Annals of Operations Research
https://doi.org/10.1007/s10479-021-04474-6

ORIGINAL RESEARCH

On routing and scheduling a fleet of resource-constrained vessels to provide ongoing continuous patrol coverage

Paul A. Chircop[1] · **Timothy J. Surendonk**[1] · **Menkes H. L. van den Briel**[2] · **Toby Walsh**[3,4,5]

Accepted: 29 November 2021

Abstract
The objective of the Patrol Boat Scheduling Problem with Complete Coverage (PBSPCC) is to find a minimum size patrol boat fleet to provide continuous coverage at a set of maritime patrol regions, ensuring that there is at least one vessel on station in each patrol region at any given time. The requirement for continuous patrol coverage is complicated by the need for vessels to be replenished on a regular basis. This combinatorial optimization problem contains both routing and scheduling components and is known to be NP-hard. In this paper, we show how recent theoretical insights can be used in conjunction with specially tailored heuristics to accelerate a column generation solution approach over a resource-space-time network construct. We show how the column generation approach can be used within a branch-and-price framework and combined with various reduction techniques to find cyclic and long-term scheduling solutions on a range of test problems.

Keywords Branch-and-price · Continuous coverage problem · Route planning · Surveillance scheduling

✉ Paul A. Chircop
paul.chircop@dst.defence.gov.au

Timothy J. Surendonk
timothy.surendonk@dst.defence.gov.au

Menkes H. L. van den Briel
menkes@hivery.com

Toby Walsh
tw@cse.unsw.edu.au

1 Defence Science and Technology Group, Sydney, NSW, Australia

2 HIVERY, Sydney, NSW, Australia

3 University of New South Wales, Sydney, NSW, Australia

4 Data61 CSIRO, Canberra, Australia

5 Technical University of Berlin, Berlin, Germany

Published online: 20 January 2022

1 Introduction and related literature

The Patrol Boat Scheduling Problem with Complete Coverage (PBSPCC) is a combinatorial optimization problem of determining the minimum number of vessels to provide continuous presence in a set of maritime patrol regions. The continuous presence requirement is complicated by the resource limitations of the vessels, which consume resources at a constant rate while on patrol or in transit. Hence, before a maximum endurance time has elapsed, each vessel must return to a port (replenishment station) to refuel, re-stock and/or re-crew. Once a vessel has been replenished, it is then available to resume its patrolling duties. The PBSPCC was introduced by Chircop et al. (2013), who outlined a simple column generation solution approach and performed a sensitivity analysis of fleet size to vessel endurance on a specific problem instance. A pictorial illustration of the problem is provided in Fig. 1. While the patrol regions are specified by spatial nodes, they may in fact represent more expansive geographical areas. Note that Fig. 1 indicates that a patrol vessel may be in any one of four mutually exclusive states: on patrol, in transit, undergoing replenishment, or idle in port (no resource consumption in this state).

Routing and scheduling naval patrol vessels for military operations has received limited coverage in the academic literature when compared to other problems of interest to decision-makers in the commercial shipping industry. Moreover, there are a number of dissimilarities between the types of objectives for which naval assets and merchant vessels are routinely deployed. The main difference is that a certain level of effectiveness is usually required in military operations planning, with the minimization of costs or the maximization of profit relegated to lower levels of importance. In general, patrol planning problems in the military and public sector domains place a greater emphasis on achieving certain operational outcomes. Even though industrial and commercial applications of routing and scheduling have a certain preeminence in the literature, there are a number of publications that have focused on tasking requirements for naval patrol vessels and their crews. We begin with a brief overview of some of these studies before narrowing our attention to the problem of routing and scheduling a fleet of resource-constrained vessels to provide ongoing continuous patrol coverage.

The literature survey conducted by Christiansen et al. (2004) on the status of ship routing and scheduling included a brief section on naval operations. Among the papers reviewed

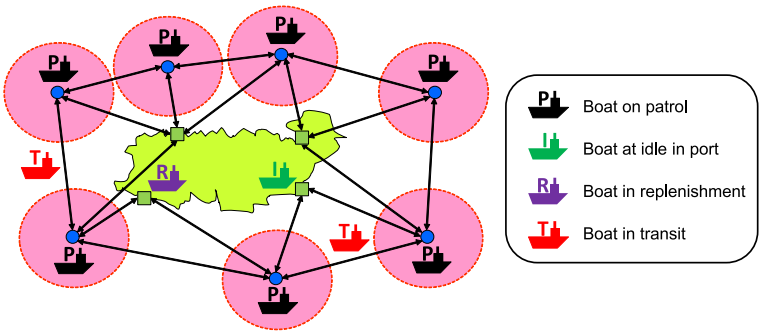


Fig. 1 Illustration of the PBSPCC. Pink circles are indicative of maritime patrol regions, but a region itself is represented by a blue circular node. Ports are displayed as green square nodes. A purple icon (R) indicates a vessel is replenishing at port, a green icon (I) indicates that a vessel is idle in port, and a black icon (P) represents a vessel on patrol. Directed edges are feasible transit lanes in the network, with a red icon (T) used to designate a vessel in transit

by Christiansen et al. (2004) is the article by Nulty and Ratliff (1991), which considered the problem of scheduling the U.S. Navy's Atlantic Fleet in order to satisfy a number of overseas deployments throughout a planning horizon. A study by Brown et al. (1996) was also cited in the Christiansen et al. (2004) survey, which examined the operational scheduling requirements for U.S. Coast Guard cutters in the Boston District. The problem considered by Brown et al. (1996) was to schedule a fleet of sixteen Coast Guard vessels on a weekly basis to ensure that patrol and search and rescue requirements were being satisfied. Darby-Dowman et al. (1995) studied the same problem as Brown et al. (1996) and developed a software system for the U.S. Coast Guard with the aid of a discrete optimization solver. A later study of patrol boat resource allocation within the U.S. Coast Guard was undertaken by Wagner and Radovilsky (2012). This study addressed the allocation of patrol boats across a large variety of districts using an extensive integer programming model. In the Royal Australian Navy context, Horn et al. (2006) considered the problem of simultaneously scheduling patrol boats and crews for regular operations using a model called Crews Boats Missions (CBM). The model used metaheuristic techniques to create schedules for a fleet of Armidale Class Patrol Boats and their crews over a yearly planning horizon. This work was subsequently complemented by Zadeh et al. (2009), who updated CBM into a more efficient software package for military planners.

Over the last decade, a new research trend has emerged to address questions related to routing and scheduling patrol assets to provide an on-station presence in a set of geographical locations. The variety of applications has included studies of maritime and land surveillance (Millar and Russell 2012; Kim et al. 2013; Fang et al. 2015), motor vehicle accident prevention (Keskin et al. 2012; Çapar et al. 2015; Dewil et al. 2015; Chircop et al. 2021), security beat planning (Hsieh et al. 2015) and critical infrastructure protection (Shieh et al. 2013). Surendonk and Chircop (2020b) conducted a comprehensive literature survey of these and other studies in order to properly situate the PBSPCC within the broader corpus of scholarship. With applications to maritime border protection, and having its origins in fleet sizing questions raised by the Royal Australian Navy, the PBSPCC was found to possess a unique set of characteristics that had not been considered by related studies. Subsequent work (Surendonk and Chircop 2020a, b) provided detailed proofs that the PBSPCC belongs firmly within the NP-hard computational complexity class via a reduction to the Hamiltonian Graph Decision Problem.

With a pure integer programming approach to the PBSPCC, the optimality gaps are simply too large to solve even moderately sized instances in reasonable time frames (Chircop 2017). Indeed, similar experiences were reported by Keskin et al. (2012) with their integer programming model of the Maximum Covering and Patrol Routing Problem (MCPRP). Implementation of the simple column generation solution approach proffered by Chircop et al. (2013) has also highlighted the difficulty of obtaining optimal, or even good-quality, solutions on large spatial networks. Moreover, this approach has been known to exhibit significant runtime degradation with longer planning horizons. However, recent work by Chircop and Surendonk (2021) has suggested a column generation enhancement by establishing a set of conditions under which an alternative objective function (*minimize the total time not spent on patrol*) may be used to arrive at an optimal solution to the original problem (*minimize the fleet size*). In the present paper, we adopt the alternative objective function approach in concert with other column generation acceleration strategies to demonstrate computational runtime improvements on a range of test problems. We then show how this enhanced column generation approach can be used within a specially tailored branch-and-price framework to find cyclic and long-term scheduling solutions.

The remainder of the paper is organized as follows. Section 2 covers some mathematical preliminaries and outlines the construction of a resource-space-time (RST) network. In Sect. 3, an integer linear programming (ILP) model is formulated for the PBSPCC. A column generation solution approach to the linear relaxation of the ILP model is then presented. This includes deriving the reduced costs, defining the subproblem, generating seed columns, specifying the pricing strategy and outlining a branch-and-price approach for solving the ILP model. Reduction techniques to find cyclic and long-term scheduling solutions are discussed in Sect. 4 and the computational results are presented and analyzed in Sect. 5. The paper concludes with Sect. 6 in which a few suggestions are proffered for further research.

2 Network construction

We begin with some notation. We use $\mathbb{N}$ to denote the natural numbers (starting at 1), $\mathbb{Z}^*$ to denote the set of non-negative integers, $\mathbb{Z}_n$ for the integers modulo n: $\mathbb{Z}_n := \{0, 1, ..., n-1\}$, and $\mathbb{R}$ for the continuum of real numbers. If X is a set, then we use $|X|$ to denote its cardinality.

2.1 Patrol network setting

A *patrol network* is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where the set of vertices $\mathcal{V} = \{1, \ldots, n\}$ represents the number of distinct spatial regions and $\mathcal{A}$ is the set of directed arcs, that is, the set of feasible transitions (in space) between any two regions. The set of vertices is the union of two mutually exclusive sets, the set of ports $\mathcal{V}_{\text{port}} = \{1, \ldots, m\}$ and the set of patrol regions $\mathcal{V}_{\text{pat}} = \{m+1, \ldots, n\}$. Hence, the number of ports is $m \geq 1$, where $m < n$, and the number of patrol regions is $n - m$. Associated with each arc $(i, j) \in \mathcal{A}$ is a non-negative and integer-valued transit time, which we denote $\mathcal{T}_{ij} \in \mathbb{N}$. For each patrol boat, the transit times on the network $\mathcal{G}$ must satisfy the triangle inequality. That is, for distinct $i, j, k \in \mathcal{V}$, we have $\mathcal{T}_{ik} \leq \mathcal{T}_{ij} + \mathcal{T}_{jk}$.

The *endurance* of a patrol boat is $T_E \in \mathbb{N}$. This is the maximum time that can be spent on patrol and in transit before being replenished at a port. On returning to a port, a patrol boat undergoes a mandatory *replenishment break* of duration $T_R \in \mathbb{Z}^*$. The *planning horizon* is $T \in \mathbb{N}$, where $T \geq T_E + T_R$. The planning horizon is divided into discrete intervals of time, indexed by $t \in \mathbb{Z}_T$. For a given patrol region $i \in \mathcal{V}_{\text{pat}}$, there must be a patrol boat on station at all time instances $t \in \mathbb{Z}_T$. We define a *patrol period* to be a pairing of a discrete time interval and the index denoting a patrol region. The set of all patrol periods is denoted by $\mathcal{L}$ and is given by $\mathcal{L} := \{(i, t) \mid i \in \mathcal{V}_{\text{pat}},\ t \in \mathbb{Z}_T\}$. Finally, we introduce a set of activities consisting of all possible patrol boat states. As seen in Fig. 1, there exist four mutually exclusive states for the patrol boats, namely, on *patrol* (P), in *transit* (T), *replenishing* at port (R), at *idle* in port (I). The set of all activities is denoted by $\Phi := \{\text{I}, \text{P}, \text{R}, \text{T}\}$.

Table 1 summarizes the preceding notation and definitions of the key sets and parameters for the patrol network context of the PBSPCC.

2.2 Resource-space-time network construction

The PBSPCC is a resource-constrained optimization problem, and therefore, we require an appropriate network structure that facilitates an efficient solution algorithm. The approach taken in this paper is to transform the original patrol network $\mathcal{G}$ into an expanded directed

Table 1 Notation and definitions for the patrol network setting

Symbol	Definition		Units
n	Number of spatial locations	$n \in \mathbb{N}, n \geq 2$	–
m	Number of ports	$m \in \mathbb{N}, 1 \leq m < n$	–
$\mathcal{V}_{\text{port}}$	Set of ports in a patrol network	$\mathcal{V}_{\text{port}} = \{1, \ldots, m\}$	–
$\mathcal{V}_{\text{pat}}$	Set of patrol regions in a patrol network	$\mathcal{V}_{\text{pat}} = \{m+1, \ldots, n\}$	–
$\mathcal{V}$	Set of spatial locations in a patrol network	$\mathcal{V} = \mathcal{V}_{\text{port}} \cup \mathcal{V}_{\text{pat}}$	–
i, j	Indices for spatial locations	$i, j \in \mathcal{V}$	–
$\mathcal{A}$	Set of directed arcs in a patrol network	$(i, j) \in \mathcal{A}$	–
$\mathcal{G}$	A patrol network	$\mathcal{G} = (\mathcal{V}, \mathcal{A})$	–
$\mathcal{T}_{ij}$	Transit time from location i to j	$\mathcal{T}_{ij} \in \mathbb{N}$	[time]
T_E	Patrol boat endurance	$T_E \in \mathbb{N}$	[time]
T_R	Replenishment break duration	$T_R \in \mathbb{Z}^*$	[time]
T	Planning horizon	$T \in \mathbb{N}, T \geq T_E + T_R$	[time]
t	Time index	$t \in \mathbb{Z}^*$	[time]
$\mathcal{L}$	Set of patrol periods	$\mathcal{L} = \{(i,t) \mid i \in \mathcal{V}_{\text{pat}},\ t \in \mathbb{Z}_T\}$	–
Φ	Set of patrol boat activities	Φ = {I(dle), P(atrol), R(eplenish), T(ransit)}	–

acyclic graph G that implicitly accounts for the resource consumption and replenishment of a patrol boat through space and time. We call such a directed acyclic graph a *resource-space-time (RST) network*. In this section, we outline the construction and characteristics of an RST network.

