# Chapter 5
# A Branch-and-Price Framework for the Maximum Covering and Patrol Routing Problem


**Paul A. Chircop, Timothy J. Surendonk, Menkes H. L. van den Briel, and Toby Walsh**

**Abstract** The Maximum covering and patrol routing problem (MCPRP) is concerned with the allocation of police patrol cars to accident hotspots on a highway network. A hotspot is represented as a time window at a precise location on the network at which motor vehicle accidents have a high probability of occurring. The nature of these accidents may be due to speeding, driver fatigue or blind-spots at intersections. The presence of police units at hotspots serves as an accident prevention strategy. In many practical applications, the number of available cars cannot cover all of the hotspots on the network. Hence, given a fleet of available units, an optimization problem can be designed which seeks to maximize the amount hotspot coverage. The cars must be routed in such a way as to avoid multiple contributions of the patrol effort to the same hotspot. Each police car is active over a predefined shift, beginning and ending the shift at a fleet station. In this paper, we introduce a method for constructing a time-space network of the MCPRP which is suitable for the application of a branch-and-price solution approach. We propose some large-scale test problems and compare our approach to a state-of-the-art Minimum cost network flow problem (MCNFP) model. We show that our branch-and-price approach can outperform the MCNFP model on selected large-scale networks for small to medium fleet sizes. We also identify problems which are too large for the MCNFP model to solve, but which can be easily handled by our approach.

P. A. Chircop (✉) · T. J. Surendonk
Defence Science and Technology Group, Sydney, NSW, Australia
e-mail: paul.chircop@dst.defence.gov.au

T. J. Surendonk
e-mail: timothy.surendonk@dst.defence.gov.au

M. H. L. van den Briel
Hivery, Sydney, NSW, Australia
e-mail: menkes@hivery.com

T. Walsh
UNSW Sydney, Data61, Kensington, NSW, Australia
e-mail: toby.walsh@data61.csiro.au

TU Berlin, Berlin, Germany



A. T. Ernst et al. (eds.), *Data and Decision Sciences in Action 2*,
Lecture Notes in Management and Industrial Engineering,
https://doi.org/10.1007/978-3-030-60135-5_5

**Keywords** Route planning · Surveillance scheduling · Branch-and-price

## 5.1 Introduction and Background

The Maximum covering and patrol routing problem (MCPRP) was first studied by Keskin et al. [10]. Given a set of highway locations and time intervals at which traffic accidents have a high probability of occurring, the problem is to find patrol routes for a set of police cars so that the aggregate coverage of all the accident hotspots is maximized. Each patrol car begins and ends its route at a fleet station on a predefined shift. Keskin et al. [10] modelled the MCPRP using a Mixed-integer programming (MIP) formulation and found that state-of-the-art commercial solvers were not always able to find good quality solutions. Hence, a number of heuristic techniques (local and tabu search) were introduced and benchmarked on a range of test problem instances. The test problems solved by Keskin et al. [10] were generated with randomized and real-world data with up to 40 hotspots and 8 patrol cars. These test problem instances were created to reflect the circumstances faced by (and the resources available to) law enforcement agencies in a particular region of the United States.[1] The paper by Keskin et al. [10] reports that the heuristic techniques were able to produce good quality but not optimal solutions to the larger problem instances with 40 hotspots.[2]

The paper published by Çapar et al. [3] showed that significant improvements could be made to the MIP formulation of [10]. With information on the structure of candidate routes in an optimal solution, the authors demonstrated that the number of variables in the formulation of [10] can be reduced. The authors also incorporated a number of bounds constraints which provide additional strength to their reformulation. The enhanced efficiency given through this reformulation of the MCPRP was demonstrated on the benchmark instances introduced in the original paper by Keskin et al. [10].[3]

More recent work by Dewil et al. [7] has shown that the MCPRP can be modelled as a Minimum cost network flow problem (MCNFP).[4] The MCNFP is solvable in

[1] The literature review conducted by Keskin et al. [10] notes that the MCPRP bears similarities to the Team orienteering problem with time windows (TOPTW). However, the distinguishing characteristic of the MCPRP is that the profit associated with visiting a hotspot is not fixed, but is rather a function of the amount of "dwell time" within that hotspot's time window. The authors state that the range of time window lengths used in the problems of their study varied from 1−270 minutes (usually assuming an 8 hour shift).

[2] The results of this study can also be found in the PhD thesis by Li [12].

[3] Çapar et al. [3] considered a number of extensions to the standard MCPRP paradigm. These extensions included the incorporation of shift breaks and allowing the patrol vehicles to begin the shift at different locations, possibly with delayed starting times.

[4] The MCNFP possesses the integrality property when the arc capacities are integer (see Ahuja et al. [1]). This means that the optimal solution is naturally integer if the problem is solved as a linear program.

polynomial time, and thus, the authors correct the claim by Keskin et al. [10] that the MCPRP is NP-hard. The study sets out a time-space network formulation of the problem on which an MCNFP model is defined. The network formulation divides individual hotspots into time sections or segments, which are constructed by considering possible transitions of vehicles which depart from the end of a hotspot (and arrive at another hotspot) or arrive at the beginning of a hotspot (having departed from another hotspot). The MCNFP paradigm also permits the time sections to be weighted differently, thus constituting an extension of the standard MCPRP. The authors demonstrate the superiority of their approach by comparing their computational results with those of Keskin et al. [10]. The MCNFP model is extended by the authors to a Multi-commodity minimum cost network flow problem (MCMCNFP) model which aims to handle overlapping shifts and different start/end locations for the patrol vehicles. In order to test the scalability of the model, the authors state that they could solve a 100 hotspot instance to optimality with up to 23 patrol cars. However, the authors also report that they could not run a 500 hotspot problem instance, even with 3 patrol cars.

Given the limitations on the MCNFP model to solve large-scale problem instances of the MCPRP (as reported by Dewil et al. [7]), our paper aims to investigate the applicability and feasibility of a branch-and-price (column generation with branch-and-bound) approach to the problem. Given that similar approaches have recently proved to be effective at solving closely related patrol routing and scheduling problems, a branch-and-price approach constitutes a natural and promising candidate for solving large-scale instances of the MCPRP.[5]

We begin our study by outlining a process for the construction of a time-space network, which provides an appropriate modelling framework for a path-based linear programming formulation of the MCPRP. A column generation master problem, reduced costs, subproblem, seed column construction and pricing strategies are then subsequently outlined. We then propose a simple branch-and-price paradigm to obtain integer solutions through the incorporation of branching cuts to the master problem. The paper concludes with a presentation and discussion of a number of computational tests performed on a range of benchmark problems, and the results are compared with the MCNFP model of Dewil et al. [7].

## 5.2 Preliminary Notation

The patrol operations network is a directed graph $G_S = (V_S, A_S)$, where $V_S = \{0\} \cup \{1, \ldots, n\}$ is the set of geographical locations and $A_S \subseteq \{(i, j) \mid i, j \in V_S,\ i \neq j\}$ is the set of feasible transitions between the elements of $V_S$. The singleton set $\{0\}$ is used to denote the fleet station, whereas the set $\{1, \ldots, n\}$ represents the number of distinct locations at which hotspots may be found. For each $i \in V_S \backslash \{0\}$ there is a set

[5]For example, see previous work on the Patrol boat scheduling problem with complete coverage (PBSPCC) by the authors of this paper [5] or the PhD thesis by Chircop [4].

of non-overlapping hotspots, where each hotspot is represented by a time window with a start time and an end time. The number of hotspots at location $i$ is given by $h_i$, and the $m$th hotspot at location $i$ is denoted by $(i, [e^i_m, l^i_m])$, where $e^i_m < l^i_m$ for all $m \in \{1, \ldots, h_i\}$ and for all $i \in V_S \backslash \{0\}$. Without loss of generality, at any given location $i \in V_S \backslash \{0\}$, if $m' < m''$, then $l^i_{m'} \leq e^i_{m''}$, where $m', m'' \in \{1, \ldots, h_i\}$. The set of all hotspots is given by $W$ and is indexed by $\ell$. The opening (start time) of hotspot $\ell$ is denoted by $\min(\ell)$ while the close (finish time) of the hotspot is denoted by $\max(\ell)$. The set of hotspots can be expressed as $W := \bigcup_{i \in V_S \backslash \{0\}} W(i)$, where, $W(i)$ is the set of time windows $\{(i, [e^i_1, l^i_1]), \ldots, (i, [e^i_{h_i}, l^i_{h_i}])\}$ at location $i$. We also define a function $\omega$ which maps hotspots to their geographical locations: $\omega : W \to V_S \backslash \{0\}$.

