and then summed up. Here, $\iota(r,m)$ is a function that returns the index of m^{th} vertex in route R_r i.e. $\iota(r,1) = \iota(r,|R_r|) = 0,\ \forall r \in \mathscr{R}$ indicating first and last vertex of any route is the depot vertex u_0. Travel time of a route is also computed in the same way, as shown in Equation (4).

$$\bar{c}_r = c_{0,\iota(r,2)} + \sum_{m=3}^{|R_r|-1} c_{\iota(r,m-1),\iota(r,m)} + c_{\iota(r,|R_r|-1),0} \quad (3)$$

$$\bar{t}_r = t_{0,\iota(r,2)} + \sum_{m=3}^{|R_r|-1} t_{\iota(r,m-1),\iota(r,m)} + t_{\iota(r,|R_r|-1),0} \quad (4)$$

We introduce tilde notation for time ($\tilde{t}_r$) to denote that the travel time computation of route R_r is feasible with respect to temporal constraints in the problem (if any). In particular, the start time t^o_m dependence, shown in Equation (2), is taken into consideration while computing the travel time associated with the route.

q-route relaxation and Dynamic Programming The concept of *q-m-routes* is same as that of *q-routes* introduced to generate lower bounds for CARP Christofides et al. (1981a). The subscript m represents a required vertex $u_m \in \mathbb{V}_R$. A *q-m-route* is any non-elementary [3] route that started at a depot, serviced some of the required arcs any non-negative number of times, and then at the end has serviced vertex $u_m \in \mathbb{V}_R$ before terminating at the depot, while meeting a total demand of q units. In particular, a *q-m*-route is a route-plan of an agent that satisfies a total of q demand units, and whose last servicing assignment is the required vertex u_m. The *q-m*-route computation is modified in Section 5 to produce lower bounds in a column generation algorithm for CARP-TU.

The *q-m*-routes are determined using Dynamic Programming (DP) on a relaxed state-space, constructed from the VRP graph $\mathbb{G}$, with cost $c_{m,n}$ for each arc $(u_m, u_n) \in A$, see Bartolini et al. (2013); Christofides et al. (1981a,b). In particular, the states are routes [4] encoded as (q,p,m); where q is the sum total demand met by the route, and u_p, u_m are the last two vertices visited by the route (u_m being the last) before returning to depot u_0. The transition between the states implies addition of another vertex (say u_n) to the end of *q-m*-route, such that $u_p \neq u_n$ to avoid 2-cycles [5], and $q + q_n \leq Q$ to ensure demand doesn't exceed the capacity restriction. Hence the newly explored state space (route) is given by $(q+q_n, m, p)$. The state space exploration is governed by the minimum cumulative *reduced cost* of the routes, given as $\sum_{u_m \in R_r} c_m - \pi_m$; where R_r is a route (sequence of required vertex), and π_m is a dual value achieved from a pricing algorithm. Note that this DP is a relaxation because cycles of length 3 or larger are also allowed. An exact DP would require defining state as all the vertices of a route, and then monitoring them to avoid cycle formation; thereby making the DP much slower.

3.3 Replicated graph and the Arc Path Arc Sequence

The concept of replicated graph is a graphical extension to the Arc Path Arc Sequence (APAS) formulation Tan and Sun (2011), where the structure of APAS is identical to the replicated graph. Both the concepts are useful for problems with temporal/time-varying data. We present a brief overview of the replicated graph in this work for brevity.

Consider the deadhead graph $G' := (V, A \backslash A_R, F^+, F^-)$ constructed from the directed multi-graph G as shown in Figure 2, where service/required arcs A_R are removed from the arc set A. This deadhead graph G' is re-drawn over $|A_R|+1$ layers for each agent, and connected downwards using arcs in A_R. The resulting graph has $|\mathscr{K}|$ (graph) components, one for each agent, as shown in Figure 3. This new layered graph is termed as replicated graph $\mathscr{G} = (\mathscr{V}, \mathscr{A}, \mathscr{F}^+, \mathscr{F}^-)$, with vertex set $\mathscr{V}$, arc set $\mathscr{A}$, and two maps $\mathscr{F}^+ : \mathscr{A} \to \mathscr{V}$ and $\mathscr{F}^- : \mathscr{A} \to \mathscr{V}$ for identifying the head and tail vertices respectively. The vertex and arc notations are indexed to identify the association with agents and layers (order of traversal) i.e. v_{ikl} is a vertex in $\mathscr{V}$, a_{mkl} is an arc in $\mathscr{A}$; where k associates the vertex $v_i \in V$ and arc $a_m \in A$ to an agent from set $\mathscr{K}$ and layer $l \in \mathscr{L}$ ($\mathscr{L} := \{1, \ldots, |A_R|+1\}$)]. Note that the indices are separated by commas if necessary, e.g. weight for arc a_1, agent $k=1$ and layer $l=11$ is denoted as $\mathscr{W}_{1,1,11}$ for brevity.