At any time within the planning horizon T, a patrol boat will be at some location with a certain resource level and performing a specific activity. For example, a patrol boat may be at a port region in a state of replenishment, or on patrol at a particular region with only a few units of resource remaining. To account for all relevant cases, an RST network contains a set of vertices V and a set of directed arcs A. The vertex set is used to label a patrol boat's location in space, the elapsed time, and the number of resource units consumed. The directed arcs represent the set of patrol boat activities, that is, idle, on patrol, replenishing or in transit. A patrol region is represented as a group of layered vertices. The layers are used to track a boat's resource consumption, while the horizontal position of a vertex is used to mark the elapsed time. Ports are composed predominately of two vertex layers, one for the un-replenished state and another for the replenished state. An illustrative example of an RST network can be found in Fig. 2.

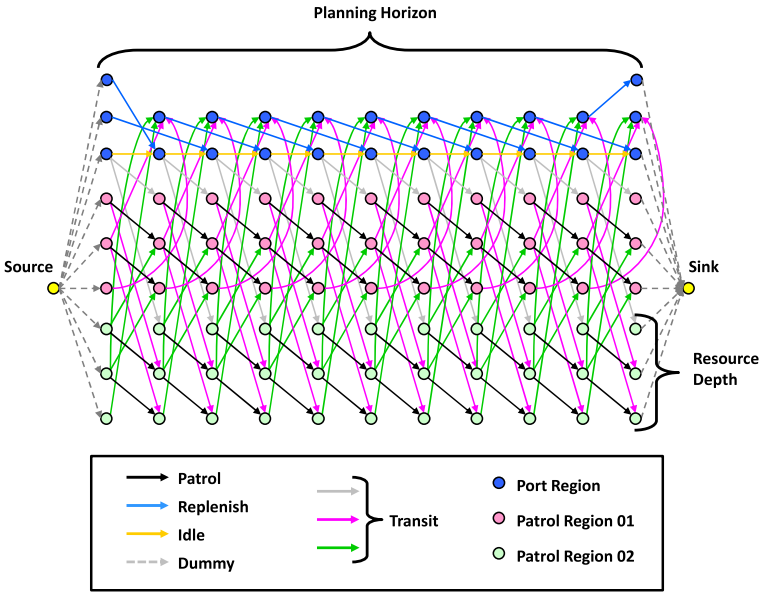


Fig. 2 An illustrative example of an RST network. (Color figure online)

As seen in Fig. 2, idle transitions (orange arcs) are permitted within a port region after resource replenishment (blue arcs) has occurred. Transition from a port to a patrol region (gray arcs) is only allowed after a patrol boat has been replenished. All transition arcs from a patrol region to a port (green and pink arcs in Fig. 2) terminate at a vertex with an emanating replenishment arc. If the resource break duration is greater than the time discretization, additional vertices can be added at the beginning and end of the planning horizon to include fractional replenishment periods (for example, see the top vertex layer of Fig. 2). The RST network can account for multiple port regions, but a patrol boat is prohibited from making a transition between any two distinct ports. The number of vertex layers representing a patrol region is contingent on the maximum amount of time a boat may spend at that region before undergoing replenishment. This is dependent on the patrol boat endurance and the distance to the closest port. Transitions within a patrol region are represented by patrol arcs (black arcs in Fig. 2) which denote one unit of resource consumed per unit time. A patrol period can therefore be expressed as the set of patrol arcs that begin at the same time in a given patrol region. To achieve complete patrol coverage over the entire planning horizon, each patrol period must be covered by at least one patrol boat.

The mathematical notation used to describe the construction of an RST network is as follows. Given a patrol network $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, endurance T_E, replenishment time T_R, and planning horizon T, we construct an RST network $G = (V, A)$ according to a transformation $(\mathcal{G}, T_E, T_R, T) \mapsto G$. The vertex set V includes a source s and a sink τ. All other vertices are divided between a patrol vertex set $V_{\text{pat}} \subset V$ and a port vertex set $V_{\text{port}} \subset V$. Except for dummy arcs, the directed arcs A in an RST network are grouped according to the set of patrol boat activities. Thus, we have a patrol arc set $A_{\text{P}} \subset A$, a transit arc set $A_{\text{T}} \subset A$, a replenishment arc set $A_{\text{R}} \subset A$, and an idle arc set $A_{\text{I}} \subset A$. The set of patrol arcs in patrol region $i \in \mathcal{V}_{\text{pat}}$ ($\mathcal{V}_{\text{pat}} \subset \mathcal{V}$) is $A^i_{\text{P}} \subseteq A_{\text{P}}$. The set of patrol arcs in patrol period $\ell \in \mathcal{L}$ is $A_{\text{P}}(\ell) \subset A^i_{\text{P}}$, where $\ell = (i, t)$ for some $t \in \mathbb{Z}_T$. For each $v \in V$, let $A_+(v)$ be the set of all arcs emanating from v, that is, the set of all $(v, w) \in A$ for fixed $v \in V$. Similarly, let $A_-(v)$

be the set of all arcs terminating at v, that is, the set of all $(u, v) \in A$ for fixed $v \in V$. The set $A_+(s)$ contains the dummy arcs emanating from the source vertex while the set $A_-(\tau)$ corresponds to the dummy arcs terminating at the sink vertex. The entire set of dummy arcs is $A_{\mathrm{D}} = A_+(s) \cup A_-(\tau)$, where $A_{\mathrm{D}} \subset A$. Let $t_{uv} \in \mathbb{Z}^*$ be the transit time of traversing arc $(u, v) \in A$. We have $t_{uv} = 0$ for all $(u, v) \in A_{\mathrm{D}}$, $t_{uv} = 1$ for all $(u, v) \in A_{\mathrm{P}} \cup A_{\mathrm{I}}$, and $t_{uv} \in \{0, 1, \ldots, T_R\}$ for all $(u, v) \in A_{\mathrm{R}}$. The procedure for constructing the set of transit arcs A_{T} is a major component of the transformation from a patrol network to an RST network. Hence, we describe this procedure in detail and provide an illustrative example before proceeding to the next section.

For a patrol region $i \in \mathcal{V}_{\mathrm{pat}}$, the proximity to the nearest port is used to calculate the number of vertex layers, which we call the *resource depth* N_i. Let $R^i_{\min} \in \mathbb{N}$ be the transit time from patrol region i to the nearest port. The resource depth for patrol region i is given by $N_i := T_E - 2R^i_{\min}$. The vertex layers for patrol region i are defined by the set $D_i := \{0, 1, \ldots, N_i\}$. Consider a boat patrolling in region i at some $\hat{a} \in D_i$ and at time $t \in \mathbb{Z}_T$. Now consider a transition to patrol region $j \in \mathcal{V}_{\mathrm{pat}}$, with required transit time $\mathscr{T}_{ij} \in \mathbb{N}$. Firstly, if $t + \mathscr{T}_{ij} > T$, then there is not enough time remaining in the planning horizon to undertake this transition, and so it can be ruled out. However, if $t + \mathscr{T}_{ij} \leq T$, then we need to check if the patrol boat is adequately resourced to undertake the transition from i to j. For $\hat{a} \in D_i$, let the resource level (that is, the amount of resource remaining) be given by $R_i(\hat{a})$, where $R_i(\hat{a}) \geq R^i_{\min}$ and $R_i(\hat{a}) := T_E - R^i_{\min} - \hat{a}$. Upon making the transition to region j, the boat will arrive at some $\hat{b} \in D_j$, where the resource level must be given by $R_j(\hat{b}) = R_i(\hat{a}) - \mathscr{T}_{ij}$. Using the fact that $R_j(\hat{b}) = T_E - R^j_{\min} - \hat{b}$, we have $T_E - R^j_{\min} - \hat{b} = T_E - R^i_{\min} - \hat{a} - \mathscr{T}_{ij}$. Solving for $\hat{b}$ yields $\hat{b} = \hat{a} + R^i_{\min} + \mathscr{T}_{ij} - R^j_{\min}$. If $\hat{b} \leq N_j$, where $N_j = T_E - 2R^j_{\min}$, then the transition is feasible. Otherwise, the patrol boat is not adequately resourced to make the transition from region i to j.

Table 2 summarizes the notation introduced for the construction of an RST network.

The example in Fig. 3 is provided to illustrate the construction of feasible transit arcs between two patrol regions (labeled 01 and 02) in an RST network. The example assumes that the patrol boat endurance is $T_E = 7$ and the transit time between the two patrol regions is $\mathscr{T}_{[01][02]} = 2$. The transit time to the nearest port is $R^{01}_{\min} = 2$ for Patrol Region 01 and $R^{02}_{\min} = 1$ for Patrol Region 02. Hence, the resource depth for Region 01 is $N_{01} = T_E - 2R^{01}_{\min} = 7 - 2 \times 2 = 3$. Similarly, $N_{02} = 5$ for Region 02. This means that the vertex layers for Region 01 are given by $D_{01} = \{0, 1, 2, 3\}$, and $D_{02} = \{0, 1, 2, 3, 4, 5\}$ for Region 02. Thus, the resource level corresponding to the bottom vertex layer of Region 01 is $R_{01}(N_{01}) = T_E - R^{01}_{\min} - N_{01} = 7 - 2 - 3 = 2$, while the resource level of the top layer is $R_{01}(0) = 5$. Similarly, for Region 02 we have $R_{02}(N_{02}) = 1$ and $R_{02}(0) = 6$. Figure 3 shows the feasible transit arcs in the RST network from Region 02 to Region 01 at time $t = 1$ (green arcs) and from Region 01 to Region 02 at time $t = 4$ (pink arcs). Clearly, a transition from Region 02 to Region 01 cannot occur when the resource level is less than 4, nor can a transition take place from Region 01 to Region 02 when the resource level is less than 3.

3 Problem formulation and solution approach

An RST network $G = (V, A)$ can be used to formulate the PBSPCC as an integer linear program (ILP), suitable for the application of a column generation approach. Let P be the set of all feasible paths through an RST network G from the source s to the sink τ, where a feasible path $p \in P$ represents a patrol vessel's trajectory through space and time. The

Table 2 Notation and definitions for the RST network setting

Symbol	Definition		Units
s	Source vertex in an RST network		–
τ	Sink vertex in an RST network		–
V_{port}	Set of port vertices in an RST network		–
V_{pat}	Set of patrol vertices in an RST network		–
V	Set of vertices in an RST network	$V = \{s, \tau\} \cup V_{\text{port}} \cup V_{\text{pat}}$	–
$A_+(v)$	Set of arcs emanating from $v \in V$		–
$A_-(v)$	Set of arcs terminating at $v \in V$		–
A_{I}	Set of idle arcs in an RST network		–
A_{P}	Set of patrol arcs in an RST network		–
A_{P}^i	Set of patrol arcs in region $i \in \mathcal{V}_{\text{pat}}$	$A_{\text{P}}^i \subseteq A_{\text{P}}$	–
$A_{\text{P}}(\ell)$	Set of patrol arcs in patrol period $\ell \in \mathcal{L}$	$A_{\text{P}}(\ell) \subset A_{\text{P}}^i, \ell = (i, t), t \in \mathbb{Z}_T$	–
A_{R}	Set of replenishment arcs in an RST network		–
A_{T}	Set of transit arcs in an RST network		–
A_{D}	Set of dummy arcs in an RST network	$A_{\text{D}} = A_+(s) \cup A_-(\tau)$	–
A	Set of directed arcs in an RST network	$A = A_{\text{I}} \cup A_{\text{P}} \cup A_{\text{R}} \cup A_{\text{T}} \cup A_{\text{D}}$	–
G	An RST network	$G = (V, A)$	–
R_{min}^i	Transit time from $i \in \mathcal{V}_{\text{pat}}$ to the nearest port	$R_{\text{min}}^i \in \mathbb{N}$	[time]
N_i	Resource depth for patrol region $i \in \mathcal{V}_{\text{pat}}$	$N_i = T_E - 2R_{\text{min}}^i$	–
D_i	Set of vertex layers for patrol region $i \in \mathcal{V}_{\text{pat}}$	$D_i = \{0, 1, \ldots, N_i\}$	–
$\hat{a}$	Index for a vertex layer set	$\hat{a} \in D_i$	–
$R_i(\hat{a})$	Resource level corresponding to vertex layer $\hat{a} \in D_i$	$R_i(\hat{a}) = T_E - R_{\text{min}}^i - \hat{a}$	[time]
t_{uv}	Transit time for traversing arc $(u, v) \in A$	$t_{uv} \in \mathbb{Z}^*$	[time]

objective is to find a smallest subset of P such that each patrol period $\ell \in \mathcal{L}$ is covered by at least one path. For each $p \in P$, let $x_{uvp} = 1$ if path $p \in P$ uses arc $(u, v) \in A$ and $x_{uvp} = 0$ otherwise. As an RST network G contains no cycles, we can invoke the Flow