## 5.3 Network Construction

Given $G_S = (V_S, A_S)$, the set of hotspots $W$ and a shift duration $T$, we can construct a time-space network $G_R = (V_R, A_R)$ for the MCPRP according to a transformation $(G_S, W, T) \mapsto G_R$. On this expanded time-space network, we have a set of patrol arcs $A_{\mathrm{P}} \subset A_R$, a set of waiting arcs $A_{\mathrm{W}} \subset A_R$, and a set of transit arcs $A_{\mathrm{T}} \subset A_R$. The set of patrol arcs in time window $\ell \in W$ is expressed as $A_{\mathrm{P}}(\ell) \subseteq A_{\mathrm{P}}$. We define $t_{uv} \in \mathbb{Z}^+$ to be the transit time of traversing arc $(u, v) \in A_R$. For each $v \in V_R$, let $A_+(v)$ be the set of all arcs emanating from $v$, and let $A_-(v)$ be the set of all arcs terminating at $v$. The source and sink vertices (representing the fleet station) are $s$ and $\tau$, respectively. Note that $A_-(s) = A_+(\tau) = \varnothing$. Equipped with the preceding definitions and notation, the time-space network construction begins with an initialization procedure which creates a source and a sink vertex, along with a layer of vertices for each spatial location $V_S \backslash \{0\}$. Each layer will initially contain $T + 1$ vertices, where the horizontal spacing between the vertices defines the time discretization. Hence, each vertex $u \in V_R$ can be expressed in terms of a location-time pair $(i, t)$, where $i \in V_S$ and $t \in \{0, \ldots, T\}$. The initialization procedure can be found in Algorithm 1.

**Algorithm 1** MCPRP: Initialization of a time-space network

**Input**: A spatial network $G_S = (V_S, A_S)$ and a shift length $T$
**procedure** INITIALIZETIMESPACENETWORK($G_S, T$)
    $V_R,\ A_R \leftarrow \varnothing$
    Create source vertex $s = (0, 0)$ and sink vertex $\tau = (0, T)$
    $V_R \leftarrow V_R \cup \{s, \tau\}$
    **for** $i \in V_S \backslash \{0\}$ **do**
        **for** $t = 0, \ldots, T$ **do**
            Create vertex $u$ with $u = (i, t)$
            $V_R \leftarrow V_R \cup \{u\}$
    **return** $G_R = (V_R, A_R)$

Once the initialization procedure has been executed, the next step is to define the hotspots on the time-space network. The hotspots for each location are represented by a series of patrol arcs. The design choice for the hotspots is based on the following insightful theorem from Keskin et al. [10], which is stated below.

**Theorem 5.1** (Keskin et al. [10]) *Let $K^*$ be an optimal solution to an instance of the MCPRP. For each hotspot $\ell \in W$ visited by a patrol vehicle $k \in K^*$, the time at which $k$ finishes patrolling $\ell$ is* $\min\left\{\max(\ell), T - t_{\{\omega(\ell)\}0}\right\}$ *if $\ell$ is the last hotspot visited on $k$'s route, and* $\max(\ell)$ *otherwise.*

Theorem 5.1 states that in an optimal solution to the MCPRP, a given patrol vehicle will remain at hotspot $\ell$ until the close of the time window if hotspot $\ell$ is not the last hotspot on $k$'s route. If, on the other hand, the hotspot $\ell$ is the last hotspot visited on patrol vehicle $k$'s route, then $k$ remains at hotspot $\ell$ until the close of the time window or until the latest time at which $k$ can leave the hotspot and arrive back at the fleet station within the shift $T$. Given this result, we can represent each hotspot with a series of patrol arcs which collectively terminate at the vertex corresponding to the end of the time window or at the vertex corresponding to the latest possible time at which a patrol vehicle must return to the fleet station. The first patrol arc in the series emanates from the vertex corresponding to the start of the time window, with the subsequent patrol arc in the series originating at the next chronological vertex within the time window, and so on. This process is illustrated in Fig. 5.1. The formal procedure for constructing the patrol arcs over the hotspots on a time-space network can be found in Algorithm 2. This procedure also includes a test to check whether a time window lies within, crosses the boundary of or lies outside of the feasibility interval $[t_{0i}, T - t_{i0}]$. Any hotspot which crosses the boundary of the feasibility interval must have its start and end time updated accordingly, while any hotspot lying entirely outside the feasibility window should be discarded.

As we will see in a later section, a set of packing constraints is required in the column generation master problem to avoid multiple contributions to the patrol effort in each hotspot. By adopting the patrol arc construction shown in Fig. 5.1, only one

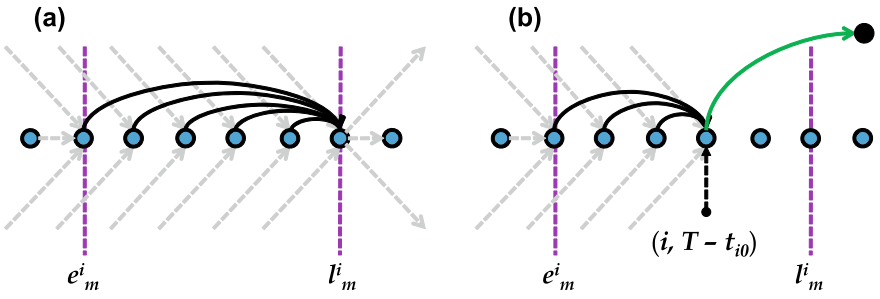


**Fig. 5.1** Part **a** corresponds to a hotspot which lies entirely within the timespan $[t_{0i}, T - t_{i0}]$. Part **b** is indicative of a time window which has a closing time greater than $T - t_{i0}$. The green arc in part **b** is the transit arc which traces back to the fleet station

**Algorithm 2** MCPRP: Patrol arc construction for hotspots

**Input**: A spatial network $G_S = (V_S, A_S)$, a shift length $T$, a set of hotspots $W$
**procedure** ConstructPatrolArcs($G_S$, $W$, $T$)
  $G_R \leftarrow$ InitializeTimeSpaceNetwork($G_S$, $T$)
  $A_P \leftarrow \varnothing$
  **for** $\ell \in W$ **do**
    $A_P(\ell) \leftarrow \varnothing$
    **if** $\left(\max(\ell) < t_{0\{\omega(\ell)\}}\right) \bigvee \left(\min(\ell) > T - t_{\{\omega(\ell)\}0}\right)$ **then**
      **continue**
    **else**
      **if** $\min(\ell) < t_{0\{\omega(\ell)\}}$ **then**
        $\min(\ell) \leftarrow t_{0\{\omega(\ell)\}}$
      **if** $\max(\ell) > T - t_{\{\omega(\ell)\}0}$ **then**
        $\max(\ell) \leftarrow T - t_{\{\omega(\ell)\}0}$
      **for** $t = \min(\ell), \ldots, \max(\ell) - 1$ **do**
        Create arc $(u, v)$ with $u = (\omega(\ell), t)$ and $v = (\omega(\ell), \max(\ell))$
        $A_P \leftarrow A_P \cup \{(u, v)\}$
        $A_P(\ell) \leftarrow A_P(\ell) \cup \{(u, v)\}$
  $A_R \leftarrow A_R \cup A_P$
  **return** $G_R = (V_R, A_R)$

packing constraint is required per hotspot. Without the insight of Theorem 5.1, a naive alternative would be to construct a patrol arc for each time interval in each hotspot, with a corresponding packing constraint for the patrol arcs in the master problem. Such a naive construction would increase the runtime for the solution of both the master problem and subproblem.