Structure of the replicated graph ensures that all deadhead arcs between two consecutive servicing tasks belong to same layer. Pair of virtual vertices, called source vertex v_s and destination/sink vertex v_d, are introduced for each agent in the zeroth layer as v_{sk0} and v_{dk0}. Source and sink vertices are connected to the depot set and exit set respectively, using virtual arcs (zero running-time) such that trajectory always starts at source, visits a vertex in depot set, followed by servicing of arcs of $\mathbb{G}$, then visits a vertex of exit set, and finally ends at the sink vertex. A source-sink virtual arc is also introduced for each agent to model idle agent.

4. CAPACITATED ARC ROUTING PROBLEM WITH TEMPORAL ATTRIBUTES DUE TO ARC UNAVAILABILITIES (CARP-TU)

In this section, we discuss the problem statement for CARP-TU; followed by three-index formulation, and set covering formulation. Relation with RPP-TU (CARP-TU without capacity restriction) is also discussed in brief in this section.

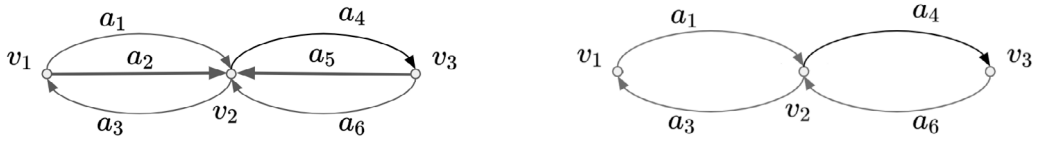
4.1 Problem statement

The formal problem statement for CARP-TU is described as follows: given a directed multi-graph $G =$

[3] A route that re-visits a vertex (other than the depot) more than once i.e. it services an arc more than once.

[4] There are multiple possible routes with demand q and u_p, u_m being the last two servicing. Only the least cost route is considered in this relaxation.

[5] Cycle with two vertices, e.g. u_p, u_m, u_p.



(a) Graph G: vertices $v_1, v_2, v_3 \in V$, arcs $a_1, a_2, a_3, a_4, a_5, a_6 \in A$, and service arcs $a_2, a_5 \in A_R$.

(b) Graph G': vertices $v_1, v_2, v_3 \in V$, arcs $a_1, a_3, a_4, a_6 \in A \backslash A_R$.

Fig. 2. An example graph picked from article on CARP-TW with refill times Lannez et al. (2015). Here v_1 is the depot set and exit set.

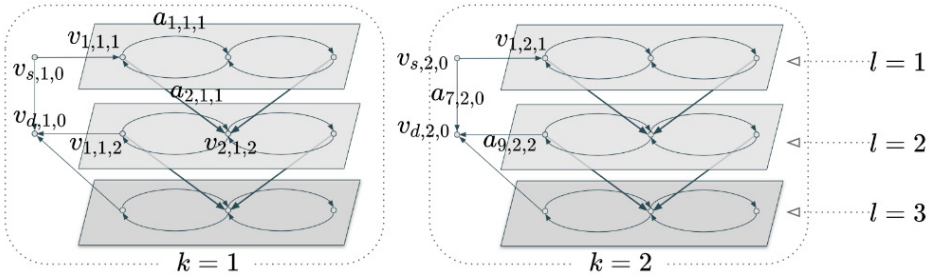


Fig. 3. A replicated graph for $|\mathscr{K}|$ agents, constructed from a base graph shown in Figure 2(a). Here $k \in \mathscr{K} (= \{1, 2\})$ is the agent set. The two dotted boxes illustrate agent-sub-graphs having 3 layers each $l \in \mathscr{L} (= \{1, 2, 3\})$, illustrated by planes.