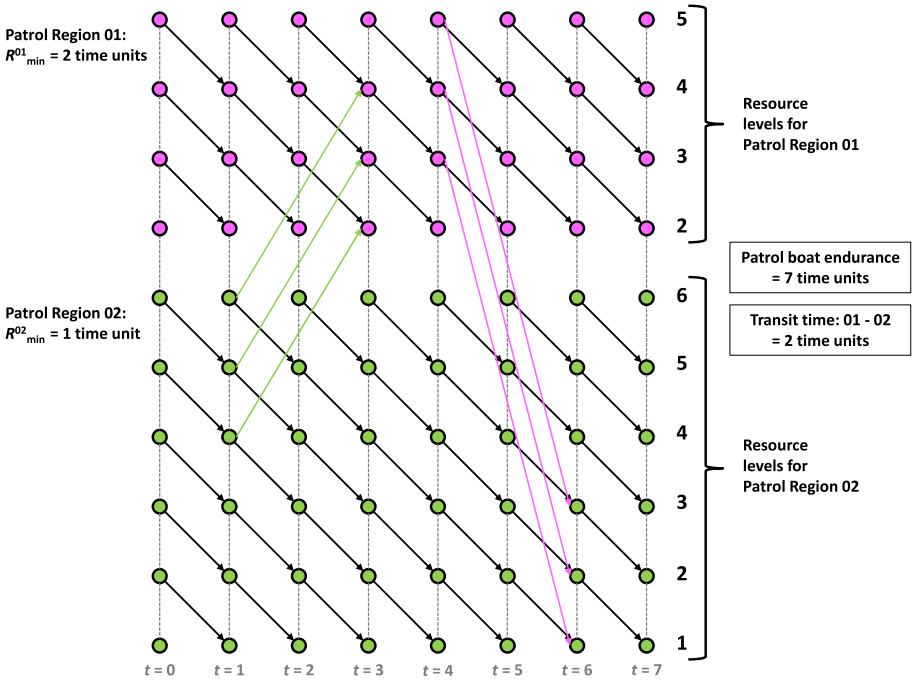


Fig. 3 Example of feasible transitions between two patrol regions in an RST network. (Color figure online)

Decomposition Theorem (Ahuja et al. 1993) to express the flow over an arc $(u, v) \in A$ 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. Let the aggregate path flow over an arc $(u, v) \in A$ be given by $x_{uv} \in \mathbb{Z}^*$, where $x_{uv} = \sum_{p \in P} x_{uvp} \lambda_p$. To track the patrol periods covered by an individual path, we introduce a binary parameter $a_{\ell p}$, where $a_{\ell p} = 1$ if $p \in P$ patrols $\ell \in \mathcal{L}$ and $a_{\ell p} = 0$ otherwise. Alternatively, this parameter can be expressed as $a_{\ell p} = \sum_{(u,v) \in A_{\mathrm{P}}(\ell)} x_{uvp}$.

We denote $\bar{c}_p = \sum_{(u,v) \in A_{\mathrm{P}}} t_{uv} x_{uvp}$ to be the total time spent on patrol and $c_p = T - \bar{c}_p$ to be the aggregate non-patrol time for any path $p \in P$. The maximum patrol time over all paths is denoted by $\bar{c}_{\max}$, where $\bar{c}_{\max} := \max \left\{ \bar{c}_p \mid p \in P \right\}$. We can determine a value for $\bar{c}_{\max}$ by applying costs $\mu_{uv} = -t_{uv}$ to arcs $(u, v) \in A_{\mathrm{P}}$ and costs $\mu_{uv} = 0$ to all other arcs in the network and then invoking a shortest path algorithm from s to τ. Given that G is a directed acyclic graph, the shortest path can be extracted by performing a series of edge relaxations over a topologically sorted list of the vertices V (Cormen et al. 2009). We denote this procedure as DAG-SP(G, μ, s, τ), which outputs a shortest path p from s to τ (assuming the cost structure μ) and the associated path cost $\delta_\mu(s, \tau)$. Thus, $\bar{c}_{\max}$ can be obtained by negation of the shortest path cost from s to τ, that is, $\bar{c}_{\max} = -\delta_\mu(s, \tau)$. We denote $\bar{c}_{ip}$ to be the total time spent on patrol in region $i \in \mathcal{V}_{\mathrm{pat}}$ for path $p \in P$, and thus we can write $\bar{c}_{ip} = \sum_{(u,v) \in A^i_{\mathrm{P}}} t_{uv} x_{uvp}$. A simple lower bound $\kappa_{\min}$ on the number of paths in an optimal solution is defined by $\kappa_{\min} := \lceil |\mathcal{L}| \,/\, \bar{c}_{\max} \rceil$, where $\lceil \circ \rceil$ is the ceiling function, which returns the smallest integer greater than or equal to the argument. Finally, a further lower bound on the aggregate non-patrol time is defined by $\varepsilon_{\min} := \kappa_{\min} (T - \bar{c}_{\max})$.

Table 3 contains a summary of the notation used in the ILP model of the PBSPCC.

Table 3 Notation and definitions for the integer linear programming model

Sets	
P	Set of all feasible paths from s to τ in an RST network
Ordered sets	
p	A feasible path $p \in P$
	An ordered set of arcs connecting s and τ
Parameters	
$\bar{c}_p$	The patrol time for path $p \in P$
	$\bar{c}_p = \sum_{(u,v)\in A_\mathrm{P}} t_{uv} x_{uvp}$
$\bar{c}_{\max}$	The maximum patrol time over all paths in P
	$\bar{c}_{\max} = \max\left\{\bar{c}_p \mid p \in P\right\}$
c_p	The non-patrol time for path $p \in P$
	$c_p = T - \bar{c}_p$
$\kappa_{\min}$	Lower bound on the number of paths
	$\kappa_{\min} = \lceil\, \lvert\mathcal{L}\rvert \,/\, \bar{c}_{\max} \,\rceil$
$\varepsilon_{\min}$	Lower bound on the aggregate non-patrol time
	$\varepsilon_{\min} = \kappa_{\min}\,(T - \bar{c}_{\max})$
x_{uvp}	Binary parameter for a path $p \in P$ over an arc $(u, v) \in A$
	$x_{uvp} = 1$ if path $p \in P$ uses arc $(u, v) \in A$ and $x_{uvp} = 0$ otherwise
$a_{\ell p}$	Binary parameter for a path $p \in P$ over a patrol period $\ell \in \mathcal{L}$
	$a_{\ell p} = 1$ if path $p \in P$ covers $\ell \in \mathcal{L}$ and $a_{\ell p} = 0$ otherwise
$\bar{c}_{ip}$	The patrol time of path $p \in P$ in region $i \in \mathcal{V}_{\mathrm{pat}}$
	$\bar{c}_{ip} = \sum_{(u,v)\in A^i_\mathrm{P}} t_{uv} x_{uvp}$
μ_{uv}	Cost applied to arc $(u, v) \in A$
	$\mu_{uv} \in \mathbb{R}$
$\delta_\mu(s, \tau)$	Cost of shortest path from s to τ assuming cost structure μ
	$\delta_\mu(s, \tau) \in \mathbb{R}$
Decision variables	
λ_p	Binary integer variable for a path $p \in P$
	$\lambda_p = 1$ if path $p \in P$ is used and $\lambda_p = 0$ otherwise
x_{uv}	Integer variable for the path flow over an arc $(u, v) \in A$
	$x_{uv} = \sum_{p\in P} x_{uvp}\lambda_p$

3.1 Master problem

With the preceding definitions and notation, we can formulate an integer linear programming master problem for the PBSPCC as follows (the dual variables of the linear programming relaxation are listed down the right-hand side):

$$\text{minimize} \quad \sum_{p \in P} \lambda_p, \tag{1}$$

$$\text{subject to} \quad \sum_{p \in P} a_{\ell p} \lambda_p \geq 1, \quad \forall \ell \in \mathcal{L}, \qquad [\pi_\ell] \tag{2}$$

$$\sum_{p \in P} \bar{c}_p \lambda_p \geq |\mathcal{L}|, \qquad [\alpha] \tag{3}$$

$$\sum_{p \in P} \bar{c}_{ip} \lambda_p \geq T, \quad \forall i \in \mathcal{V}_{\text{pat}}, \quad [\beta_i] \tag{4}$$

$$\sum_{p \in P} c_p \lambda_p \geq \varepsilon_{\min}, \qquad [\gamma] \tag{5}$$

$$\sum_{p \in P} \lambda_p \geq \kappa_{\min}, \qquad [\zeta] \tag{6}$$

$$\lambda_p \in \{0, 1\}, \quad \forall p \in P. \tag{7}$$

The objective function (1) seeks to minimize the number of patrol vessels. The objective is followed by a series of covering constraints given through (2). There is a single covering constraint for each patrol period $\ell \in \mathcal{L}$, ensuring that each patrol period is patrolled by at least one vessel. In addition to (7), which enforces a binary integrality condition on the decision variables, we have a set of constraints (3)–(6) pertaining to various bounds which can be inferred from the optimization problem. Constraint (3) ensures that the aggregate patrol time is at least the number of patrol periods. The constraints given through (4) guarantee that the aggregate patrol time allotted to each region $i \in \mathcal{V}_{\text{pat}}$ is bounded below by the planning horizon T. A lower bound on the aggregate non-patrol time ($\varepsilon_{\min}$) can be found in (5), while (6) gives a lower bound on the number of patrol boats ($\kappa_{\min}$) required to satisfy the complete coverage requirements.

As noted in the introductory section, recent work by Chircop and Surendonk (2021) has shown that under certain conditions, the objective (1) may be swapped out for one which seeks to minimize the non-patrol time in order to accelerate a conventional column generation approach. By adopting this alternative objective function, the goal is not necessarily to find a set of paths through the RST network that minimize the non-patrol time, but rather, to obtain an optimal solution to the original problem (1)–(7). The conditions under which such a swap can be carried out are described below.

Consider the following reformulated integer linear programming master problem:

$$\text{minimize} \quad \sum_{p \in P} c_p \lambda_p, \tag{8}$$

$$\text{subject to} \quad (2) - (7). \tag{9}$$

Let z^*_{LP} be the optimal objective value of the linear programming relaxation of (8)–(9), that is, the objective obtained by relaxing the integrality requirement on the decision variables and setting $\lambda_p \geq 0$ for all $p \in P$. Suppose we have a set of paths $Q \subset P$ which collectively satisfy the constraints (2)–(7), with $\lambda^Q_p = 1$ if $p \in Q$ and $\lambda^Q_p = 0$ otherwise, and let $z' := \sum_{p \in P} \lambda^Q_p$. Let z^*_{pat} be the optimal objective value of (8)–(9) and suppose that $\sum_{p \in P} c_p \lambda^Q_p \geq z^*_{\text{pat}}$. Finally, assume that the set Q satisfies the following condition:

$$z' < \frac{|\mathcal{L}| + z^*_{\text{LP}}}{T} + 1. \tag{10}$$

Then $z' = z^*_{\text{size}}$, where z^*_{size} is the optimal objective value of (1)–(7). The condition specified by (10) allows us to check if any *feasible* solution to (8)–(9) is an *optimal* solution to (1)–(7). Due to the suggested column generation runtime enhancements with the alternative objective function (Chircop and Surendonk 2021), we adopt it in the next section to derive the reduced cost of a path through an RST network.

3.2 Column generation subproblem

As a preliminary step to outlining the definition of the reduced costs and the column generation subproblem, we introduce two useful surjective mappings that associate patrol arcs to their respective patrol periods and patrol regions. These mappings are $\phi : A_{\text{P}} \to \mathcal{L}'$ and $\psi : A_{\text{P}} \to \mathcal{V}_{\text{pat}}$, where $\mathcal{L}' := \{1, \ldots, |\mathcal{L}|\}$ is an index set of the patrol periods. As each $v \in V_{\text{pat}}$ can be represented by the triple $(i, \hat{a}, t)$, where $i \in \mathcal{V}_{\text{pat}}$, $\hat{a} \in D_i$ and $t \in \mathbb{Z}_T$, we can define ψ as follows:

$$\psi(u, v) := i, \quad \forall (u, v) \in A_{\text{P}}, \text{ where } v = (i, \hat{a}, t),\ \hat{a} \in D_i,\ t \in \mathbb{Z}_T.$$

If $(u, v) \in A_{\text{P}}$, with $v = (i, \hat{a}, t)$, then we define ϕ as:

$$\phi(u, v) := \begin{cases} t & \text{if } i = m + 1, \\ (i - m - 1)T + t & \text{if } i \in \{m + 2, \ldots, n\}. \end{cases}$$

To deduce the nature of the column generation subproblem over an RST network, the mathematical structure of the reduced cost of a path needs to be established. The reduced cost $\bar{r}_p$ of a path $p \in P$ through an RST network G is defined by $\bar{r}_p := c_p - \boldsymbol{\pi}^{\text{t}}\mathbf{A}_p$, where $\boldsymbol{\pi}^{\text{t}}$ is a row vector of the dual variables of the master problem, $\mathbf{A}_p$ is the column corresponding to variable λ_p, and c_p is the cost coefficient of λ_p in the objective function (8). In terms of the underlying arc variables of the RST network, the reduced cost of path $p \in P$ can be written as follows:

$$\begin{aligned}
\bar{r}_p &= c_p - \boldsymbol{\pi}^{\text{t}}\mathbf{A}_p, \\
&= c_p - \sum_{\ell \in \mathcal{L}} \pi_\ell a_{\ell p} - \alpha \bar{c}_p - \sum_{i \in \mathcal{V}_{\text{pat}}} \beta_i \bar{c}_{ip} - \gamma c_p - \zeta, \\
&= \Bigg(\sum_{(u,v) \in A \setminus A_{\text{P}}} t_{uv} x_{uvp} \Bigg) - \Bigg(\sum_{(u,v) \in A_{\text{P}}} \pi_{\phi(u,v)} x_{uvp} \Bigg) - \alpha \Bigg(\sum_{(u,v) \in A_{\text{P}}} t_{uv} x_{uvp} \Bigg) \\
&\quad - \Bigg(\sum_{(u,v) \in A_{\text{P}}} \beta_{\psi(u,v)} t_{uv} x_{uvp} \Bigg) - \gamma \Bigg(\sum_{(u,v) \in A \setminus A_{\text{P}}} t_{uv} x_{uvp} \Bigg) - \zeta \Bigg(\sum_{(s,v) \in A_+(s)} x_{svp} \Bigg), \\
&= \sum_{(u,v) \in A \setminus A_{\text{P}}} \big[(1 - \gamma) t_{uv} \big] x_{uvp} \\
&\quad - \sum_{(u,v) \in A_{\text{P}}} \big[\pi_{\phi(u,v)} + \big(\alpha + \beta_{\psi(u,v)} \big) t_{uv} \big] x_{uvp} - \sum_{(s,v) \in A_+(s)} \zeta x_{svp}.
\end{aligned}$$

If we define cost coefficients μ_{uv} as follows:

$$\mu_{uv} := \begin{cases} (1 - \gamma) t_{uv} & \text{if } (u, v) \in A_{\text{I}} \cup A_{\text{R}} \cup A_{\text{T}}, \\ -\pi_{\phi(u,v)} - \big(\alpha + \beta_{\psi(u,v)} \big) t_{uv} & \text{if } (u, v) \in A_{\text{P}}, \\ -\zeta & \text{if } (u, v) \in A_+(s), \\ 0 & \text{otherwise}, \end{cases}$$

then the reduced cost of path $p \in P$ can be expressed as $\bar{r}_p = \sum_{(u,v)\in A} \mu_{uv} x_{uvp}$.