Once the hotspots have been identified and constructed on the time-space network, the next step is to create transit arcs between the source vertex and each location. In addition, another set of transit arcs is required to connect each geographical location with the sink vertex. Following the construction of these transit arcs, a set of waiting arcs is required for each layer of vertices in the time-space network. Waiting arcs correspond to dead time, where a patrol vehicle is stationed at a geographical location but is not actively contributing to the patrol effort. First, waiting arcs are constructed between the end of each time window and the start of all subsequent time windows at the same location. Second, a set of waiting arcs is created to connect a location's arrival vertex with the start of each time window at that location. Finally, waiting arcs are constructed which connect the end of each time window at a given location with that location's termination vertex.[6] The procedure is fully described in Algorithm 3.

Following the construction of the primary transit and waiting arcs, we need to account for secondary transit and waiting arcs which correspond to potential move-

[6]The arrival vertex at location $i \in V_S$ is the vertex $v \in V_R$ such that $v = (i, t_{0i})$. The termination vertex at location $i \in V_S$ is the vertex $u \in V_R$ such that $u = (i, T - t_{i0})$.

**Algorithm 3** MCPRP: Primary transit and waiting arc construction

**Input**: A spatial network $G_S = (V_S, A_S)$, a shift length $T$, a set of hotspots $W$
**procedure** ConstructTransitWaitingArcs($G_S$, $W$, $T$)
  $G_R \leftarrow$ ConstructPatrolArcs($G_S$, $W$, $T$)
  $A_{\mathrm{T}},\ A_{\mathrm{W}} \leftarrow \varnothing$
  **for** $i \in V_S \backslash \{0\}$ **do**
    Create arcs $(s, v)$ and $(u, \tau)$ with $v = (i, t_{0i})$ and $u = (i, T - t_{i0})$
    $A_{\mathrm{T}} \leftarrow A_{\mathrm{T}} \cup \{(s, v), (u, \tau)\}$
    **for** $\ell \in W(i)$ **do**
      Create arcs $(v, w)$ and $(x, u)$ with $w = (i, \min(\ell))$ and $x = (i, \max(\ell))$
      $A_{\mathrm{W}} \leftarrow A_{\mathrm{W}} \cup \{(v, w), (x, u)\}$
      **for** $\ell' \in W(i)$ **do**
        **if** $\min(\ell') > \max(\ell)$ **then**
          Create arc $(y, z)$ with $y = (i, \max(\ell))$ and $z = (i, \min(\ell'))$
          $A_{\mathrm{W}} \leftarrow A_{\mathrm{W}} \cup \{(y, z)\}$
  $A_R \leftarrow A_R \cup A_{\mathrm{T}} \cup A_{\mathrm{W}}$
  **return** $G_R = (V_R, A_R)$

**Algorithm 4** MCPRP: Secondary transit and waiting arc construction

**Input**: A spatial network $G_S = (V_S, A_S)$, a shift length $T$, a set of hotspots $W$
**procedure** ConnectLocations($G_S$, $W$, $T$)
  $G_R \leftarrow$ ConstructTransitWaitingArcs($G_S$, $W$, $T$)
  **for** $i \in V_S \backslash \{0\}$ **do**
    **for** $\ell \in W(i)$ **do**
      **for** $j \in V_S \backslash \{0, i\}$ such that $b_{ij} = 1$ **do**
        **if** $\max(\ell) + t_{ij} \leq T - t_{j0}$ **then**
          Create arc $(u, v)$ with $u = (i, \max(\ell))$ and $v = (j, \max(\ell) + t_{ij})$
          $A_{\mathrm{T}} \leftarrow A_{\mathrm{T}} \cup \{(u, v)\}$
          **if** $\exists\, \ell' \in W(j)$ such that $\min(\ell') \leq \max(\ell) + t_{ij} \leq \max(\ell')$ **then**
            **continue**
          **else**
            **for** $\ell' \in W(j)$ **do**
              **if** $\min(\ell') > \max(\ell) + t_{ij}$ **then**
                Create arc $(v, w)$ with $w = (j, \min(\ell'))$
                $A_{\mathrm{W}} \leftarrow A_{\mathrm{W}} \cup \{(v, w)\}$
  $A_R \leftarrow A_R \cup A_{\mathrm{T}} \cup A_{\mathrm{W}}$
  **return** $G_R = (V_R, A_R)$

ments of patrol vehicles between different geographical locations. For the end of each time window $\max(\ell)$ at a given geographical location $i \in V_S \backslash \{0\}$, a set of transit arcs is created to link location $i$ with all other locations $j \in V_S \backslash \{0, i\}$ for which a feasible transit lane exists, that is, for all $j$ such that $b_{ij} = 1$ and $\max(\ell) + t_{ij} \leq T - t_{j0}$,

**Algorithm 5** MCPRP: Post processing of the time-space network

**Input**: A spatial network $G_S = (V_S, A_S)$, a shift length $T$, a set of hotspots $W$
**procedure** POSTPROCESSING($G_S$, $W$, $T$)
  $G_R \leftarrow$ CONNECTLOCATIONS($G_S$, $W$, $T$)
  $V_{\text{temp}}$, $A_{\text{temp}} \leftarrow \varnothing$
  **for** $(u, v) \in A_{\text{P}}$ **do**
    **if** $(A_-(u) = \varnothing) \wedge (u \neq s)$ **then**
      $A_{\text{temp}} \leftarrow A_{\text{temp}} \cup \{(u, v)\}$
    **else**
      **continue**
  **for** $u \in V_R \backslash \{s, \tau\}$ **do**
    **if** $A_-(u) = \varnothing$ **then**
      $V_{\text{temp}} \leftarrow V_{\text{temp}} \cup \{u\}$
    **else**
      **continue**
  $V_R \leftarrow V_R \backslash V_{\text{temp}}$, $A_R \leftarrow A_R \backslash A_{\text{temp}}$, $A_{\text{P}} \leftarrow A_{\text{P}} \backslash A_{\text{temp}}$
  **return** $G_R = (V_R, A_R)$

where $(b_{ij})_{i,j=1,\ldots,n}$ is the adjacency matrix of the spatial network. For each feasible transit arc constructed between $u = (i, \max(\ell))$ and $v = (j, \max(\ell) + t_{ij})$, we need to check if there exists an $\ell' \in W(j)$ such that $\min(\ell') \leq \max(\ell) + t_{ij} \leq \max(\ell')$. If this condition is satisfied, then the connecting transit arc between $i$ and $j$ hits a hotspot at location $j$, and no further work is required. However, if the condition is not satisfied, then a series of waiting arcs are required to connect node $v = (j, \max(\ell) + t_{ij})$ with each node $w = (j, \min(\ell'))$ such that $\ell' \in W(j)$ and $\min(\ell') > \max(\ell) + t_{ij}$. The procedure to construct the secondary transit and waiting arcs is formally outlined in Algorithm 4.

The final stage of the time-space network construction is a post-processing phase. At this final stage, any vertices in the time-space network which do not contain any incoming arcs are deleted. In other words, a vertex $u \in V_R \backslash \{s, \tau\}$ will be deleted if $A_-(u) = \varnothing$. In addition, all arcs $(u, v) \in A_+(u)$ which proceed from such a vertex must be deleted from the network. These arcs, if they exist, will be patrol arcs, since all waiting and transit arcs are constructed from vertices with non-empty in-arc sets. The post-processing phase is illustrated in Fig. 5.2 and formally summarized in pseudocode in Algorithm 5.