(V, A, F^+, F^-), cost and time of traversal for each arc c_m & t_m, $\forall a_m \in A$, a subset of arcs to be serviced $A_R \subseteq A$, demand for each service/required arc q_m, $\forall a_m \in A_R$, and a set of agents $\mathscr{K}$ with capacity restriction Q, a list of unavailabilities $(Z = \cup_{a_m \in A} Z_m)$ for each arc such that an entry $(\underline{\omega}, \overline{\omega}) \in Z_m$ describes the time interval of duration $\overline{\omega} - \underline{\omega}$ between which arc a_m is unavailable; determine optimal tours for the agents such that each agent satisfies at most Q units of demand (i.e. the sum of demand of all the required arcs serviced by an agent must be at most Q). The cost function for optimization penalizes the cost of traversal (spatial) and the time of completion (temporal) of all tasks. A factor β is used to adjust the relative penalty of temporal cost with respect to the spatial cost. It is *assumed* that the vertices are always available and they don't have any time restrictions, unlike the arcs.

4.2 Formulation of CARP-TU

The CARP-TU is an optimization problem with spatial and temporal decisions, where the optimization cost is governed by both service and deadhead arc traversals. Figure 4 shows two parallel arcs where the least cost deadhead traversal of arc a_1 is smaller than arc a_2, however the time of traversal is larger, $t_1 \geq t_2$. Since the unavailability window of arc a_3 is $[t_2 + \epsilon, T]$, where T is a large time value, the choice of deadhead traversal between a_1 and a_2 will impact the time component of upcoming paths (unlike CARP-TW where time taken by deadhead paths don't change). Hence, the usual set covering formulation for CARP is not ideally appropriate for CARP-TU. We, therefore, formulate CARP-TU using three-indexes of replicated graph. We also introduce a set covering formulation for CARP-TU here, that will be used to generate lower and upper bound for the optimal CARP-TU solution in Section 5.

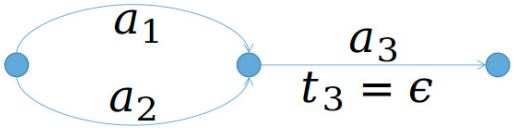


Fig. 4. Three arcs, where cost of arc a_1 is smaller than arc a_2, while time of arc a_2 is smaller than arc a_1. Arc a_3 is unavailable after $t_2 + \epsilon$ and before T, implying traversal of this arc a_3 is possible before time t_2 or after time T.

Three-index formulation The spatial decisions of the agents are denoted by X_{mkl} for every arc $a_{mkl} \in \mathscr{A}$, while the schedule decisions are denoted by Γ_{ikl} for every vertex $v_{ikl} \in \mathscr{V}$ in the replicated graph (see Section 3.3 for description on replicated graph). Here, q and i are identifiers for arc $a_m \in A$ and vertex $v_i \in V$ respectively, $k \in \mathscr{K}$ is positive integer indicating an agent, and $l \in \mathscr{L}$ is a non-negative integer indicating the order of traversal.

$$X_{mkl} = \begin{cases} 1, & \text{if arc } a_m \text{ is traversed by agent } k \text{ in between} \\ & \text{servicing of } (l-1)^{th} \text{ and } l^{th} \text{ required arc} \\ 0, & \text{otherwise} \end{cases}$$

$\Gamma_{ikl} \rightarrow$ departure time of agent k at vertex v_i, in between servicing of $(l-1)^{th}$ and l^{th} required arc

The formulation is mathematically stated as:

$$\min \sum_{m,k,l} \mathscr{W}_{mkl} X_{mkl} + \beta\, \Gamma \tag{5a}$$

$$\text{where, } \mathscr{W}_{mkl} = \begin{cases} W_{mk} & a_m \in A, k \in \mathscr{K} \\ 0 & \text{otherwise} \end{cases}$$

$$s.t.\ \Gamma \geq \Gamma_{dk0},\ \forall k \in \mathscr{K} \tag{5b}$$

$$\sum_{a_{mkl} \in \delta^+_{ikl}} X_{mkl} - \sum_{a_{mkl} \in \delta^-_{ikl}} X_{mkl} \tag{5c}$$

$$= \begin{cases} -1, & \text{if } i = s,\ l = 0; \forall k \in \mathscr{K}\} \\ 1, & \text{if } i = d,\ l = 0; \forall k \in \mathscr{K}\} \\ 0, & \text{otherwise} \end{cases}$$