Having uncovered the mathematical form of the reduced cost of a path through an RST network, we can determine the nature of the pricing subproblem. Assume that the set of paths generated at a given iteration of the column generation routine is given by $P' \subset P$. Then the pricing subproblem is to find a path $p \in P \backslash P'$ of minimum negative reduced cost. This can be written as a combinatorial optimization problem as follows:

$$\text{minimize} \quad \sum_{(u,v)\in A} \mu_{uv} x_{uv}, \tag{11}$$

$$\text{subject to} \quad \sum_{(s,v)\in A_+(s)} x_{sv} = 1, \tag{12}$$

$$\sum_{(u,v)\in A_-(v)} x_{uv} = \sum_{(v,w)\in A_+(v)} x_{vw}, \qquad \forall v \in V \backslash \{s, \tau\}, \tag{13}$$

$$\sum_{(u,\tau)\in A_-(\tau)} x_{u\tau} = 1, \tag{14}$$

$$x_{uv} \in \{0, 1\}, \qquad \forall (u, v) \in A. \tag{15}$$

The structure of (11)–(15) reveals that finding a minimum reduced cost path is a pure shortest path problem over an RST network.

3.3 Restricted master problem

The column generation process is initialized by relaxing the integrality requirement on the decision variables (setting $\lambda_p \geq 0$) and selecting a set of paths $P' \subset P$ satisfying the constraints. This yields an initial *restricted master problem* (RMP) with an associated *initial primal basis* (seed columns). The seed columns may be selected by any means, so long as the set covering constraints corresponding to the coverage requirements of the patrol periods are satisfied. (Seed column construction strategies are covered in the next section.) Therefore, using primed notation to indicate a restriction to paths $P' \subset P$, we can express the RMP for the minimum non-patrol time formulation of the PBSPCC as follows:

$$\text{minimize} \quad \sum_{p\in P'} c_p \lambda_p, \tag{16}$$

$$\text{subject to} \quad (2)' - (6)', \tag{17}$$

$$\lambda_p \geq 0, \quad \forall p \in P'. \tag{18}$$

The column generation procedure takes a set of dual variables $\boldsymbol{\pi}$ from the RMP (16)–(18) to construct a subproblem over an RST network G, which is a shortest path problem over a directed acyclic graph (11)–(15). If the path returned from the subproblem has negative reduced cost, a new variable and the associated column defining the path are added to the RMP. The process continues in a cyclical manner until no negative reduced cost path can be returned by the subproblem. This framework corresponds to column generation at the *root node* (Ford and Fulkerson 1958; Dantzig and Wolfe 1960; Gilmore and Gomory 1961). Recall that the RMP does not impose an integrality requirement on the decision variables, only non-negativity. However, the goal is to obtain an optimal integer solution

to the master problem (1)–(7). If the variables λ_p are all integral once the column generation process has terminated, the problem is said to have been *solved at the root node*. If not, the column generation procedure can be embedded in a *branch-and-price* algorithm to arrive at an integer solution. The full branch-and-price procedure is described in a later section.

3.4 Seed columns and pricing strategy

In order to start the column generation procedure, an initial feasible basis and associated seed columns must be selected for the RMP. Although Chvatal (1983) originally outlined a general first phase procedure for constructing an initial primal basis, Lübbecke (2001) has noted that specially tailored construction techniques should be preferred, given that the initialization of the RMP can influence the runtime efficiency of a column generation approach. Desaulniers et al. (2002) have observed that a good initial primal basis can reduce the number of iterations between the RMP and the subproblem, thus improving column generation efficiency. This is consistent with the following insight of Vanderbeck (1994): "*With an appropriate initial set of columns, one can get a good start in the column generation procedure.*" Lübbecke (2001) has stated that the negation of this assertion also holds, that is, a poorly selected initial basis can result in degraded column generation performance.

Given the set covering constraints (2), an initial basis must cover each patrol period in the underlying RST network. In order to obtain a good start to the column generation procedure, an initial set of high-quality covering paths should be obtained. In order to obtain an initial basis of quality, we proffer a *greedy shortest path heuristic* (GSPH). The GSPH first applies a cost $\mu_{uv} = -T$ to each patrol arc $(u, v) \in A_{\mathrm{P}}$ and a cost $\mu_{uv} = 0$ to each arc $(u, v) \in A \backslash A_{\mathrm{P}}$. A shortest path p is then found over the RST network with the applied cost structure. By examining the arcs $(w, x) \in p$ such that $(w, x) \in A_{\mathrm{P}}$, the arc costs of the RST network are updated by setting $\mu_{uv} = 0$ for all $(u, v) \in A_{\mathrm{P}}$ such that $\phi(u, v) = \phi(w, x)$, and $\mu_{wx} = t_{wx}$ for all $(w, x) \in p$ such that $(w, x) \in A \backslash A_{\mathrm{P}}$. A new shortest path is subsequently returned from the network with the updated cost structure and the procedure cycles, terminating when a non-negative shortest path cost is found. This condition is usually satisfied when all patrol periods have been covered by the set of generated paths. For cases in which the GSPH fails to cover every patrol period, a *Big-M method* with a single artificial variable may be added to the initial restricted master problem. The GSPH is summarized in Algorithm 1.

Algorithm 1 Greedy Shortest Path Heuristic (GSPH)

Input: An RST network $G = (V, A)$ with source $s \in V$ and sink $\tau \in V$
procedure GSPH(G, s, τ)
  for $(u, v) \in A \backslash A_{\mathrm{P}}$ **do**
    $\mu_{uv} \leftarrow 0$
  end for
  for $(u, v) \in A_{\mathrm{P}}$ **do**
    $\mu_{uv} \leftarrow -T$
  end for
  $P' \leftarrow \varnothing$
  $(p, \delta_\mu(s, \tau)) \leftarrow$ DAG-SP(G, μ, s, τ)
  while $\delta_\mu(s, \tau) < 0$ **do**
    $P' \leftarrow P' \cup \{p\}$
    for $(w, x) \in p$ **do**
      if $(w, x) \in A_{\mathrm{P}}$ **then**
        for $(u, v) \in A_{\mathrm{P}}$ such that $\phi(u, v) = \phi(w, x)$ **do**
          $\mu_{uv} \leftarrow 0$
        end for
      else
        $\mu_{wx} \leftarrow t_{wx}$
      end if
    end for
    $(p, \delta_\mu(s, \tau)) \leftarrow$ DAG-SP(G, μ, s, τ)
  end while
  return P'
end procedure

A common strategy to accelerate column generation algorithms is to return multiple negative reduced cost columns at each call to the subproblem(s). Lübbecke and Desrosiers (2005) have noted that returning multiple negative reduced cost columns is easily handled by dynamic programming algorithms and that implementing such a strategy can decrease the number of calls to the subproblem(s). Furthermore, Desaulniers et al. (2002) have stated that such a strategy can be made even more efficient if the pool of columns returned from a single call to the subproblem(s) closely resembles the structure of an optimal integer solution. The same authors suggest that greedy heuristics may be employed profitably to price out a collection of columns for problems in which an optimal integer solution is likely to be task-disjoint. By considering the patrol periods as tasks, such a strategy can be carefully exploited to achieve runtime efficiency gains for our column generation approach.

If the minimum non-patrol time formulation of the PBSPCC is adopted, a planning horizon T can be selected to arrive at solutions that are strongly task-disjoint. Since it is possible for more than one vessel to be on station in a patrol region at any given time, a poorly chosen planning horizon may result in a solution that minimizes the non-patrol time without minimizing the number of patrol boats used. We refer to this phenomenon as *over-searching* the patrol regions. (An example of over-searching is provided by Chircop and Surendonk (2021).) In order to avoid over-searching and increase the likelihood that solutions consist of columns that are strongly task-disjoint, the planning horizon should be set to an integer multiple of the aggregate endurance and replenishment break, that is, $T = J(T_E + T_R)$,

where $J \in \mathbb{N}$. While we do not have a theoretical proof of this assertion, the computational experiments conducted as part of this research show strong evidence for it. (Future work could include a theoretical investigation as to whether this assertion is true or false.)

In line with the insights of Lübbecke and Desrosiers (2005) and Desaulniers et al. (2002), we propose a greedy heuristic for the selection of multiple reduced cost paths with a planning horizon of the form $T = J(T_E + T_R)$. The procedure bears a close resemblance to the greedy shortest path heuristic for the seed columns in Algorithm 1. The heuristic is executed when a call to the subproblem is made in an iteration of the column generation procedure. After the dual costs from the RMP have been applied to the RST network, the least negative reduced cost path is returned via the application of a shortest path algorithm. Following a similar pattern to Algorithm 1, the arc costs in the RST network are subsequently updated by their respective transit times. The procedure cycles until the reduced cost of a returned path is non-negative. The heuristic is outlined as a subroutine of the column generation algorithm for the PBSPCC which can be found in Algorithm 2.

Algorithm 2 Column generation procedure

1: **Input**: An RST network $G = (V, A)$ with source $s \in V$ and sink $\tau \in V$
2: $P' \leftarrow \text{GSPH}(G, s, \tau)$
3: **procedure** GENERATECOLUMNS(G, s, τ, P')
4: Construct RMP from P' with associated variables $\{\lambda_p\}_{p \in P'}$
5: Solve the RMP to get dual variables $\boldsymbol{\pi}$ and apply these as arc costs (μ) to G
6: $(p, \delta_\mu(s, \tau)) \leftarrow \text{DAG-SP}(G, \mu, s, \tau)$
7: $\bar{r}_p^* \leftarrow \delta_\mu(s, \tau)$
8: **if** $\bar{r}_p^* \geq 0$ **then**
9: **END**
10: **else**
11: **while** $\bar{r}_p^* < 0$ and finite **do**
12: Add new variable λ_p and associated column to RMP
13: $P' \leftarrow P' \cup \{p\}$
14: **for** $(w, x) \in p$ **do**
15: **if** $(w, x) \in A_{\text{P}}$ **then**
16: **for** $(u, v) \in A_{\text{P}}$ such that $\phi(u, v) = \phi(w, x)$ **do**
17: $\mu_{uv} \leftarrow t_{uv}$
18: **end for**
19: **else**
20: $\mu_{wx} \leftarrow t_{wx}$
21: **end if**
22: **end for**
23: $(p, \delta_\mu(s, \tau)) \leftarrow \text{DAG-SP}(G, \mu, s, \tau)$
24: $\bar{r}_p^* \leftarrow \delta_\mu(s, \tau)$
25: **end while**
26: **goto** 4
27: **end if**
28: **end procedure**

3.5 Branch-and-price

In this section, we outline an approach to obtain solutions to the integer linear program of the PBSPCC (1)–(7). Columns generated at the root node are usually insufficient to obtain quality integer solutions. This is because the pool of columns generated at the root node is usually not

large enough to close the integrality gap below 1% (Desaulniers et al. 2002). Indeed, even if the root node column pool contains all the necessary information for an optimal solution, this can still present difficulties for state-of-the-art integer programming solvers to obtain quality solutions in a reasonable time frame. However, if the column generation process is embedded in a branch-and-bound tree, integral solutions may be obtained. This paradigm is known by the term *branch-and-price* (Barnhart et al. 1998). There are many ways to perform branch-and-price when attempting to solve integer linear programs, spanning a range of exact and heuristic strategies (Vanderbeck 2000). The branching rules and the manner in which they are imposed are dependent on the structure and properties of the underlying subproblem(s). In general, branching decisions are either (i) directly enforced on the subproblem (so long as the structure is maintained) or they are (ii) incorporated as cuts in the master problem (Conforti et al. 2014). In our branch-and-price approach to the PBSPCC, we adopt strategy (ii). (For example, this branching strategy is used by Ben Amor and Valério de Carvalho (2005) for a branch-and-price approach to the cutting stock problem with an underlying multi-commodity flow model.)

At a given node in the branch-and-bound tree, suppose we have a solution to the relaxed problem (16)–(18) given by $\{\tilde{\lambda}_p \in \mathbb{R}_{\geq 0} \mid p \in P'\}$, where the *tilde* indicates the presence of fractional values. The corresponding flow over an arc $(u, v) \in A$ in the underlying RST network is given by $\tilde{x}_{uv} = \sum_{p \in P'} x_{uvp} \tilde{\lambda}_p$. If all of the arc flows are integral, that is $\tilde{x}_{uv} \in \mathbb{Z}^*$ for all $(u, v) \in A$, then the node can be pruned. (Note that we require integrality of the path flow variables for a feasible integer solution, not just integrality of the underlying RST network arc variables (Vanderbeck 2005).) Otherwise, we can select an arc $(u, v) \in A$ with fractional flow to branch on by creating two new nodes in the branch-and-bound tree as follows:

$$\sum_{p \in P'} x_{uvp} \lambda_p \leq \lfloor \tilde{x}_{uv} \rfloor \quad \text{or} \quad \sum_{p \in P'} x_{uvp} \lambda_p \geq \lceil \tilde{x}_{uv} \rceil.$$

A new node is instantiated by incorporating one of the above inequalities into the relaxed problem (16)–(18) and executing the column generation procedure. In addition, the dual variable associated with the new constraint is imposed as a cost on the relevant arc in the subproblem (11)–(15).