## 5.4 Master Problem

The time-space network $G_R = (V_R, A_R)$ constructed according to the procedures of the previous section can be used to formulate the MCPRP as a linear program, suitable for the application of column generation. Let $P$ be the set of all feasible paths through

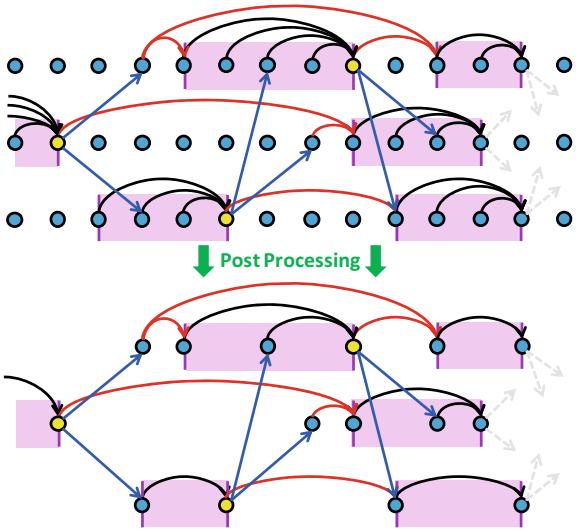


**Fig. 5.2** (TOP) Construction of secondary transit and waiting arcs on the time-space network. The yellow vertices correspond to the end times of hotspots from which transit (blue) arcs are constructed. If a transit arc emanating from a yellow vertex does not land within a hotspot at another location, waiting (red) arcs are constructed from the target vertex to the opening of all subsequent hotspots. Additional waiting arcs are constructed between the closing of a hotspot and the opening of the following hotspot at the same geographical location. (BOTTOM) The time-space network after the post-processing phase (deletion of superfluous arcs and vertices). A vertex, with or without an outgoing patrol (black) arc, is deleted if there are no arcs pointing to it

the network $G_R$ from $s$ to $\tau$. For each $p \in P$, let $x_{uvp} = 1$ if path $p$ uses arc $(u, v) \in A_R$ and $x_{uvp} = 0$ otherwise. As the time-space network $G_R$ contains no cycles, we can express the integral flow $x_{uv} \in \mathbb{Z}^+$ over an arc $(u, v) \in A_R$ in terms of path variables $\lambda_p \in \{0, 1\}$ for all $p \in P$, where $\lambda_p = 1$ if path $p$ is used and $\lambda_p = 0$ otherwise. Hence, we can write $x_{uv} = \sum_{p \in P} x_{uvp} \lambda_p$. Denote $\bar{c}_p = \sum_{(u,v) \in A_P} t_{uv} x_{uvp}$ to be the total time spent on patrol for path $p \in P$, and $\bar{c}_{pi} = \sum_{\ell_i \in W} \sum_{(u,v) \in A_P(\ell_i)} t_{uv} x_{uvp}$ to be the time path $p \in P$ spends on patrol in location $i \in V_S \backslash \{0\}$, where $\ell_i \in \{\ell \in W \mid \omega(\ell) = i\}$. Moreover, let $a_{\ell p} = \sum_{(u,v) \in A_P(\ell)} x_{uvp}$ for all $p \in P$ and $\ell \in W$, so that $a_{\ell p} = 1$ if $p \in P$ patrols hotspot $\ell \in W$ and $a_{\ell p} = 0$ otherwise. By relaxing the integrality constraints on the path variables, that is, setting $\lambda_p \geq 0$ for all $p \in P$, we can formulate a master linear programming problem for the MCPRP as follows:

$$\text{maximize} \quad \sum_{p\in P} \bar{c}_p \lambda_p, \tag{5.1}$$

$$\text{subject to} \quad \sum_{p\in P} a_{\ell p} \lambda_p \leq 1, \qquad \forall \ell \in W, \qquad [\pi_\ell] \tag{5.2}$$

$$\sum_{p\in P} \bar{c}_p \lambda_p \leq \bar{C}, \qquad [\alpha] \tag{5.3}$$

$$\sum_{p\in P} \bar{c}_{pi} \lambda_p \leq \bar{C}_i, \qquad \forall i \in V_S \backslash \{0\}, \qquad [\beta_i] \tag{5.4}$$

$$\sum_{p\in P} \lambda_p \leq \kappa_{\max}, \qquad [\gamma] \tag{5.5}$$

$$\lambda_p \geq 0, \qquad \forall p \in P. \tag{5.6}$$

The objective function (5.1) seeks to maximize the aggregate time spent on hotspot patrol. The statement of the objective is followed by a series of packing constraints given through (5.2). There is a single packing constraint for each hotspot $\ell \in W$. The packing constraints are included to ensure that each hotspot is patrolled by at most one vehicle, thereby prohibiting multiple contributions to a single hotspot. The constraints (5.3) and (5.4) are optional bounds constraints for the formulation. Constraint (5.3) stipulates an upper bound $\bar{C}$ on the aggregate patrol time delivered by the vehicles across all hotspots on the network. The constraints given by (5.4) use a set of values $\{\bar{C}_i \mid i \in V_S \backslash \{0\}\}$ to enforce upper bounds on the aggregate patrol effort delivered to all the hotspots at each geographical location. The upper bounds $C$ and $\bar{C}_i$ are given through (5.7) below. Constraint (5.5) provides an upper bound on the number of available patrol vehicles through $\kappa_{\max}$. Finally, the constraints (5.6) enforce non-negativity conditions on the flow of each vehicle over each arc in the time-space network.

$$\bar{C} := \sum_{i=1}^{n} \sum_{m=1}^{h_i} \left( l_m^i - e_m^i \right), \qquad \bar{C}_i := \sum_{m=1}^{h_i} \left( l_m^i - e_m^i \right), \quad \forall i \in V_S \backslash \{0\}. \tag{5.7}$$

If the set $P$ of feasible paths through the time-space network is large, then it may not be practicable to write out the full master problem (5.1)–(5.6). In such cases, the problem can be initialized with a subset of paths $P' \subset P$ to form an initial Restricted master problem (RMP). Columns representing paths of negative reduced cost can then be added to the RMP in an iterative fashion by feeding the dual

variables[7] of the RMP to a column generation subproblem.[8] A candidate initialization procedure for the RMP of the MCPRP is proffered in Sect. 5.6, while the nature of the column generation subproblem is discussed in Sect. 5.5.

## 5.5 Reduced Costs and Subproblem

Paths of negative reduced cost are determined by solving a pricing subproblem over the underlying time-space network $G_R$. Uncovering the algebraic form of the reduced cost of a path $p \in P$ naturally leads to revealing the structure of the pricing subproblem. First, we can construct some useful surjective mappings in order to define the reduced cost of a path: $\phi : A_P \to W'$ and $\psi : A_P \to V_S \backslash \{0\}$. The set $W' = \{1, \ldots, |W|\}$ is an integer-valued index set corresponding to the hotspots on the network. Thus, for each patrol arc, $\phi$ maps to the hotspot index, while $\psi$ maps to the geographical location. The reduced cost $\bar{r}_p$ of a path $p \in P$ through the time-space network $G_R$ is given through $\bar{r}_p := \boldsymbol{\upsilon}^T \mathbf{A}_p - \bar{c}_p$, where $\boldsymbol{\upsilon}^T$ is a row vector of the dual variables of the master problem (5.1)–(5.6), $\mathbf{A}_p$ is the column corresponding to variable $\lambda_p$, and $\bar{c}_p$ is the cost coefficient of $\lambda_p$ in the objective function (5.1). Therefore, the reduced cost $\bar{r}_p$ of a path $p \in P$ through the time-space network $G_R$ can be written in terms of the underlying arc variables as follows:

$$\bar{r}_p = \boldsymbol{\upsilon}^T \mathbf{A}_p - \bar{c}_p, \tag{5.8}$$

$$= \left( \sum_{\ell \in W} \pi_\ell a_{\ell p} \right) + \alpha \bar{c}_p + \beta_i \bar{c}_{pi} + \gamma - \bar{c}_p, \tag{5.9}$$

$$= \left( \sum_{\ell \in W} \pi_\ell \sum_{(u,v) \in A_P(\ell)} x_{uvp} \right) + \alpha \left( \sum_{(u,v) \in A_P} t_{uv} x_{uvp} \right) + \left( \sum_{(u,v) \in A_P} \beta_{\psi(u,v)} t_{uv} x_{uvp} \right) \tag{5.10}$$

$$+ \gamma \left( \sum_{(s,v) \in A_+(s)} x_{svp} \right) - \left( \sum_{(u,v) \in A_P} t_{uv} x_{uvp} \right),$$

$$= \left( \sum_{(u,v) \in A_P} \left[ \pi_{\phi(u,v)} + \left( \alpha + \beta_{\psi(u,v)} - 1 \right) t_{uv} \right] x_{uvp} \right) + \left( \sum_{(s,v) \in A_+(s)} \gamma \, x_{svp} \right). \tag{5.11}$$

[7] The dual variables of the RMP can be found in square parentheses along the right-hand side of (5.1)–(5.6).

[8] This essentially describes the column generation technique for solving prohibitively large linear programs (see [6] for a comprehensive introduction). The fundamental insight of the column generation approach is to generate the columns of the constraint matrix on-the-fly by recourse to an optimization subproblem. The idea was originally suggested by Ford and Fulkerson [8], but was first implemented by Gilmore and Gomory [9].