$\forall v_{ikl} \in \mathscr{V}$

$$\sum_{a_{mkl} \in H_m} X_{mkl} = 1, \quad \forall a_m \in A_R; \tag{5d}$$

where, $H_m = \{a_{mkl} \in \mathscr{A}_R \mid k \in \mathscr{K}, l \in \mathscr{L}\}$

$$\sum_{a_{mkl} \in \tilde{H}_k} q_m X_{mkl} \leq Q, \quad \forall\ k \in \mathscr{K}; \tag{5e}$$

where, $\tilde{H}_k = \{a_{mkl} \in \mathscr{A}_R \mid a_m \in A_R, l \in \mathscr{L}\}$

$$\Gamma_{jkl'} \geq \mathscr{W}_{mkl} + \Gamma_{ikl} + (1-\tau)X_{mkl};\ \forall a_{mkl} \in \mathscr{A} \tag{5f}$$

Γ_{ikl} *variable at tail vertex* $v_{ikl} = \mathscr{F}^-(a_{mkl})$,

$\Gamma_{jkl'}$ *variable at head vertex* $v_{jkl'} = \mathscr{F}^+(a_{mkl})$

$$\left.\begin{array}{l} \textit{either,} \\ \quad \Gamma_{jkl} \leq \underline{\omega} - W_m + \tau(1 - X_{mkl}) \\ \textit{or,} \\ \quad \Gamma_{jkl} \geq \overline{\omega} - \tau(1 - X_{mkl}) \end{array}\right\} \tag{5g}$$

$$X_{mkl} \in \{0,1\}, \Gamma_{jkl} \geq 0 \tag{5h}$$

The cost function is given in Equation (5a) that minimizes the total cost of traversal and β (> 0) times the maximum servicing time among all agents (see Equation (5b)). The generated solution is spatially traversable due to the flow constraints in Equation (5c), and the running time constraint in Equation (5f) (see MTZ formulation Miller et al. (1960)). Obligation on the spatial movements is due to the task requirements and capacity restrictions imposed by Equation (5d) and Equation (5e) respectively. Correct schedule for the routes are ensured by the running time constraints in Equation (5f) and unavailability constraints in Equation (5g).

Set covering formulation for spatial sub-problem The spatial sub-problem of CARP-TU may be represented using route selection variables, as shown in Formulation 6. Here, only the order of service arcs is encoded into the route identifiers i.e. each route R_r is a sequence of required arcs. By an appropriate choice + modification of the cost for the CVRP spatial sub-problem formulation, it is possible to generate lower and upper bounds for CARP-TU. The route based set covering formulation is shown in 6.

$$\min_{x_r} \quad \sum_{r \in \mathscr{R}} \bar{c}_r x_r \tag{6a}$$

$$s.t. \quad \sum_{r \in \mathscr{R}} b_{mr} x_r \geq 1,\ \forall u_m \in \mathbb{V}_{RC} \tag{6b}$$

$$\sum_{r \in \mathscr{R}} x_r \geq 1;\ \forall k \in \mathscr{K} \tag{6c}$$

$$\sum_{r \in \mathscr{R}} x_r = |\mathscr{K}| \tag{6d}$$

$$x_r \in \{0\} \cup \mathbb{N} \tag{6e}$$

Formulating RPP-TU as CARP-TU The optimal solution of RPP-TU is bounded by $|A_R| + 1$ servicing tasks per agent, where A_R is the number of required arcs in G; see replicated graph in Section 3.3. The bound is valid under the assumption [6] that there exist a solution for the given CARP-TU problem dataset. Due to this bound, an exact conversion of RPP-TU to CARP-TU is possible by defining demand of each required arc as 1, and the capacity of the agents to be $|A_R|$.