Our branch-and-price approach essentially follows the paradigm outlined above but is augmented with heuristic rules for the selection of arcs (to branch on) and the determination of candidate nodes to fathom. The arc selection heuristic is able to choose a combination of fractional arcs for branching. Recall that the set of arcs A in an RST network can be disaggregated according to the various patrol vessel activities $\Phi = \{\text{I(dle)}, \text{P(atrol)}, \text{R(eplenish)}, \text{T(ransit)}\}$ and the D(ummy) arcs A_{D}. Once a subset $\Phi' \subseteq \Phi \cup \{D\}$ of arc groups is selected, the most fractional arc in each group can be determined. To determine the most fractional arc in the set $A_\varphi \subset A$, where $\varphi \in \Phi'$, we first introduce the mapping:

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

The most fractional arc $(u, v)^*_\varphi \in A_\varphi$ is defined by:

$$(u, v)^*_\varphi := \underset{(u,v) \in A_\varphi}{\arg\min}\, F(u, v).$$

Once the most fractional arc has been found for each $\varphi \in \Phi'$, two disjunctive branches from the current node can be created. On the left branch, we impose:

$$\sum_{p \in P'} x_{[(u,v)^*_\varphi]p} \lambda_p \leq \lfloor \tilde{x}_{(u,v)^*_\varphi} \rfloor, \quad \forall \varphi \in \Phi'.$$

Similarly, on the right branch we enforce:

$$\sum_{p \in P'} x_{[(u,v)^*_\varphi]p} \lambda_p \geq \lceil \tilde{x}_{(u,v)^*_\varphi} \rceil, \quad \forall \varphi \in \Phi'.$$

The motivation for enforcing multiple inequalities within a single branching decision is to arrive quickly at feasible integer solutions to the minimum non-patrol time formulation of the PBSPCC. Once a feasible solution to the minimum non-patrol time formulation (8)–(9) has been obtained, we can check if it is optimal with respect to the minimum fleet size formulation (1)–(7) via condition (10).

Selection of a node for branching follows a type of depth-first search (Christiansen and Nygreen 1998; Desaulniers et al. 2002) which is also designed to arrive at a feasible integer solution quickly. For each unexplored candidate node, we form the sets:

$$X_0 := \{(u, v) \in A \mid \mathbf{1}_{\mathbb{R}^+}(\tilde{x}_{uv}) = 1\},$$
$$X_1 := X_0 \cap \{(u, v) \in A \mid \tilde{x}_{uv} - \lfloor \tilde{x}_{uv} \rfloor = 0\},$$

where $\mathbf{1}_{\mathbb{R}^+} : \mathbb{R} \to \{0, 1\}$ is the indicator function with respect to the positive real numbers $\mathbb{R}^+$. The node with the greatest ratio $|X_1|/|X_0|$ is selected for branching, with ties being broken by the node with the least objective function value. If $|X_1|/|X_0| = 1$ at a given node, all arc flows in the RST network are integral. However, integrality of the path flow variables $\tilde{\lambda}_p$ must also be verified. Nodes which are arc integral but not path integral can be pruned from the tree. Once a candidate integer solution has been found, it is checked against condition (10) to verify that the solution is optimal for the minimum fleet size problem (1)–(7). If the solution is not optimal, we have an upper bound which can be used for fathoming and pruning other unexplored nodes. Alternatively, the integer solution may be checked against the constraint (6) if a lower bound has been set.

4 Problem reduction techniques

The branch-and-price framework outlined in the previous section can be augmented with a number of specially tailored problem reduction techniques for large-scale patrol networks and/or instances with long planning horizons. The problem reduction techniques can be applied at the geographical (spatial) and/or planning (temporal) levels. At the geographical level, a patrol network may be clustered or partitioned into mutually exclusive subsets, thus permitting each sub-network to be solved individually and then recombined to form a single solution. In the temporal domain, a large planning horizon may be constructed from smaller sub-blocks, with the terminal configuration of one planning block matching the initial conditions of the subsequent one.

4.1 Cyclic schedules

The first problem reduction technique is designed to find a set of patrol vessel schedules that permute with each other over a given planning block. Solutions of this type are cyclic

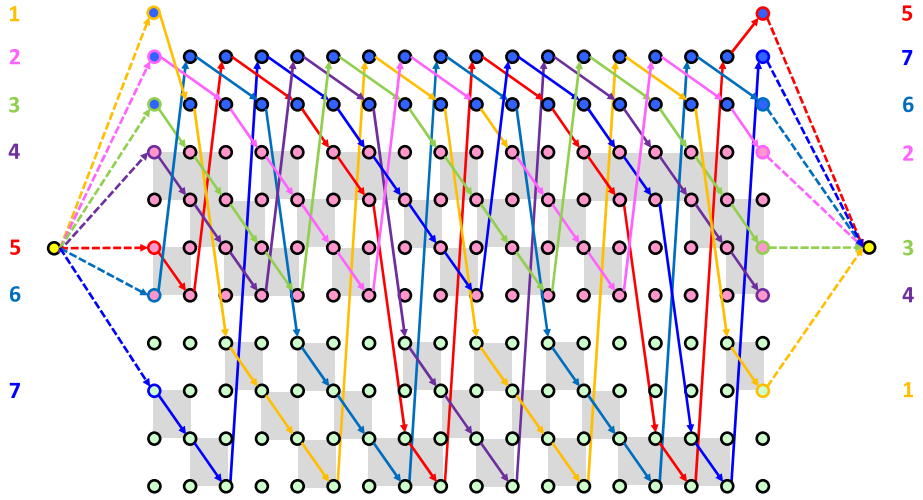


Fig. 4 Example RST network showing a cyclic scheduling solution. The schedule can be written in cyclic permutation notation as (1, 7, 2, 4, 6, 3, 5). Qualitatively, this means that path 1 is continued by path 7, which is continued by path 2,..., which is continued by path 5, which is continued by path 1,... The gray shading denotes patrol arcs which cover a given patrol period. Note that some patrol periods are covered by more than one path

and therefore have the advantage of providing complete patrol coverage indefinitely. In this section, we propose a heuristic regime for finding cyclic scheduling solutions over an RST network. An important design feature of the heuristic is that the structure of the column generation subproblem is preserved.

We begin by defining a one-to-one correspondence between the sets $A_+(s)$ and $A_-(\tau)$. That is, to each element $(s, v) \in A_+(s)$ we associate a unique element $(u, \tau) \in A_-(\tau)$ such that all the elements of $A_+(s)$ map to all the elements of $A_-(\tau)$. We construct a bijection $f : A_+(s) \to A_-(\tau)$ such that $f(s, v) = (u, \tau)$ where $v = (i, \hat{a}, 0)$, $u = (i, \hat{a}, T)$, $i \in \mathcal{V}$, and $\hat{a} \in D_i$. (Note that for $i \in \mathcal{V}_{\text{port}}$, $D_i = \{0, 1, \ldots, T_R\}$, while for $i \in \mathcal{V}_{\text{pat}}$, $D_i = \{0, 1, \ldots, N_i\}$.) Let $Q \subset P$ be a collection of paths through an RST network such that each patrol period $\ell \in \mathcal{L}$ is covered, and let $\lambda_p^Q = 1$ if $p \in Q$ and $\lambda_p^Q = 0$ otherwise. Suppose that:

$$\sum_{p \in P} x_{svp} \lambda_p^Q = \sum_{p \in P} x_{[f(s,v)]p} \lambda_p^Q, \quad \forall (s, v) \in A_+(s).$$

Then the set of paths Q is called a *cyclic scheduling solution*. An illustrative example with $|Q| = 7$ is provided in Fig. 4 for an RST network with one port and two patrol regions. The transit time between any two distinct locations is one unit, with $T_E = 5$, $T_R = 2$ and $T = 17$.

The first stage of the heuristic is to construct a new RST network $\bar{G} = (\bar{V}, \bar{A})$ using the set of paths Q over the original RST network $G = (V, A)$. The new RST network has the same properties as the original, although it may be constructed with a different planning horizon, $\bar{T}$ say. Denote the source and sink vertices in the new RST network as $\bar{s}$ and $\bar{\tau}$, respectively, and let a feasible path be denoted by $\bar{p} \in \bar{P}$, where $\bar{P}$ is the set of all feasible paths from $\bar{s}$ to $\bar{\tau}$ in $\bar{G}$. The set of source vertex arcs in the new RST network $\bar{A}_+(\bar{s})$ is symmetrical to the arcs used by Q in the original sink vertex set $A_-(\tau)$. When forming the new RST network, the sets $\bar{A}_+(\bar{s})$ and $\bar{A}_-(\bar{\tau})$ are initialized by the empty set $\varnothing$. Then, for each $(u, \tau) \in A_-(\tau)$

such that $\sum_{p \in Q} x_{u\tau p} \lambda_p^Q = 1$, we add an arc $(\bar{s}, \bar{v})$ to $\bar{A}_+(\bar{s})$, where $u = (i, \hat{a}, T) \in V$ and $\bar{v} = (i, \hat{a}, 0) \in \bar{V}$. Then, for each $(\bar{s}, \bar{v}) \in \bar{A}_+(\bar{s})$, we add $(\bar{u}, \bar{\tau})$ to $\bar{A}_-(\bar{\tau})$, where $\bar{v} = (i, \hat{a}, 0) \in \bar{V}$ and $\bar{u} = (i, \hat{a}, \bar{T}) \in \bar{V}$. Moreover, the following constraints are added to the initial restricted master problem pertaining to the new underlying RST network:

$$\sum_{\bar{p} \in \bar{P}'} x_{\bar{s}\bar{v}\bar{p}} \lambda_{\bar{p}} \geq 1, \quad \forall (\bar{s}, \bar{v}) \in \bar{A}_+(\bar{s}), \tag{19}$$

$$\sum_{\bar{p} \in \bar{P}'} x_{\bar{u}\bar{\tau}\bar{p}} \lambda_{\bar{p}} \geq 1, \quad \forall (\bar{u}, \bar{\tau}) \in \bar{A}_-(\bar{\tau}). \tag{20}$$

As before, the dual variables associated with these constraints are imposed as costs on the relevant arcs in the column generation subproblem.

When a feasible collection of paths $\bar{Q} \subset \bar{P}$ is obtained with constraints (19)–(20) satisfied at unity, we have obtained a cyclic scheduling solution with $|\bar{Q}| = |Q|$. However, there is no guarantee that such a solution over the new RST network will be immediately found. In such a case, the original RST network is re-visited to find a different solution from the branch-and-price tree, and a new RST network can be constructed to find a cyclic collection of the requisite size. This process can be reiterated until a cyclic scheduling solution is found, or otherwise terminated with the best solution after a predetermined number of unsuccessful attempts. Note that symmetries may exist in which the same unsuccessful configuration of source and sink arcs is replicated by more than one collection of paths. Such a phenomenon can be handled in a heuristic manner by creating a tabu list of collections of sink vertex arcs. The tabu list can be further employed to prune unfeasible nodes in the branch-and-price tree. In addition to the tabu list, a bounds constraint may be inferred. When a set of paths Q fails to produce a cyclic scheduling solution in the new RST network, the following constraint can added to the restricted master problem of the original RST network:

$$\sum_{p \in P'} \sum_{(u,\tau) \in A_-(\tau)} x_{u\tau p} \lambda_p \leq |Q| - 1. \tag{21}$$

The constraint (21) can be added to each unexplored node in the search tree to prevent the unsuccessful path configuration from being regenerated.

It may be possible for two or more paths in the set Q to terminate at the same arc in the sink vertex set $A_-(\tau)$. In this case, the sets $\bar{A}_+(\bar{s})$ and $\bar{A}_-(\bar{\tau})$ in the new RST network are constructed in a slightly different manner to the process outlined above. Figure 5 provides an example that illustrates the alternative arc construction on the source and sink vertices in the new RST network. In the original RST network, one arc (u_1, τ) in the sink vertex set is used by paths p_1, p_2 and p_3, while another (u_2, τ) is shared by paths p_4 and p_5. In the new RST network, we must construct the source and sink vertex sets so that $|\bar{A}_+(\bar{s})| = |\bar{A}_-(\bar{\tau})| = |Q|$, in accordance with the constraints (19)–(20) that are used to derive cyclic scheduling solutions. To construct the set $\bar{A}_+(\bar{s})$, new vertices $\bar{v}_{1,1}$, $\bar{v}_{1,2}$, $\bar{v}_{1,3}$ are required for the arc shared by p_1, p_2 and p_3, and vertices $\bar{v}_{2,1}$, $\bar{v}_{2,2}$ are introduced for the arc used by p_4 and p_5. Similarly, mirror vertices $\bar{u}_{1,1}$, $\bar{u}_{1,2}$, $\bar{u}_{1,3}$, $\bar{u}_{2,1}$, $\bar{u}_{2,2}$ are created to construct the set $\bar{A}_-(\bar{\tau})$. Since p_6 does not share arc (u_3, τ) with another path, the construction of its corresponding source and sink arcs in the new RST network follows the standard process outlined previously.