Hence, the reduced cost of path $p \in P$ can be expressed as $\bar{r}_p = \sum_{(u,v)\in p} \mu_{uv} x_{uv}$, where the coefficients $\mu_{uv}$ are given through: $\mu_{uv} = \pi_{\phi(u,v)} + (\alpha + \beta_{\psi(u,v)} - 1)t_{uv}$ if $(u, v) \in A_{\mathrm{P}}$, $\mu_{uv} = \gamma$ if $(u, v) \in A_+(s)$ and $\mu_{uv} = 0$ otherwise. Therefore, the pricing subproblem can be written as a shortest path problem over $G_R$ as follows:

$$\text{minimize} \quad \sum_{(u,v)\in A_R} \mu_{uv} x_{uv}, \tag{5.12}$$

$$\text{subject to} \quad \sum_{(s,v)\in A_+(s)} x_{sv} = 1, \tag{5.13}$$

$$\sum_{(u,v)\in A_-(v)} x_{uv} = \sum_{(v,w)\in A_+(v)} x_{vw}, \quad \forall v \in V_R \backslash \{s, \tau\}, \tag{5.14}$$

$$\sum_{(u,\tau)\in A_-(\tau)} x_{u\tau} = 1, \tag{5.15}$$

$$x_{uv} \in \{0, 1\}, \quad \forall (u, v) \in A_R. \tag{5.16}$$

Given that $G_R$ is a directed acyclic graph, the shortest path problem given through (5.12)–(5.16) can be solved by first applying the dual costs from the RMP and then performing edge relaxation over a topologically sorted list of the vertices in $G_R$. This procedure is given through DAG-SP$(G_R, \mu, s, \tau)$, which outputs a shortest path $p$ through $G_R$ from $s$ to $\tau$ assuming the cost structure $\mu$ and the associated path cost $\delta_p(s, \tau)$. The entire shortest path procedure is summarized in Algorithm 6.

**Algorithm 6** MCPRP: Dual based shortest path through a time-space network

**Input**: A time-space network $G_R = (V_R, A_R)$ with source $s \in V_R$ and sink $\tau \in V_R$, vector of dual variables $\boldsymbol{\upsilon}$
**procedure** MCPRP_DUALSHORTESTPATH($G_R, s, \tau, \boldsymbol{\upsilon}$)
  **for** $(u, v) \in A_{\mathrm{P}}$ **do**
    $\mu_{uv} \leftarrow \pi_{\phi(u,v)} + \left(\alpha + \beta_{\psi(u,v)} - 1\right) t_{uv}$
  **for** $(u, v) \in A_+(s)$ **do**
    $\mu_{uv} \leftarrow \gamma$
  **for** $(u, v) \in A_R \backslash (A_{\mathrm{P}} \cup A_+(s))$ **do**
    $\mu_{uv} \leftarrow 0$
  $(p, \delta_p(s, \tau)) \leftarrow$ DAG- SP$(G_R, \mu, s, \tau)$
  $\bar{r}_p^* \leftarrow \delta_p(s, \tau)$
  **return** $(p, \bar{r}_p^*)$

## 5.6 Seed Columns

The path-based linear programming formulation of the MCPRP is of set packing type, and thus, the problem of constructing a feasible initial primal basis is not onerous (since the patrol coverage constraints are non-binding). We have chosen to adopt a Randomized shortest path heuristic (RSPH) to generate a set of candidate paths for the initialization of the RMP. The random cost structure derived from the RSPH is intended to produce an initial set of paths which share the patrol coverage effort as evenly as possible. This was preferred to a straightforward application of a greedy shortest path heuristic in which the hotspots are equally weighted. For large fleet sizes relative to the number of hotspots on the network, the application of a straightforward greedy heuristic would most likely produce an initial basis consisting of both good and bad quality columns. This would be an undesirable outcome compared to an initial basis in which the patrol effort is more evenly distributed. Hence, the RSPH was implemented in an attempt to increase the likelihood of producing initial basis sets of better quality.

The RSPH first applies a cost $\mu_{ij} = -T$ to each patrol arc $(i, j) \in A_{\mathrm{P}}$ and cost $\mu_{ij} = 0$ to each arc $(i, j) \in A_R \backslash A_{\mathrm{P}}$. For each hotspot $\ell \in W$ in the time-space network, a patrol arc $(i, j)$ is chosen at random from the set $A_{\mathrm{P}}(\ell)$. The time index $t$ at the tail of arc $(i, j)$, denoted by $(i, j)_t$, is then multiplied by a random number $\tilde{r}$ drawn from the uniform probability distribution U(0, 1). This value is then negated and added to the cost of each patrol arc in the hotspot. Once the patrol arc costs have been updated in this manner, a shortest path is invoked over the time-space network. By examining the returned shortest path $p$, we can determine all arcs $(i, j) \in p$ such that $(i, j) \in A_{\mathrm{P}}$. We then update the costs of all $(u, v) \in A_{\mathrm{P}}$ such that $\phi(u, v) = \phi(i, j)$ according to $\mu_{uv} = T$. On the other hand, if $(i, j) \in p$ and $(i, j) \in A_R \backslash A_{\mathrm{P}}$, then the arc cost is updated through $\mu_{ij} = 0$. A new shortest path is subsequently returned from the network with the updated cost structure, and the heuristic continues in a cyclical manner, terminating when the aforementioned procedure has been called $\kappa_{\max}$ times. The procedure is formally outlined in Algorithm 7.

## 5.7 Pricing Out Candidate Paths

The column generation procedure is initialized by running the randomized construction heuristic presented in Algorithm 7 over the time-space network. Starting from the initial basis produced by the construction heuristic, columns are generated one-at-a-time by solving the pricing subproblem using the dual costs from the current iteration of the RMP. As long as paths of negative reduced cost are returned from the pricing subproblem, the procedure continues on in a cyclical fashion and the paths are added as columns/variables to the RMP. The column generation procedure

**Algorithm 7** MCPRP: Randomized shortest path heuristic for column generation initialization

**Input**: A time-space network $G_R = (V_R, A_R)$ with source $s \in V_R$ and sink $\tau \in V_R$
**procedure** RSPH($G_R, s, \tau$)
  **for** $(u, v) \in A_R \backslash A_P$ **do**
    $\mu_{uv} \leftarrow 0$
  **for** $\ell \in W$ **do**
    Randomly select an arc $(i, j) \in A_P(\ell)$
    **for** $(u, v) \in A_P(\ell)$ **do**
      $\mu_{uv} \leftarrow -T - \tilde{r}(0, 1) \times (i, j)_t$
  $P' \leftarrow \varnothing$
  **for** $k = 1, \ldots, \kappa_{\max}$ **do**
    $(p, \delta_p(s, \tau)) \leftarrow$ DAG- SP($G_R, \mu, s, \tau$)
    $P' \leftarrow P' \cup \{p\}$
    **for** $(i, j) \in p$ **do**
      **if** $(i, j) \in A_P$ **then**
        **for** $(u, v) \in A_P$ such that $\phi(u, v) = \phi(i, j)$ **do**
          $\mu_{uv} \leftarrow T$
  **return** $P'$

terminates once the reduced cost of a path returned from the pricing subproblem is non-negative.[9] A straightforward implementation of the procedure is summarized in Algorithm 8.

## 5.8 Branch-and-Price

As column generation is directly applicable to real variable problems, it can be embedded within a branch-and-bound tree structure in order to solve large-scale integer programming problems. This augmented application of column generation is known as branch-and-price (see [2]). In this section, we propose a straightforward branch-and-price approach to the MCPRP on the time-space network outlined heretofore. If the application of column generation at the root node fails to return an integer solution, we can impose branching cuts to the RMP with respect to the most

[9]By solving the RMP as a linear program and obtaining its dual variables, a subproblem can be solved to determine a new column (variable) to add to the RMP. The subproblem can accomplish this by casting the pricing step of the simplex algorithm (find a variable with negative reduced cost to enter the basis) as an optimization problem. The process iterates between the RMP and the subproblem, terminating when no variable can price-out favourably.