5. ALGORITHM

5.1 Bound computations for CARP-TU

The q-m-routes (see Preliminaries in Section 3.2.3) act as a relaxation to the true route set, hence producing lower bounds in the column generation algorithm for CARP Dror (2000). However, in CARP-TU, such q-m-routes generated using least cost paths are not a suitable choice for lower bound, since the least cost (lc) solution may have large time delays thereby increasing the spatio-temporal cost. Quickest time (qt) paths are also an invalid choice for determining lower bounds for CARP-TU due to similar reasoning. We propose a combination cost estimate for q-m-routes, as shown in Equation (7a) and (7b), where the tilde notation over time implies that it satisfies all unavailability restrictions (see Preliminaries in Section 3.2.2).

$$\bar{c}_r^{LB} = \sum_{i \in R_r} c_i^{lc} + \beta t_i^{qt} \tag{7a}$$

$$\leq \bar{c}_r^{lc} + \beta \tilde{t}_r^{qt} \tag{7b}$$

$$\leq \bar{c}_r + \beta \tilde{t}_r$$

In contrast to the q-route based dynamic programming, the proposed approach utilizes $\bar{c}_r^{LB}$ to generate costs for the q-m-routes instead of the true arc costs. This ensures cost of all CARP-TU solutions [7] with servicing sequence given by the generated q-route (R_r) are lower bounded by cost of the q-route itself. In addition, there is no computational change due to the proposed approach. Note that an *upper bound to the optimal CARP-TU* solution with a given servicing sequence R_r is given by $\min(\bar{c}_r^{lc} + \tilde{t}_r^{lc}, \bar{c}_r^{qt} + \tilde{t}_r^{qt})$.

5.2 Column generation algorithm for spatial sub-problem

The modified column generation algorithm for lower bound computation of CARP-TU is illustrated in Algorithm 2. Two variants are proposed for the algorithm for analyzing the upper and lower bound improvements. The *first variant* evaluates the standard column generation algorithm with least cost VRP graph, and then modifies the costs later using Equation (7a) and (7b) (changes introduced in step S3 of Algorithm 2). The *second variant* evaluates the column generation over the mixed cost VRP graph, i.e. the VRP arc costs are derived directly from Equation (7a) (step S1 of Algorithm 2 is modified accordingly).

[6] A theoretical study, that characterizes the dataset to guarantee non-empty feasible region, is out of the scope of this work.

[7] There may be multiple deadheading choices between two service tasks.

Algorithm 2 Column generation for CARP-TU

Input: CARP problem data

Output: A dual solution Π, lower bound c_{LB}, feasible route and/or partial route solution X (if any), and upper bound (if any) c_{UB}

Initialization : Route subset $\tilde{\mathscr{R}} = \emptyset$, Lagrangian and dual variables $\Lambda = \Pi = 0$, and subset $\mathscr{N} \subseteq \tilde{\mathscr{R}}$

for *maxit1* iterations **do**
  S1: *Generate routes using q-paths (DP)*
  $\tilde{\mathscr{R}} \leftarrow$ Store q-m-routes
  S2: *Lower bound + breaking condition*
  if no new q-m-routes **then**
    $c_{LB} \leftarrow c_{LRP}$
    if c_{LB} equals c_{UB} **then**
      break
    end
  end
  for *maxit2* iterations **do**
    S3: *Determine best route, one for each required arcs*
    for each required arcs $a_i \in A_R$ **do**
      $\mathscr{N} \leftarrow$ Select route and store for evaluation
      $LB_u, UB_u \leftarrow$ Compute proposed lower bound and upper bound based on unavailability dataset
    end for
    S4: *Compute upper bound and mark if its feasible solution*
    if $\mathscr{N}$ produces a feasible solution **then**
      $c_{UB} \leftarrow$ Compute using UB_u
      $X \leftarrow$ Save the feasible route solution
    end
    S5: *Determine sub-gradients for maximizing LRP, dual solution and reduced cost (see Bartolini et al. (2013))*
    $\Lambda, c_{LRP}, \Pi \leftarrow$ Compute for generating improved routes
  end for
end for
return Π, c_{LB}, X, c_{UB}

6. RESULTS

In this section, the algorithm is tested over benchmark dataset, as well as randomly generated bulk dataset. Numerical comparison of two algorithm variants are also discussed in this section.