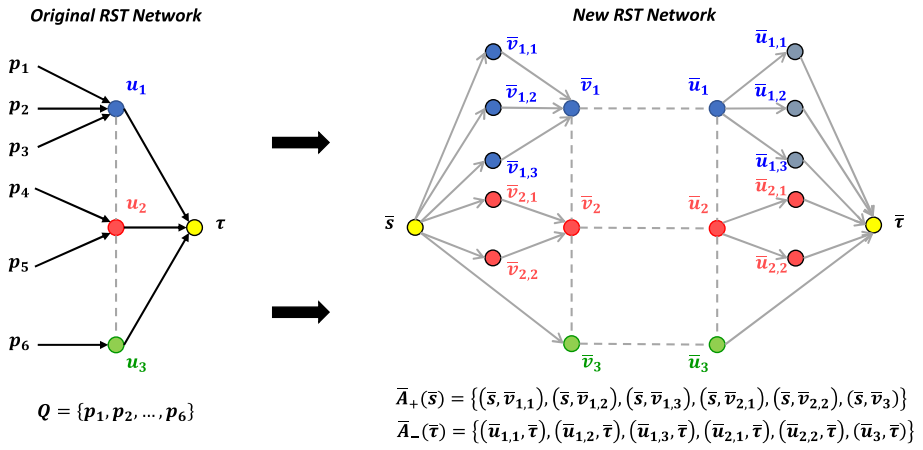


Fig. 5 Construction of the source and sink vertex sets $\bar{A}_+(\bar{s})$ and $\bar{A}_-(\bar{\tau})$, respectively, in the new RST network from a set of paths Q in the original RST network. If any paths in Q terminate at the same sink arc in $A_-(\tau)$, then the introduction of additional vertices is required in the new RST network so that constraints (19)–(20) remain valid for deriving cyclic scheduling solutions

4.2 Rolling horizon

Patrol boat schedules need not permute with each other in order to provide complete coverage in the long term or indefinitely. For cases in which cyclic scheduling solutions are difficult to find over a given planning block, alternative methods of extending the individual patrol vessel schedules can be considered. Furthermore, there may be other factors which render cyclic scheduling solutions undesirable, for example, in adversarial contexts where it is advantageous for individual patrol vessel schedules to exhibit less predictable behavior. Therefore, as an alternative to cyclic schedules, we also propose a *rolling horizon* approach to finding long-term scheduling solutions.

The rolling horizon approach follows the same method which was outlined for cyclic schedules, but with one crucial difference with respect to the construction of the sink vertex set $\bar{A}_-(\bar{\tau})$. Once a solution over the original RST network has been found, the source vertex set $\bar{A}_+(\bar{s})$ for the new RST network is constructed in an identical manner to the process for cyclic scheduling solutions, with flow constraints given by (19). The sink vertex set in the new RST network, however, mirrors the original set, so that $|A_-(\tau)| = |\bar{A}_-(\bar{\tau})|$. Thus, we have:

$$\bar{A}_-(\bar{\tau}) = \{(\bar{u}, \bar{\tau}) \mid \bar{u} = (i, \hat{a}, \bar{T}),\ i \in \mathcal{V},\ \hat{a} \in D_i\}.$$

Therefore, in place of (20), the following constraints are activated in the restricted master problem pertaining to the new RST network:

$$\sum_{\bar{p} \in \bar{P}'} x_{\bar{u}\bar{\tau}\bar{p}} \lambda_{\bar{p}} \geq 0, \quad \forall (\bar{u}, \bar{\tau}) \in \bar{A}_-(\bar{\tau}).$$

Once a solution matching the desired number of paths has been found over the new RST network, the terminal configuration can be used to initialize another planning block, and the process may continue until a desired horizon length has been achieved. Each new planning block may be of the same length, or differently sized blocks may be joined together. In the

event that an initial configuration for a new planning block produces an unsuccessful result, it is added to a branch-and-price tabu list to prevent its regeneration.

In some instances, a terminal configuration may prove impossible to extend into a new planning block using an equivalent number of paths if there exists a patrol boat with a low resource level. In such cases, setting a limit on the amount of resource consumed at the final time interval in each patrol region may render the solution extension process easier. The terminal resource level restriction can be implemented by removing a given number of sink arcs from the bottom layers of each patrol region in the RST network. To do this, we introduce a *terminal resource limit parameter* $\hat{N} \in \mathbb{Z}^*$. For all $i \in \mathcal{V}_{\text{pat}}$ and $l \in \{1, \ldots, \hat{N}+1\}$, the sink arcs $(\bar{u}, \bar{\tau})$ are deleted from $\bar{A}_-(\bar{\tau})$, where $\bar{u} = (i, N_i - l + 1, \bar{T})$. For example, when $\hat{N} = 0$, the arc $(\bar{u}, \bar{\tau})$ with $\bar{u} = (i, N_i, \bar{T})$ is deleted from $\bar{A}_-(\bar{\tau})$ for each patrol region $i \in \mathcal{V}_{\text{pat}}$. When $\hat{N} = 1$, we delete the sink arcs with $\bar{u} = (i, N_i, \bar{T})$ and $\bar{u} = (i, N_i - 1, \bar{T})$ for each patrol region $i \in \mathcal{V}_{\text{pat}}$. The rolling horizon approach can dynamically update the value of $\hat{N}$ as the number of failed attempts to extend a solution increases.

4.3 Clustering

Desaulniers et al. (2002) have suggested that problem partitioning can be used as a preprocessing technique to decompose large-scale problems into smaller ones. Decomposition in the temporal domain was considered in the previous sections, in which the cyclic schedule and rolling horizon approaches were proffered to find solutions over long planning horizons. In the spatial domain, a patrol network may be decomposed into smaller geographical subsets to be solved separately and then recombined to form a solution over the entire patrol network. This procedure is known as *clustering*, and is applicable to large-scale and symmetric patrol networks. Examples of the application of clustering in concert with the aforementioned temporal decomposition techniques are provided in the next section on a range of test problems.

5 Computational results

In this section, the branch-and-price framework described in Sect. 3 is combined with the problem reduction techniques outlined in Sect. 4 and applied to a variety of test problem instances. Problem instances are solved with the cyclic schedule method in most cases. For the remaining problem instances, the rolling horizon approach is adopted. Clustering of large-scale and symmetric patrol networks is also performed on select problems in concert with the aforementioned temporal decomposition techniques. We begin by introducing the patrol networks used for the test problem instances, followed by an analysis of the column generation root node performance and a presentation of benchmark solutions obtained via branch-and-price. (The computational results presented in this section were derived using 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 problems and RST networks, were coded in the Java programming language and the Eclipse integrated development environment.)

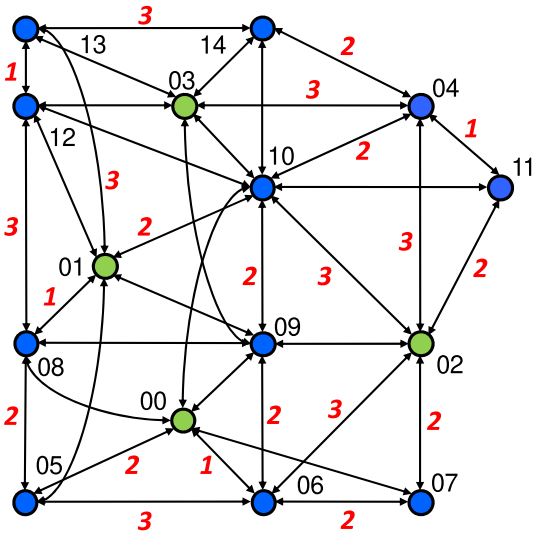


Fig. 6 Patrol Network 6, containing four ports and eleven patrol regions. Patrol regions are represented by blue vertices while the ports are given by green vertices. Feasible bi-directional connections between the spatial locations are shown with indicative transit times in red text

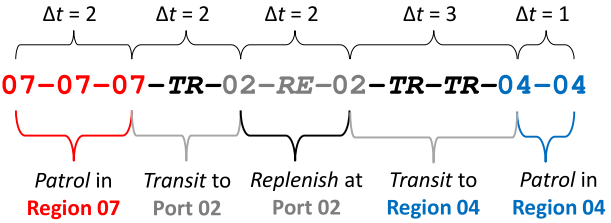


Fig. 7 Schedule notation key for patrol vessels. This example refers to Patrol Network 6 (see Fig. 6) and assumes a replenishment break duration $T_R = 2$

5.1 Patrol networks and schedule notation

A set of 20 patrol networks is used to generate test problem instances to benchmark our solution approaches to the PBSPCC. These networks are intended to reflect a range of graph-theoretic topologies, for example, minimally and maximally connected, multiple and single port, symmetric and asymmetric. Some of the network designs are motivated by real-world examples of patrol boat operations (Chircop 2017). The number of ports $|\mathcal{V}_{\text{port}}|$ and the number of patrol regions $|\mathcal{V}_{\text{pat}}|$ in each patrol network $\mathcal{G}$ are given in Table 4. One of the patrol networks (number 6) is shown in Fig. 6, where patrol regions are represented by blue vertices, ports are given by green vertices, and two-way connecting edges define feasible transitions between the spatial locations. For details on the topological features of all the patrol networks, the reader is referred to Appendix D of the PhD thesis by Chircop (2017).

A notation key for patrol vessel schedules is shown in Fig. 7. This figure can be used as an aid to interpreting the scheduling solutions presented in the following sections. Unit time intervals are represented by dashed line segments (–), and these are punctuated by a reference to a port, patrol region, transition (*TR*) or replenishment break (*RE*). The example provided in Fig. 7 refers to a portion of a patrol vessel schedule operating within Patrol Network 6 (see Fig. 6) where $T_R = 2$. The schedule segment describes a vessel patrolling in Region 07, transiting to Port 02 for a replenishment break, then moving on to patrol Region 04.

Table 4 Summary of patrol networks used to generate test problem instances

$\mathcal{G}$	$\lvert\mathcal{V}_{\text{port}}\rvert$	$\lvert\mathcal{V}_{\text{pat}}\rvert$	$\mathcal{G}$	$\lvert\mathcal{V}_{\text{port}}\rvert$	$\lvert\mathcal{V}_{\text{pat}}\rvert$	$\mathcal{G}$	$\lvert\mathcal{V}_{\text{port}}\rvert$	$\lvert\mathcal{V}_{\text{pat}}\rvert$	$\mathcal{G}$	$\lvert\mathcal{V}_{\text{port}}\rvert$	$\lvert\mathcal{V}_{\text{pat}}\rvert$
1	1	6	6	4	11	11	2	11	16	1	6
2	1	4	7	6	13	12	8	16	17	1	6
3	3	3	8	4	9	13	1	9	18	1	4
4	1	10	9	8	10	14	1	5	19	4	12
5	3	7	10	3	12	15	1	3	20	2	3

5.2 Problem instances, results and solutions

A list of 60 problem instances can be found in Table 5. The leading number of each problem instance refers to the underlying patrol network from Table 4. An asterisk (*) is used to indicate a problem instance to which the rolling horizon solution approach is applied. For each instance, Table 5 summarizes the key input parameters, namely, the patrol vessel endurance T_E, replenishment break duration T_R, the planning horizon of the initial RST network T, and the planning horizon of the new RST network(s) $\bar{T}$. When the rolling horizon approach is applied to a problem instance, a solution is sought over a total horizon length of $T + 10 \times \bar{T}$.

Table 6 summarizes the column generation performance at the root node in terms of 'Col.'—the number of columns generated, and 'Ti.'—the CPU time in seconds. The column generation schema pertains to the linear programming RMP formulation (16)–(18) from Sect. 3.3, and the seed column construction heuristic and pricing strategy outlined in Sect. 3.4. Table 6 contains a further column 'Root?', with a 'Yes' entry indicating instances that are naturally integer at the root node (the so-called integrality property). From Table 6, it can be seen that integral root node solutions were found for Instances 1a and 20c. Both of these were found to possess cyclic block scheduling structures. For example, from Fig. 8 it can be observed that the root node solution for Instance 1a is cyclic over a period of 22 time units.

For the remaining instances without the integrality property, the column generation procedure took less than 30 sec to solve the RMP, with the exception of one (Instance 10c). For most instances, the number of generated columns was of order 10^2, with only a few problems requiring the generation of more than 10^3 columns. The worst performing case was Instance 10c, for which over 5000 columns were generated in approximately 11 minutes. We note that Instance 10c is predicated on a large symmetric patrol network and a long planning horizon, and hence, 10c is a good candidate for the application of temporal and spatial reduction techniques. When compared to previous column generation approaches to the PBSPCC, the results in Table 6 provide strong evidence for the superiority of the procedure adopted in this paper. The improvements are particularly apparent on instances possessing large-scale patrol networks and/or long planning horizons. For a detailed comparison with less sophisticated column generation approaches, the reader is referred to Chapter 5 of the PhD thesis by Chircop (2017).

In concert with the problem reduction techniques (Sect. 4), the branch-and-price approach (Sect. 3.5) was applied to the problem instances that did not solve at the root node. Table 7 contains a summary of the solutions for all 60 problem instances. The relevant column headings are to be interpreted as follows. 'Ti.' is the CPU time in seconds. 'L.B.' is a lower bound on the number of vessels required, and is obtained from solving the linear programming relaxation of (1)–(7). $|Q|$ is the number of vessels used in a complete patrol coverage solution.