**Algorithm 8** MCPRP: Column generation procedure

1: **Input**: A time-space network $G_R = (V_R, A_R)$ with source $s \in V_R$ and sink $\tau \in V_R$
2: $P' \leftarrow \text{RSPH}(G_R, s, \tau)$
3: **procedure** MCPRP_GENERATECOLUMNS($G_R, s, \tau$)
4: Construct RMP from $P'$ with associated variables $\{\lambda_p \mid p \in P'\}$
5: Solve the RMP to get dual variables $\boldsymbol{\upsilon}$
6: $(p, \bar{r}_p^*) \leftarrow$ MCPRP_DUALSHORTESTPATH($G_R, s, \tau, \boldsymbol{\upsilon}$)
7: **if** $\bar{r}_p^* \geq 0$ **then**
8: **break**
9: **else**
10: Add new variable $\lambda_p$ and associated column to RMP
11: $P' \leftarrow P' \cup \{p\}$
12: **goto** 4

fractional transit arc in the time-space network.[10] The procedure works as follows. Assume we have a fractional (non-integral) solution to the MCPRP with a basis defined by a set of paths $P' \subset P$. Let $(u, v) \in A_T$. The flow $F$ over the arc $(u, v)$ is given by the sum of the path variables in the current basic solution which use that arc, that is, $F(u, v) := \sum_{p \in \{q \in P' \mid (u,v) \in q\}} \lambda_p$. The most fractional transit arc in the current non-integral solution is therefore given by $(u, v)^* := \arg\min_{(u,v) \in A_T} \tilde{F}(u, v)$, where we have:

$$\tilde{F}(u, v) := \begin{cases} \frac{1}{2} - (F(u, v) - \lfloor F(u, v) \rfloor) & \text{if } F(u, v) - \lfloor F(u, v) \rfloor < \frac{1}{2}, \\ \frac{1}{2} - (\lceil F(u, v) \rceil - F(u, v)) & \text{otherwise.} \end{cases} \tag{5.17}$$

Once the most fractional transit arc has been identified, we can create left and right disjunctive branches under the current fractional solution (a node in the exploratory tree). Therefore, new restricted master problems are required for each branching decision on the left and right. A new RMP created from a branching decision inherits the form of its antecedent tree node with the addition of the following constraint (cut):

$$\sum_{p \in \{q \in P' \mid (u,v)^* \in q\}} \lambda_p \leq \lfloor F((u, v)^*) \rfloor \quad \text{(LEFT)}, \tag{5.18}$$

$$\sum_{p \in \{q \in P' \mid (u,v)^* \in q\}} \lambda_p \geq \lceil F((u, v)^*) \rceil \quad \text{(RIGHT)}. \tag{5.19}$$

[10] An investigation of alternative branching strategies, for example, selecting various combinations of arc variables at a time, is beyond the scope of this paper, but is recommended for future research.

Note that the incorporation of a branching cut (5.18)/(5.19) to the master problem requires that the subproblem's cost structure be modified by adding a dual penalty cost to the arc $(u, v)^*$.

Branching cuts are added to various fractional nodes in the tree in order to find an integer solution to the full problem. The search tree maintains a best (relaxed) upper bound $z_{\text{UB}}$ and a best (integer) lower bound $z_{\text{LB}}$. Given a list of unfathomed nodes, we select the node with objective $z'$ such that $z_{\text{UB}} - z'$ is a minimum. Ties between nodes with the same objective can be broken by preferring the node with the greatest ratio of integral non-zero arc variables to the number of non-zero arc variables. In the event that this ratio is unity, we have an integral solution over the arc variables of the underlying network. However, integrality of the network arc variables is not a sufficient condition for the integrality of the path variables. There may be tree nodes for which a solution has $x_{uv} \in \mathbb{Z}^+$ for all $(u, v) \in A_R$, where $\lambda_p \notin \{0, 1\}$ for some $p \in P'$.[11] Therefore, integrality of the path variables must be checked at each node in order to obtain a feasible integer solution. Any nodes which are arc integral but not path integral are pruned from the tree. When a new integer solution has been found, it can be checked against the current best integer lower bound $z_{\text{LB}}$. If the new integer solution is better than the current best lower bound, the lower bound is updated. Otherwise, the newly found integer solution can be pruned from the tree. When the gap between the best relaxed upper bound and the best integer lower bound is closed, and if complementary slackness and feasibility conditions are satisfied, an optimal solution $z^*$ has been found, and the branch-and-price procedure can be terminated.

## 5.9 Computational Results

### *5.9.1 Results on Sample Test Problems*

In order to benchmark our proposed branch-and-price approach to the MCPRP, we randomly generated a set of 40 geographical networks to be used as the basis for the design of a broad range of test problem instances. Two grid sizes were used to generate the test networks: $30 \times 30$ min and $60 \times 60$ min.[12] Given a grid structure, the test problems were generated by randomly placing hotspot locations within the grid, and then assigning time windows of various lengths to each location.[13]

[11] Vanderbeck [13] notes that this phenomenon (i.e. fractional path flows translating to integral arc flows) can occur when the subproblem is a shortest path problem, which is precisely our case. To see why this is so, recall that $x_{uv} = \sum_{p \in P} x_{uvp} \lambda_p$ for all $(u, v) \in A_R$. It is straightforward to see that if $\lambda_p \in \{0, 1\}$ for all $p \in P$, then $x_{uv} \in \mathbb{Z}^+$ for all $(u, v) \in A_R$, since $x_{uvp} \in \{0, 1\}$ for each $(u, v) \in A_R$ and $p \in P$. However, the converse does not hold. That is, it is mathematically possible to find a set of paths taking fractional values which translates into an integral arc flow solution.

[12] For a grid structure of size $30 \times 30$ min, a patrol vehicle will traverse the breadth/length of the structure in 30 min.

[13] In total, we generated 8 networks with 40 hotspots, 8 networks with 60 hotspots, 16 networks with 80 hotspots and 8 networks with 100 hotspots. Each test network was randomly generated over

In total, 478 separate test problem instances were run, where the number of cars ranged from 2 to at most 35.[14] The branch-and-price approach was always able to find a provably optimal integer solution. The identification of optimality came through the observation that the objective function value at the root node (which is an upper bound on the objective of the optimal integer solution) always matched the objective function value of the integer solution found in each test problem instance. Given the observed absence of an intergrality gap, we can conclude that the time-space network construction of the subproblem is obviously a strong formulation for the MCPRP.

A general observation is that when the vehicle flow on the underlying time-space network was small, that is, when the fleet size was small compared to the number of hotspots to be covered, the branch-and-price approach consistently produced integer root node solutions. The amount of branching required generally increased as the number of vehicles increased, as did the CPU runtime. Another general trend is that for an equivalent number of patrol vehicles, small grid sizes with short hotspot durations were harder to solve (in terms of the number of branching decisions and the CPU runtime) than larger grid sizes with long hotspot durations. This can be attributed to the increased number of routing choices for instances with short hotspot durations and shorter travel times between hotspots (especially given that the shift duration was constant across the entire problem space).

The test problem instances for the 80 hotspot network included 8 test problems containing 80 locations with a single hotspot affixed to each location and another 8 test problems with 40 locations and 2 hotspots for each location. The general runtime trend was better for the second set of instances (that is, the ones with two hotspots per location). Again, this can be attributed to the increased number of routing choices incurred with an increased number of locations on the network grid. This highlights the importance of distinguishing the number of locations from the number of hotspots on the network. The results for the 80 hotspot category suggest that this distinction is non-trivial.

Finally, we observed that the RSPCH was able to solve the 1 min time window problem instances at the root node, that is, no additional columns needed to be generated and no branching was required, even for instances with fleet sizes yielding complete patrol coverage. We note that a problem instance of the MCPRP with 1 min time windows constitutes a special rendering of the Team Orienteering Problem (TOP). This special case is called the TOPTW, in which the profit is 1 for each vertex visited, but with additional constraints imposing strict visiting times for profit collection at the vertices (see [14]).

---

a grid structure of size $30 \times 30$ min or $60 \times 60$ min. The time window length of the hotspots was either 1 min, 5−15 min, 30 min or 30−90 min. All test problem instances were run with an 8 hour shift (that is, 480 min) and a time discretization of 1 min.

[14]The computational results of all test problem instances can be found in Appendix G of the PhD thesis by Chircop [4].