6.1 Validation with CARP

We perform validation of our algorithm against column generation for CARP by Bartolini et al. (2013), over the benchmark dataset proposed by Eglese Eglese and Li (1992). The results of the simulation study are shown in Table 1. Observe that lower bound computation over a large instance of 51 required arcs was successful. For spatial cost only ($\beta = 0$) scenario, the results produced by the first variant of our proposed algorithm match the benchmark lower bound results form literature (columns: *LB* and *LB-spatial*). On introducing time into the cost function ($\beta = 1$) as shown in Equation (5a) and excluding unavailability ($Z = \{\emptyset\}$), the lower bound value doubled approximately (columns: *LB-spatial* and *LB-no-unav*); because the time values are chosen slightly (randomly) larger than the cost of the arcs. On adding unavailability list data, the new lower bound increased further, and surpassed the $2 * UB$ mark, thereby ensuring non-trivial bounds (columns: *UB* vs *LB-unav*).

LB computation over random graphs for RPP-TU A set of random graphs were generated for three vertex-arc settings, as shown in Table 2. A total of 30 random graph instances were evaluated for each vertex-arc setting, resulting in a total of 90 simulations. By ensuring existence of a Hamiltonian cycle as well as well-defined temporal (unavailability) dataset in each of the random graphs, an existence of a solution is guaranteed. It is observed that on average the first variant (column: *Var 1*) produces better lower bounds than the second variant (column: *Var 2*), however for some instances the lower bound computations are improved significantly when using the second variant (column: *lbv2>lbv1* shows number of instances where *Var2* provided better lower bounds). In case of upper bounds, the second variant always generates more number of elementary[8] bounds (column: *ubv1>=ubv2* shows number of instances with better upper bounds due to *Var 2*).

7. CONCLUSION

Capacitated Arc Routing Problem with Temporal Unavailability (CARP-TU) is suitable for route planning of inspection robots in railway network. Camera based track inspection Resendiz et al. (2013), track geometry measurement using instrument wagons IRT (2000), etc are some of the tasks that would benefit from this research. This work highlights the differences with CARP-TW (Time Window) and formulates CARP-TU using three-index replicated graph based formulation. A set covering based formulation for the spatial sub-problem for CARP-TU is also proposed in this work. Further, it is observed that RPP-TU is directly formulated as CARP-TU due to the layered nature of the replicated graph.

Two variants of lower bound computation is proposed in this work, and then compared numerically to analyse the trade-offs. The study shows benefit of incorporating least cost and quickest time data in q-route computation, which is validated using 90 simulation results over random graphs. In our observation, its best to incorporate both variants for generating q-routes, where the second variant has significant advantage w.r.t upper bounds.

ACKNOWLEDGEMENTS

This work is supported by IITB-Monash Research Academy.

REFERENCES

(2000). Monash Institute of Railway Technology. https://www.monash.edu/irt/expertise.

Bartolini, E., Cordeau, J.F., and Laporte, G. (2013). Improved lower bounds and exact algorithm for the capacitated arc routing problem. *Mathematical Programming*, 137(1), 409–452. doi:10.1007/s10107-011-0497-4. URL https://doi.org/10.1007/s10107-011-0497-4.

[8] Elementary implies that the routes will service an arc only once, see Preliminaries in Section 3.2.3.

Table 1. Validating our algorithm by generating similar bounds for CARP, and increased bounds for CARP-TU

Dataset	No of required arcs	No of agents	UB Bartolini et al. (2013)	LB Bartolini et al. (2013)	LB-spatial (proposed)	LB-no-unav (proposed)	LB-unav (proposed)
Eglese1A	51	5	3548	3395	3323	6970	8927
Eglese1B	51	7	4498	4246	4207	8777	11686

Table 2. 90 simulations over randomly generated graphs

				Var 1 (lbv1)	Var 2 (lbv2)	lbv2 > lbv1	max(lbv2-lbv1)	ubv1 >= ubv2
No. of V	No. of A	No. of A_R	Avg LB (no unav)	Avg LB (with unav)	Avg LB (with unav)	No. of better instances	Best improvement	No. of extra UB instances
20	32	8	220.26	231.32	228.84	5	219.17	5
40	63	15	429.76	512.16	461.69	0	0	1
60	94	22	569.6	670.89	476.39	4	760.19	0

Blais, M. and Laporte, G. (2003). Exact solution of the generalized routing problem through graph transformations. *The Journal of the Operational Research Society*, 54(8), 906–910.