Table 5 Test problem instances

Inst.	T_E	T_R	T	$\bar{T}$	Inst.	T_E	T_R	T	$\bar{T}$	Inst.	T_E	T_R	T	$\bar{T}$
1a	18	4	88	88	8a	12	0	12	24	15a	27	2	58	58
1b	16	0	16	48	8b	14	4	18	36	15b	37	0	74	74
1c	24	4	28	56	8c	20	2	22	44	16a	10	5	30	30
1d	28	0	28	56	8d	26	4	30	30	16b	16	0	16	16
2a	12	0	24	48	9a	15	0	15	30	16c	22	4	26	26
2b	30	0	30	30	9b	24	4	28	56	16d	27	0	27	27
3a	10	5	30	36	10a	19	0	19	38	17a	16	2	18	36
3b	18	0	18	36	10b	16	1	34	34	17b	26	4	30	30
4a	23	2	25	50	10c	16	4	60	60	17c	30	4	34	34
4a*	23	2	25	25	10d*	28	4	32	32	17d	42	0	42	42
4b*	40	4	44	44	11a	15	2	17	34	18a	14	2	48	48
5a	10	0	30	30	11b	16	0	16	32	18b	28	4	32	64
5b	28	0	28	56	12a	12	0	12	24	18c	36	0	72	72
6a	13	0	13	26	12a*	12	0	12	18	19a	16	3	19	38
6b	14	0	14	28	12b	14	0	14	28	19b	26	4	30	30
6c	16	1	17	34	12c	18	0	18	36	19c	20	2	44	44
6d	18	0	18	36	13a	21	4	25	25	19d*	18	1	19	19
7a	12	2	14	28	13b	24	4	28	28	20a	32	5	37	37
7b	14	2	16	32	14a	17	2	19	38	20b	40	0	40	40
7c	18	2	20	40	14b	26	4	30	30	20c	13	2	45	45

'Test' is a numerical value derived from the expression on the right-hand side of (10) and is used to verify a solution's optimality. The value of 'Opt?' is Boolean, equal to 'Yes' if $|Q| <$ Test and 'No' otherwise; a 'Yes' entry indicates that a solution is optimal, while a 'No' entry means that optimality has not been established. Finally, T_{sol} is the long-term planning horizon for which a given solution is valid; an entry of ∞ indicates that a solution is cyclic, which can therefore be extended indefinitely.

Of the 60 test problem instances, 42 were solved to optimality. For half of the optimal solutions, it was found that $|Q| =$ L.B., that is, there was no integrality gap between the optimal solution to the linear programming relaxation and that of the integer program. For the other half of optimal solutions except one, it was observed that $|Q| = \lceil \text{L.B.} \rceil$, indicating only a fractional integrality gap. The largest integrality gap for all optimal solutions occurred on Instance 17c, for which $|Q| = 9$ and L.B. $= 8.0$. As the solution to the linear programming relaxation constitutes a lower bound on the optimal integer programming solution, the results suggest that our modeling formulation of PBSPCC is strong. Of the 18 solutions that were not provably optimal, 16 of these can be classified as good quality solutions, that is, $|Q| - \lceil \text{L.B.} \rceil = 1$ for these solutions.

The cyclic scheduling heuristic was used on all but five problem instances to obtain solutions over an infinite planning horizon. For the five instances solved with the rolling

Table 6 Column generation at the root node for test problem instances

Inst.	Col.	Ti.	Root?	Inst.	Col.	Ti.	Root?	Inst.	Col.	Ti.	Root?
1a	452	8.4	Yes	8a	134	0.4	No	15a	227	1.2	No
1b	94	0.2	No	8b	159	0.7	No	15b	229	2.2	No
1c	141	0.6	No	8c	329	2.6	No	16a	585	2.0	No
1d	164	0.8	No	8d	216	2.7	No	16b	139	0.5	No
2a	119	0.2	No	9a	194	0.8	No	16c	430	2.9	No
2b	85	0.4	No	9b	339	4.9	No	16d	131	1.1	No
3a	70	0.1	No	10a	202	1.2	No	17a	1045	6.0	No
3b	56	0.2	No	10b	837	6.2	No	17b	1576	21.5	No
4a	252	1.8	No	10c	5179	696.6	No	17c	555	5.9	No
4a*	–	–	–	10d*	642	8.9	No	17d	344	6.0	No
4b*	460	20.4	No	11a	375	1.5	No	18a	596	2.4	No
5a	558	2.9	No	11b	249	1.3	No	18b	695	7.4	No
5b	137	1.7	No	12a	155	0.7	No	18c	145	2.0	No
6a	180	0.7	No	12a*	–	–	–	19a	379	1.7	No
6b	190	0.7	No	12b	237	1.5	No	19b	739	8.9	No
6c	308	1.6	No	12c	422	3.6	No	19c	1446	23.0	No
6d	247	1.5	No	13a	1060	7.6	No	19d*	241	1.1	No
7a	356	1.4	No	13b	1369	13.3	No	20a	62	0.3	No
7b	390	2.0	No	14a	281	0.7	No	20b	73	0.6	No
7c	473	4.0	No	14d	771	3.8	No	20c	101	0.1	Yes

Fig. 8 Scheduling solution for Instance 1a ($T_E = 18$, $T_R = 4$, $T = 88$), with Problem Network 1 shown on the right-hand side. This solution is cyclic over a period of 22 time units, is naturally integer at the root node and exhibits a block scheduling structure. A total of 12 vessels are required for complete patrol coverage, with replenishment at port synchronized within the two equal groupings of the fleet. Exactly two vessels are needed to cover each region, with one time unit of overlapping patrol incurred per handover

horizon heuristic, two (4b* and 10d*) were found to have optimal cyclic structures. The rolling horizon approach was implemented by using the default lower bound $\kappa_{\min}$ (6) on the first planning block of length T (run 1). The lower bound imposed on the planning blocks of length $\bar{T}$ (runs 2–11) replaced $\kappa_{\min}$ with $|Q|$, that is, the cardinality of the solution over the planning block of length T. The transit arcs of the RST networks were selected

Table 7 Summary of solutions to test problem instances

Inst.	Ti.	L.B.	$\|Q\|$	Test	Opt?	T_{sol}	Inst.	Ti.	L.B.	$\|Q\|$	Test	Opt?	T_{sol}
1a	8.4	12.0	12	12.45	Yes	∞	11a	139.3	15.3	16	16.31	Yes	∞
1b	217.2	10.0	10	10.75	Yes	∞	11b	434.3	13.3	14	14.18	Yes	∞
1c	11.3	10.0	10	10.57	Yes	∞	12a	197.6	20.0	21	20.33	No	∞
1d	46.3	8.0	8	8.71	Yes	∞	12a*	1608.1	20.0	20	20.33	Yes	192
2a	2.2	6.0	6	7.00	Yes	∞	12b	356.3	19.0	20	19.71	No	∞
2b	1.8	5.0	5	5.67	Yes	∞	12c	2941.9	18.0	19	19.00	No	∞
3a	1.7	8.0	8	8.00	Yes	∞	13a	37.3	13.2	15	14.19	No	∞
3b	0.7	4.0	4	4.89	Yes	∞	13b	924.6	13.2	14	14.04	Yes	∞
4a	88.5	17.0	18	17.80	No	∞	14a	17.6	9.3	10	10.07	Yes	∞
4a*	24.2	17.0	17	17.80	Yes	275	14b	4.4	7.2	8	8.15	Yes	∞
4b*	366.1	14.0	14	14.82	Yes	∞	15a	10.7	5.0	5	5.79	Yes	∞
5a	67.2	9.6	10	10.48	Yes	∞	15b	61.4	4.0	4	4.92	Yes	∞
5b	50.1	8.0	8	8.71	Yes	∞	16a	74.0	14.4	15	15.24	Yes	∞
6a	44.8	15.0	16	15.75	No	∞	16b	109.3	7.4	8	8.43	Yes	∞
6b	30.5	14.6	16	15.39	No	∞	16c	207.1	8.2	9	9.20	Yes	∞
6c	54.1	14.9	16	15.70	No	∞	16d	161.6	7.0	7	7.81	Yes	∞
6d	466.2	13.6	15	14.41	No	∞	17a	171.5	10.4	11	11.09	Yes	∞
7a	221.5	20.6	23	21.51	No	∞	17b	39.9	8.3	9	9.25	Yes	∞
7b	270	19.2	21	20.11	No	∞	17c	183.5	8.0	9	9.14	Yes	∞

Table 7 continued

Inst.	Ti.	L.B.	$\|Q\|$	Test	Opt?	T_{sol}	Inst.	Ti.	L.B.	$\|Q\|$	Test	Opt?	T_{sol}
7c	436.8	17.5	19	18.38	No	∞	17d	46.9	7.0	7	7.79	Yes	∞
8a	79.3	11.0	12	11.83	No	∞	18a	13.7	7.6	8	8.44	Yes	∞
8b	48.8	14.0	14	14.67	Yes	∞	18b	13.8	5.6	6	6.55	Yes	∞
8c	308.4	11.0	12	12.00	No	∞	18c	33.5	5.0	5	5.61	Yes	∞
8d	98.5	12.0	12	12.40	Yes	∞	19a	131.3	17.2	18	18.16	Yes	∞
9a	22.9	12.9	15	13.77	No	∞	19b	12.9	15.3	16	16.33	Yes	∞
9b	174.4	13.5	15	14.46	No	∞	19c	26.6	15.3	16	16.19	Yes	∞
10a	458.5	16.0	17	16.88	No	∞	19d*	104.9	15	16	15.93	No	209
10b	10.1	18.2	19	19.11	Yes	∞	20a	53.1	5.0	5	5.76	Yes	∞
10c	20.1	21.3	22	22.20	Yes	∞	20b	3.4	4.0	4	4.80	Yes	∞
10d*	16.5	16.3	17	17.33	Yes	∞	20c	0.1	9.0	9	10.00	Yes	∞

as the default for branching decisions on all problem instances where the branch-and-price procedure was invoked. For select problems, branching on the transit arcs was augmented to include combinations of replenishment and dummy arcs. There was no significant variance observed between the results produced by the default and augmented branching strategies, with respect to the number of nodes explored and the CPU time.

The cyclic scheduling heuristic performed effectively on most problem instances, meaning that the number of failed attempts to find a cyclic solution was bounded above by 20 on all but three instances. The worst case performance with respect to the number of failed attempts was on Instance 20a, where an optimal solution was found on attempt 44. However, given that Instance 20a took under a minute to solve, the performance of the heuristic can be described as effective with respect to the CPU time. On the other hand, the heuristic performed extremely well (no failed attempts) on some problems, for example, Instance 12c. However, this problem exhibited the worst runtime performance over all instances. This is likely due to the large size of the RST network and the number of quality feasible paths through it. Therefore, we conclude that it is possible for the cyclic scheduling heuristic to perform well with slow runtime (Instance 12c) and it is also possible for it to perform poorly with fast runtime (Instance 20a). The frequency of failed heuristic attempts on each problem instance is summarized in Fig. 9, along with the number of nodes explored during the execution of branch-and-price procedures.

The cyclic scheduling heuristic produced a number of solutions with interesting structures. Firstly, Instance 1d was the only solution obtained exhibiting total cyclicity, that is, the patrol vessel schedules were found to form a complete cyclic permutation. The solution, depicted in Fig. 10, is composed of 8 patrol vessel schedules and can be expressed in cycle notation as $(1, 2, 3, 4, 5, 6, 7, 8)$. As the solution constitutes an 8-cycle over a planning horizon of 56 time units, this implies that self-cyclic schedules can be formed over a planning horizon of length $8 \times 56 = 448$, that is, each vessel returns to its original state after a period of 448 time units. As Problem Network 1 is symmetric (see the right-hand side of Fig. 8), it is not surprising to observe a regular pattern of behavior for the patrol vessels, that is, a vessel leaves port, patrols one region, returns to port, patrols one region, and so on. However, similar patterns of patrol vessel behavior were not typically observed for the instances on asymmetric networks. One such example is the optimal self-cyclic solution obtained for Instance 17a, which can be found in Fig. 11. Patrol Network 17 is characterized by an asymmetric structure, as shown at the top of Fig. 11, and hence, the solution to Instance 17a is dynamic with respect to vessel transitions through the network. For example, Region 02 requires six separate patrols from four different vessels over the planning horizon, and half of the patrols are of short duration.

The clustering technique was successfully applied in concert with the cyclic scheduling approach to a number of problem instances with underlying large-scale and symmetric network structures. In some cases, clustering was able to produce optimal solutions with increased runtime efficiency. Consider Instance 19c with the large-scale symmetric network depicted in Fig. 12. With the application of clustering to one half of the network, a cyclic scheduling solution was found, and by symmetry, could be duplicated on the other half of the network without having to run the algorithms again. By comparison with the value given by 'Test' on the whole network structure, the solution's optimality was established. The duplication process can be found in the bottom portion of Fig. 12.