### 5.9.2 Benchmarking Against the MCNFP Model of Dewil et al. [7]

In order to benchmark and validate our branch-and-price approach to the MCPRP, we compared its performance on the test problem instances of the previous section against the MCNFP model presented in [7]. The MCNFP model was implemented with the network simplex algorithm from the open-source LEMON C++ libraries (see [11]). For each of the test problems, the MCNFP model produced the same optimal objective as the branch-and-price approach. This provides a strong validation for the correctness of the time-space network construct outlined in this article.[15] The MCNFP model demonstrated superior runtime performance to the branch-and-price approach on almost all of the test problem instances. The performance differential became more apparent when the number of patrol cars started to saturate the underlying time-space network. These particular instances required a considerable amount of branching with the branch-and-price approach, and hence, the runtime increased with the number of patrol cars. This demonstrates that the MCNFP formulation of the MCPRP is still the gold standard for networks of the scale tested here.

### 5.9.3 Large-Scale Problem Instances

In addition to the small- to medium-scale networks of the previous section, we also designed two large-scale problems for which the branch-and-price approach could outperform the MCNFP model over a broad range of fleet sizes. The first large scale network consisted of 200 hotspots over a $100 \times 100$ min grid, with the time window lengths ranging between 30 and 90 min. The second network, designed over the same grid size, contained 250 hotspots, with the time window lengths ranging between 30 and 60 min. For both networks, we considered fleet sizes from 2 to 25 patrol cars. On these problem instances, the branch-and-price approach was able to outperform the MCNFP model, with the exception of a small number of cases. Figure 5.3 shows the results for the 200 hotspot case, while Fig. 5.4 contains the results for the 250 hotspot case. These problem instances correspond to situations in which the underlying time-space network is unsaturated with patrol cars, and so many of the integer solutions found with the branch-and-price approach solve at the root node in a shorter time frame than the MCNFP model. However, when we increased the fleet size beyond 25 patrol cars, more intensive branching was required

[15] The computational results for the branch-and-price approach to the MCPRP were produced on a 2.70 GHz dual-core processor on a 32-bit operating system with 4.00 GB of RAM. All primal and dual solutions to the linear programs were obtained with CPLEX 12.6. The column generation and shortest path algorithms, along with the required data structures for the master problem and the time-space network, were coded using the Java programming language and the Eclipse Integrated development environment (IDE). The MCNFP model was run from an executable (on the same operating system) compiled from source code supplied by the authors of [7] and the LEMON C++ libraries using the Microsoft Visual Studio (2012) IDE.

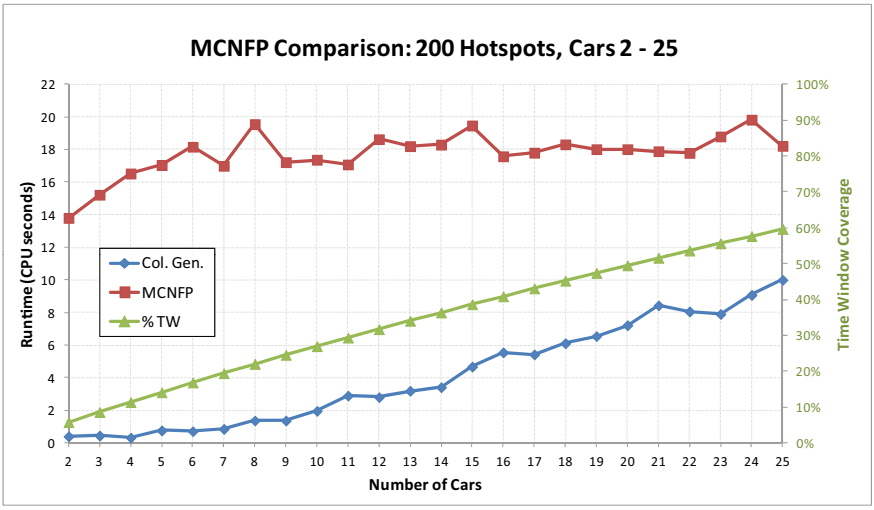


**Fig. 5.3** Comparison of the runtime performance of the MCNFP model with the branch-and-price approach on a 200 hotspot test problem. The time windows range between 30 and 90 min, the fleet size ranges from 2 to 25 patrol cars and the amount of time window coverage is shown on the right-hand vertical axis

for the branch-and-price approach, and the MCNFP model began to exhibit better runtime performance. In fact, we observed that the solution time for the MCNFP was independent of the number of patrol cars, in contrast to the branch-and-price approach.

Finally, we generated some further test problem instances on two large-scale networks which could not be solved by the MCNFP model of [7], but which could be easily handled by the branch-and-price approach.[16] These test problems are the largest instances of the MCPRP solved to-date. The size of the network grid structure was $100 \times 100$ min for both networks. The first network ($H300\_a$) contained 300 spatial locations and 300 hotspots, with time window lengths in the range 30−90 min. The second network ($H500\_a$) contained 250 locations with 500 hotspots of length 30 min, with two hotspots allotted to each location on the network grid.[17] The results are summarized in Table 5.1 for problem instances with 2−25 patrol vehicles.[18] The majority of these problem instances were solved at the root node (no branching

[16]The MCNFP model crashed (due to memory capacity constraints) when we attempted to solve these large-scale instances.

[17]The aggregate duration of all the hotspots in $H300\_a$ was 17,606 min. The aggregate duration for $H500\_a$ was 15,000 min. The shift duration in both cases was 8 hours (480 min).

[18]The headings used in Table 5.1 are as follows: Car—the number of vehicles. R.Obj.—the objective function value at the root node. R.Ti.—the CPU time (seconds) at the root node. R.Col.—the number of columns generated at the root node. No.—the number of nodes explored (fathomed) in the branch-and-price tree. S.Ti.—the CPU time (seconds) taken to find the integer solution. Hot.—the number of hotspots visited.

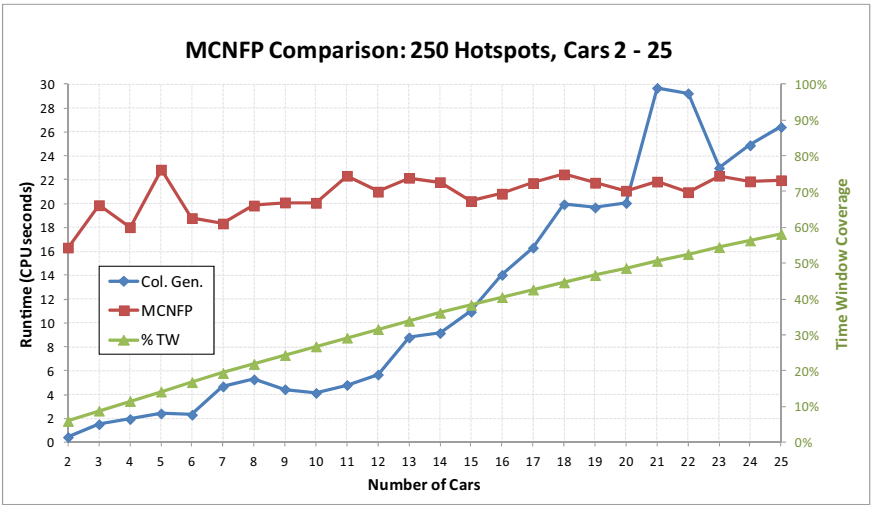


**Fig. 5.4** Comparison of the runtime performance of the MCNFP model with the branch-and-price approach on a 250 hotspot test problem. The time windows range between 30 and 60 min, the fleet size ranges from 2 to 25 patrol cars and the amount of time window coverage is shown on the right-hand vertical axis

required). However, given the extremely large size of the underlying time-space networks, the column generation process was much slower than the previously tested networks. For example, when $H500\_a$ was solved with 25 cars, only two branching decisions were required to find an integer solution, but the runtime was approximately 5 min. We again observed that more intensive branching was required as the number of cars began to saturate each network. For example, we tested $H500\_a$ with 48 vehicles, which took approximately 40 min to solve with 22 branching decisions required.