Budai, G., Huisman, D., and Dekker, R. (2006). Scheduling preventive railway maintenance activities. *Journal of the Operational Research Society*, 57(9), 1035–1044. doi:10.1057/palgrave.jors.2602085. URL https://doi.org/10.1057/palgrave.jors.2602085.

Calogiuri, T., Ghiani, G., Guerriero, E., and Mansini, R. (2019). A branch-and-bound algorithm for the time-dependent rural postman problem. *Computers & Operations Research*, 102, 150 – 157. doi:https://doi.org/10.1016/j.cor.2018.07.016. URL http://www.sciencedirect.com/science/article/pii/S0305054818302004.

Christofides, N., Mingozzi, A., and Toth, P. (1981a). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1), 255–282.

Christofides, N., Mingozzi, A., and Toth, P. (1981b). State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2), 145–164.

Cordeau, J.F., Ghiani, G., and Guerriero, E. (2014). Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. *Transportation Science*, 48(1), 46–58. doi:10.1287/trsc.1120.0449. URL https://doi.org/10.1287/trsc.1120.0449.

Dao, C., Basten, R., and Hartmann, A. (2018). Maintenance scheduling for railway tracks under limited possession time. *Journal of Transportation Engineering, Part A: Systems*, 144(8), 04018039. doi:10.1061/JTEPBS.0000163. URL https://ascelibrary.org/doi/abs/10.1061/JTEPBS.0000163.

Dror, M. (2000). *Arc Routing Theory, Solutions and Applications*.

Eglese, R.W. and Li, L.Y.O. (1992). Efficient routeing for winter gritting. *Journal of the Operational Research Society*, 43(11), 1031–1034. doi:10.1057/jors.1992.160. URL https://doi.org/10.1057/jors.1992.160.

Ghiani, G. and Improta, G. (2000). An efficient transformation of the generalized vehicle routing problem. *European journal of operational research*, 122(1), 11–17.

Lannez, S., Artigues, C., Damay, J., and Gendreau, M. (2015). A railroad maintenance problem solved with a cut and column generation matheuristic. *Networks*, 66(1), 40–56. doi:10.1002/net.21605. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21605.

Miller, C.E., Tucker, A.W., and Zemlin, R.A. (1960). Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4), 326329. doi:10.1145/321043.321046. URL https://doi.org/10.1145/321043.321046.

Pecin, D. and Uchoa, E. (2019). Comparative analysis of capacitated arc routing formulations for designing a new branch-cut-and-price algorithm. *Transportation Science*, 53(6), 1673–1694. doi:10.1287/trsc.2019.0900. URL https://doi.org/10.1287/trsc.2019.0900.

Peng, F., Ouyang, Y., and Somani, K. (2013). Optimal routing and scheduling of periodic inspections in large-scale railroad networks. *Journal of Rail Transport Planning & Management*, 3(4), 163 – 171. doi:https://doi.org/10.1016/j.jrtpm.2014.02.003. URL http://www.sciencedirect.com/science/article/pii/S221097061400016X.

Pour, S.M., Drake, J.H., Ejlertsen, L.S., Rasmussen, K.M., and Burke, E.K. (2018). A hybrid constraint programming/mixed integer programming framework for the preventive signaling maintenance crew scheduling problem. *European Journal of Operational Research*, 269(1), 341 – 352. doi:https://doi.org/10.1016/j.ejor.2017.08.033. URL http://www.sciencedirect.com/science/article/pii/S0377221717307646.

Resendiz, E., Hart, J.M., and Ahuja, N. (2013). Automated visual inspection of railroad tracks. *IEEE Transactions on Intelligent Transportation Systems*, 14(2), 751–760. doi:10.1109/TITS.2012.2236555.

Sun, J., Tan, G., and Hou, G. (2011). Branch-and-bound algorithm for the time dependent chinese postman problem. In *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, 949–954. doi:10.1109/MEC.2011.6025622.

Tan, G. and Sun, J. (2011). An integer programming approach for the rural postman problem with time dependent travel times. In B. Fu and D.Z. Du (eds.), *Computing and Combinatorics*, 414–431. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tan, G., Sun, J., and Hou, G. (2013). The time-dependent rural postman problem: polyhedral results. *Optimization Methods and Software*, 28(4), 855–870. doi:10.1080/10556788.2012.666240. URL https://doi.org/10.1080/10556788.2012.666240.