## 5.10 Conclusions and Future Work

This paper has introduced a branch-and-price framework, underpinned by a specially tailored time-space network, for obtaining solutions to the MCPRP. We introduced a number of test problems for benchmarking, consisting of different numbers of hotspots, time window durations and grid sizes. These test problems were used to validate our approach against the MCNFP model of [7]. While the MCNFP model outperforms the runtime efficiency of the branch-and-price approach on small- to medium-scale problems, it was shown that on large-scale problem instances with certain fleet sizes, the branch-and-price approach can outperform the MCNFP model. We also introduced two large-scale problems, one with 300 hotspots and another with 500 hotspots, which could not be solved by the MCNFP model, but which could be

**Table 5.1** Summary of computational results for two large-scale networks

| Car | Test problem $H300\_a$ | | | | | | Test problem $H500\_a$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R.Obj. | R.Ti. | R.Col. | No. | S.Ti. | Hot. | R.Obj. | R.Ti. | R.Col. | No. | S.Ti. | Hot. |
| 2 | 735 | 0.6 | 7 | 0 | 0.6 | 13 | 676 | 2.2 | 16 | 0 | 2.2 | 27 |
| 3 | 1,085 | 1.3 | 12 | 0 | 1.3 | 19 | 995 | 2.6 | 20 | 0 | 2.6 | 41 |
| 4 | 1,434 | 2.2 | 23 | 0 | 2.2 | 26 | 1,312 | 3.6 | 28 | 0 | 3.6 | 52 |
| 5 | 1,767 | 3.2 | 33 | 0 | 3.2 | 32 | 1,623 | 7.5 | 54 | 0 | 7.5 | 63 |
| 6 | 2,096 | 4.5 | 48 | 0 | 4.5 | 38 | 1,930 | 10.1 | 74 | 0 | 10.1 | 76 |
| 7 | 2,423 | 5.2 | 56 | 0 | 5.2 | 45 | 2,234 | 13.6 | 100 | 0 | 13.6 | 87 |
| 8 | 2,745 | 5.7 | 56 | 2 | 11.0 | 51 | 2,533 | 16.4 | 118 | 0 | 16.4 | 98 |
| 9 | 3,067 | 5.7 | 60 | 0 | 5.7 | 56 | 2,829 | 22.3 | 155 | 0 | 22.3 | 108 |
| 10 | 3,385 | 8.1 | 86 | 0 | 8.1 | 63 | 3,123 | 22.8 | 162 | 0 | 22.8 | 121 |
| 11 | 3,696 | 7.1 | 79 | 1 | 10.2 | 68 | 3,411 | 27.3 | 192 | 0 | 27.3 | 131 |
| 12 | 4,007 | 9.7 | 105 | 0 | 9.7 | 73 | 3,695 | 32.7 | 229 | 1 | 50.5 | 143 |
| 13 | 4,308 | 10.7 | 119 | 0 | 10.7 | 79 | 3,973 | 39.1 | 271 | 0 | 39.1 | 154 |
| 14 | 4,609 | 14.0 | 136 | 0 | 14.0 | 83 | 4,246 | 44.1 | 306 | 3 | 84.6 | 166 |
| 15 | 4,900 | 14.1 | 152 | 0 | 14.1 | 89 | 4,519 | 49.9 | 346 | 0 | 49.9 | 176 |
| 16 | 5,189 | 16.6 | 176 | 0 | 16.6 | 96 | 4,789 | 67.0 | 447 | 0 | 67.0 | 187 |
| 17 | 5,477 | 18.1 | 191 | 0 | 18.1 | 103 | 5,053 | 63.6 | 436 | 0 | 63.6 | 198 |
| 18 | 5,764 | 16.7 | 180 | 0 | 16.7 | 108 | 5,313 | 73.8 | 507 | 0 | 73.8 | 207 |
| 19 | 6,048 | 20.5 | 213 | 0 | 20.5 | 113 | 5,572 | 94.4 | 633 | 0 | 94.4 | 216 |
| 20 | 6,325 | 23.9 | 247 | 0 | 23.9 | 117 | 5,829 | 93.2 | 636 | 0 | 93.2 | 229 |
| 21 | 6,602 | 27.2 | 279 | 0 | 27.2 | 121 | 6,083 | 117.0 | 764 | 1 | 140.5 | 238 |
| 22 | 6,872 | 27.4 | 280 | 0 | 27.4 | 126 | 6,332 | 105.8 | 727 | 0 | 105.8 | 247 |
| 23 | 7,140 | 27.6 | 290 | 0 | 27.6 | 130 | 6,579 | 130.7 | 888 | 0 | 130.7 | 258 |
| 24 | 7,408 | 30.6 | 316 | 1 | 37.2 | 136 | 6,823 | 160.3 | 1,019 | 1 | 201.5 | 266 |
| 25 | 7,673 | 27.4 | 288 | 0 | 27.4 | 140 | 7,065 | 224.4 | 1,374 | 2 | 301.4 | 274 |

easily handled with the branch-and-price framework. These large-scale problems are the largest instances of the MCPRP solved to-date.

One avenue for further research of solution approaches to the MCPRP is to investigate the applicability of the branch-and-price framework to the network model developed by Dewil et al. [7]. Utilizing the model of [7] within a branch-and-price framework would not significantly change the structure of the master problem introduced in this paper, except that the packing constraints would correspond to time sections of the hotspots. The master problem's objective function would also need to be modified to incorporate any weights applied to the time sections. The branch-and-price approach could also be applied to variants of the MCPRP which account for overlapping shifts and/or different start/end locations for the patrol cars. The consideration of a heterogeneous fleet for the MCPRP is another possible growth path for the framework introduced in this paper. In this case, we hypothesize that separate subproblems on distinct and specially tailored time-space networks would need to

be considered for each vehicle type. We note that if the patrol vehicles do not share a single transit speed, then Theorem 5.1 (see [10]) is no longer valid, and therefore, the patrol arc construct for the hotspots introduced in this paper would need to be revised.

**Acknowledgements** The authors would like to sincerely thank Dr Reginald Dewil (KU Leuven) for supplying the source code of the Minimum cost network flow problem (MCNFP) model at short notice.

## References

1. Ahuja R, Magnanti T, Orlin J (1993) Network flows: theory, algorithms, and applications. Prentice Hall
2. Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: column generation for solving huge integer programs. Oper Res 46:316–329
3. Çapar İ, Keskin BB, Rubin PA (2015) An improved formulation for the maximum coverage patrol routing problem. Comput Oper Res 59:1–10
4. Chircop PA (2017) Column generation approaches to patrol asset scheduling with complete and maximum coverage requirements. PhD thesis, University of New South Wales, Sydney, Australia
5. Chircop PA, Surendonk TJ, van den Briel MHL, Walsh T (2013) A column generation approach for the scheduling of patrol boats to provide complete patrol coverage. In: Piantadosi J, Anderssen RS, Boland J (eds) Proceedings of the 20th international congress on modelling and simulation. Modelling and Simulation Society of Australia and New Zealand, pp 1110–1116
6. Desrosiers J, Lübbecke ME (2005) A primer in column generation. In: Desaulniers G, Desrosiers J, Solomon MM (eds) Column generation. Springer, US, pp 1–32
7. Dewil R, Vansteenwegen P, Cattrysse D, Oudheusden DV (2015) A minimum cost network flow model for the maximum covering and patrol routing problem. Eur J Oper Res 247:27–36
8. Ford LR, Fulkerson DR (1958) A suggested computation for maximal multi-commodity network flows. Manag Sci 5:97–101
9. Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting-stock problem. Oper Res 9:849–859
10. Keskin BB, Li SR, Steil D, Spiller S (2012) Analysis of an integrated maximum covering and patrol routing problem. Transp Res Part E(mohana) Logist Transpo Rev 48:215–232
11. Király Z, Kovács P (2012) Efficient implementations of minimum-cost flow algorithms. Acta Univ Sapientiae, Inform 4:67–118
12. Li SR (2012) Vehicle routing models in public safety and health care. PhD thesis, The University of Alabama TUSCALOOSA
13. Vanderbeck F (2005) Implementing mixed integer column generation. In: Desaulniers G, Desrosiers J, Solomon MM (eds) Column generation. Springer, US, pp 331–358
14. Vansteenwegen P, Souffriau W, Van Oudheusden D (2011) The orienteering problem: a survey. Eur J Oper Res 209:1–10