As mentioned previously, the worst performing instance with respect to column generation at the root node was Instance 10c, and this was most likely due to the combination of a large-scale symmetric network with a long planning horizon. Clustering proved to be quite beneficial for this problem, with an optimal solution found in only 3% of the time taken to solve the RMP relaxation for the whole network at the root node. The clustering technique

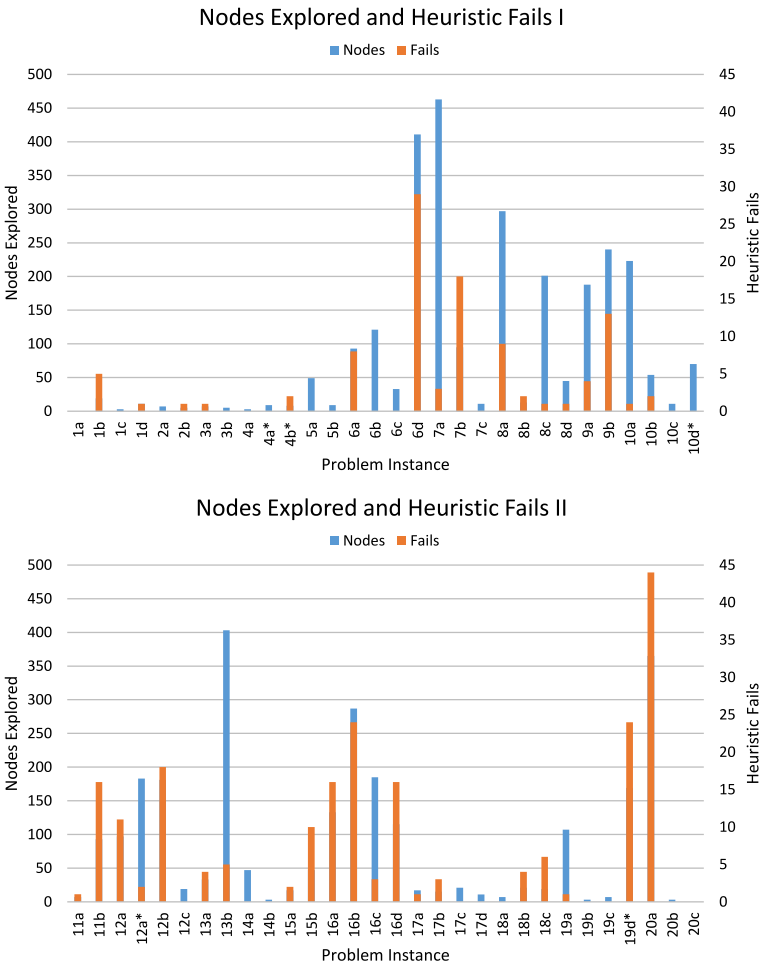


Fig. 9 Nodes explored and heuristic fails for test problem instances

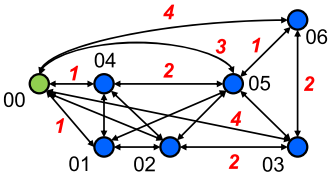
t = 0 – 28:

1. TR-00-TR-TR-03-03-03-03-03-03-03-03-03-03-03-03-03-03-03-03-03-03-TR-TR-00-TR-TR-
2. 02-02-02-02-02-02-02-02-02-02-02-02-02-02-02-02-TR-TR-00-TR-TR-03-03-03-03-03-03-
3. 05-05-05-05-05-05-05-05-05-05-TR-TR-00-TR-TR-06-06-06-06-06-06-06-06-06-06-06-06-
4. 03-03-03-03-03-TR-TR-00-TR-TR-01-01-01-01-01-01-01-01-01-01-01-01-01-01-01-01-01-
5. 00-TR-TR-04-04-04-04-04-04-04-04-04-04-04-04-04-04-04-04-04-04-TR-TR-00-TR-TR-05-
6. 06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-TR-TR-00-TR-TR-04-04-04-04-04-04-
7. 01-01-01-01-01-01-01-01-01-01-TR-TR-00-TR-TR-02-02-02-02-02-02-02-02-02-02-02-02-
8. 04-04-04-04-TR-TR-00-TR-TR-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-

t = 28 – 56:

1. 01-01-01-01-01-01-01-01-01-01-01-01-01-01-01-01-01-01-TR-TR-00-TR-TR-02-02-02-02-
2. 03-03-03-03-03-03-03-03-03-03-03-03-03-03-03-03-TR-TR-00-TR-TR-05-05-05-05-05-05-05-
3. 06-06-06-06-06-06-06-06-06-06-TR-TR-00-TR-TR-03-03-03-03-03-03-03-03-03-03-03-03-03-
4. 01-TR-TR-00-TR-TR-02-02-02-02-02-02-02-02-02-02-02-02-02-02-02-02-02-02-02-02-TR-TR-
5. 05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-05-TR-TR-00-TR-TR-06-06-
6. 04-04-04-04-04-04-04-04-04-04-04-04-04-04-TR-TR-00-TR-TR-01-01-01-01-01-01-01-01-01-
7. 02-02-02-02-02-02-02-TR-TR-00-TR-TR-04-04-04-04-04-04-04-04-04-04-04-04-04-04-04-04-
8. TR-TR-00-TR-TR-06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-06-TR-

Fig. 10 An optimal solution for Instance 1d, with $T_E = 28$ and $T_R = 0$. The solution is composed of 8 patrol vessel schedules that form a cyclic permutation over a planning horizon $T = 56$. In cycle notation, the solution can be expressed as (1, 2, 3, 4, 5, 6, 7, 8). This means that schedule 1 is continued by 2, etc., and schedule 8 is continued by 1



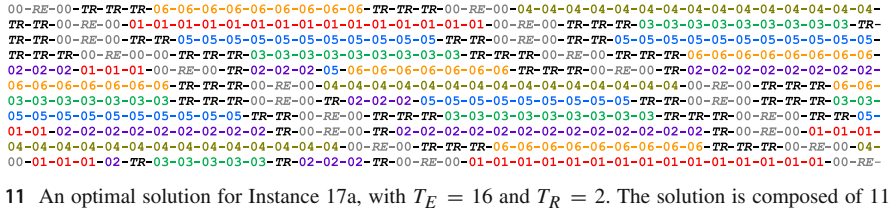
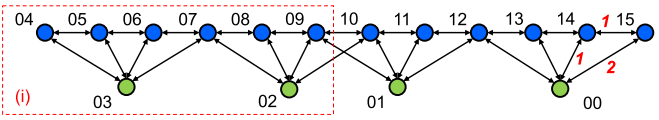


Fig. 11 An optimal solution for Instance 17a, with $T_E = 16$ and $T_R = 2$. The solution is composed of 11 self-cyclic patrol vessel schedules over a planning horizon $T = 36$. In cycle notation, the solution can be expressed as $(1)(2)\cdots(11)$



Duplicate: 03 -> 00, 02 -> 01, 09 -> 10, 08 -> 11, 07 -> 12, 06 -> 13, 05 -> 14, 04 -> 15

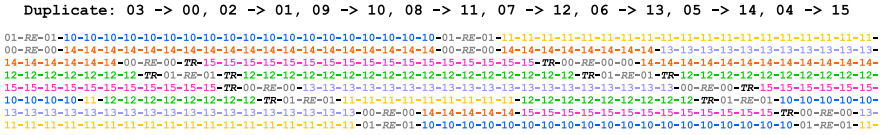
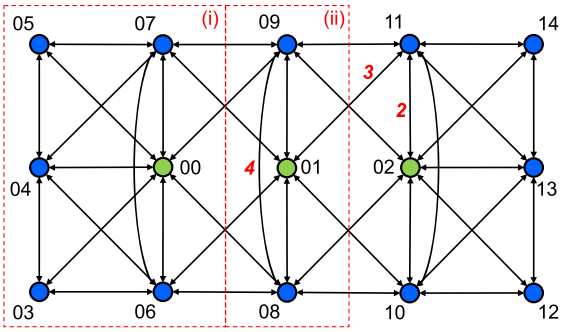


Fig. 12 An optimal solution for Instance 19c, with $T_E = 20$ and $T_R = 2$. The solution was obtained by clustering one half of the network, then duplicating the solution for the other half. In this case, each patrol vessel schedule is self-cyclic over a planning horizon $T = 44$

was applied to two portions of the patrol network, as shown in Fig. 13. The solution for cluster (i) was able to be duplicated by symmetry for the port and five regions on the right-hand side of the network, while cluster (ii), consisting of one port and two patrol regions, was very easy to solve.

The clustering approach was not always able to deliver provably optimal solutions. One problem which presented some difficulty was Instance 9b, a large-scale asymmetric network which can be found in Fig. 14. While this solution is of good quality, it is not provably optimal for the choice of patrol vessel endurance and replenishment break duration. However, an interesting feature of the solution is that there exist a few periods of port idle time, occurring for vessel schedules 3 and 15. This is noteworthy as the majority of solutions obtained for the 60 test problems did not include any idle time. As can be inferred from Fig. 14, the idle time is needed to allow the patrol vessel schedules to align in a manner that satisfies cyclicity and the complete coverage requirements.

In addition to the problems that solved at the root node (Instances 1a and 20c), we observed that the rolling horizon approach could exhibit the same effect, that is, branch-and-price was not always required to extend a solution into a new planning block of length $\bar{T}$. This phe-



t = 0 – 30:

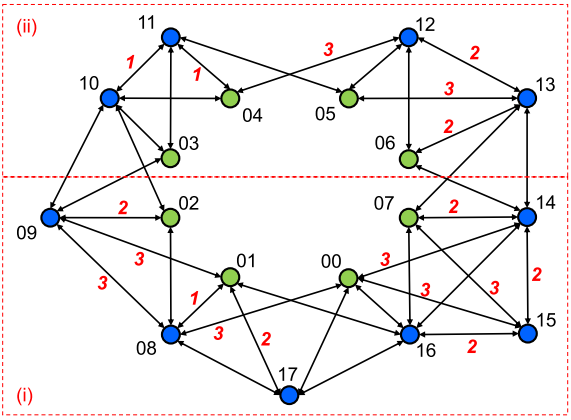
```
1.  RE-00-TR-07-07-07-07-07-07-07-07-07-07-07-07-TR-00-RE-RE-RE-00-TR-04-04-04-04-04-04-
2.  RE-RE-00-TR-TR-03-03-03-03-03-03-03-03-03-03-TR-TR-00-RE-RE-RE-00-TR-TR-03-03-03-03-03-
3.  00-RE-RE-RE-00-TR-TR-05-05-05-05-05-05-05-05-05-05-TR-TR-00-RE-RE-RE-00-TR-TR-05-05-05-
4.  TR-00-RE-RE-RE-00-TR-06-06-06-06-06-06-06-06-06-06-06-06-TR-00-RE-RE-RE-00-TR-07-07-07-
5.  07-07-07-07-TR-00-RE-RE-RE-00-TR-04-04-04-04-04-04-04-04-04-04-04-04-TR-00-RE-RE-RE-00-
6.  03-03-03-03-03-03-TR-TR-00-RE-RE-RE-00-TR-TR-03-03-03-03-03-03-03-03-03-03-TR-TR-00-RE-
7.  06-06-06-06-06-06-06-06-TR-00-RE-RE-RE-00-TR-07-07-07-07-07-07-07-07-07-07-07-07-TR-00-
8.  05-05-05-05-05-05-05-05-TR-TR-00-RE-RE-RE-00-TR-TR-05-05-05-05-05-05-05-05-05-05-TR-TR-
9.  04-04-04-04-04-04-04-04-04-04-04-04-TR-00-RE-RE-RE-00-TR-06-06-06-06-06-06-06-06-06-06-
10. 08-TR-01-RE-RE-RE-01-TR-08-08-08-08-08-08-08-08-08-08-08-08-TR-01-RE-RE-RE-01-TR-08-08-
11. 09-TR-01-RE-RE-RE-01-TR-09-09-09-09-09-09-09-09-09-09-09-09-TR-01-RE-RE-RE-01-TR-09-09-
12. 08-08-08-08-08-08-08-08-08-08-TR-01-RE-RE-RE-01-TR-08-08-08-08-08-08-08-08-08-08-08-08-
13. 09-09-09-09-09-09-09-09-09-09-TR-01-RE-RE-RE-01-TR-09-09-09-09-09-09-09-09-09-09-09-09-
14. RE-02-TR-11-11-11-11-11-11-11-11-11-11-11-11-TR-02-RE-RE-RE-02-TR-13-13-13-13-13-13-13-
15. RE-RE-02-TR-TR-12-12-12-12-12-12-12-12-12-12-TR-TR-02-RE-RE-RE-02-TR-TR-12-12-12-12-12-
16. 02-RE-RE-RE-02-TR-TR-14-14-14-14-14-14-14-14-14-14-TR-TR-02-RE-RE-RE-02-TR-TR-14-14-14-
17. TR-02-RE-RE-RE-02-TR-10-10-10-10-10-10-10-10-10-10-10-10-TR-02-RE-RE-RE-02-TR-11-11-11-
18. 11-11-11-11-TR-02-RE-RE-RE-02-TR-13-13-13-13-13-13-13-13-13-13-13-13-TR-02-RE-RE-RE-02-
19. 12-12-12-12-12-12-TR-TR-02-RE-RE-RE-02-TR-TR-12-12-12-12-12-12-12-12-12-12-TR-TR-02-RE-
20. 10-10-10-10-10-10-10-10-TR-02-RE-RE-RE-02-TR-11-11-11-11-11-11-11-11-11-11-11-11-TR-02-
21. 14-14-14-14-14-14-14-14-TR-TR-02-RE-RE-RE-02-TR-TR-14-14-14-14-14-14-14-14-14-14-TR-TR-
22. 13-13-13-13-13-13-13-13-13-13-13-13-TR-02-RE-RE-RE-02-TR-10-10-10-10-10-10-10-10-10-10-
```

t = 30 – 60:

```
1.  04-04-04-04-04-04-TR-00-RE-RE-RE-00-TR-06-06-06-06-06-06-06-06-06-06-06-06-TR-00-RE-RE-
2.  03-03-03-03-03-03-TR-TR-00-RE-RE-RE-00-TR-TR-03-03-03-03-03-03-03-03-03-03-TR-TR-00-RE-
3.  05-05-05-05-05-05-05-05-TR-TR-00-RE-RE-RE-00-TR-TR-05-05-05-05-05-05-05-05-05-05-TR-TR-
4.  07-07-07-07-07-07-07-07-07-07-TR-00-RE-RE-RE-00-TR-04-04-04-04-04-04-04-04-04-04-04-04-
5.  TR-06-06-06-06-06-06-06-06-06-06-06-06-TR-00-RE-RE-RE-00-TR-07-07-07-07-07-07-07-07-07-
6.  RE-RE-00-TR-TR-03-03-03-03-03-03-03-03-03-03-TR-TR-00-RE-RE-RE-00-TR-TR-03-03-03-03-03-
7.  RE-RE-RE-00-TR-04-04-04-04-04-04-04-04-04-04-04-04-TR-00-RE-RE-RE-00-TR-06-06-06-06-06-
8.  00-RE-RE-RE-00-TR-TR-05-05-05-05-05-05-05-05-05-05-TR-TR-00-RE-RE-RE-00-TR-TR-05-05-05-
9.  06-06-TR-00-RE-RE-RE-00-TR-07-07-07-07-07-07-07-07-07-07-07-07-07-07-TR-00-RE-RE-RE-00-TR-04-
10. 08-08-08-08-08-08-08-08-08-08-TR-01-RE-RE-RE-01-TR-08-08-08-08-08-08-08-08-08-08-08-08-
11. 09-09-09-09-09-09-09-09-09-09-TR-01-RE-RE-RE-01-TR-09-09-09-09-09-09-09-09-09-09-09-09-
12. TR-01-RE-RE-RE-01-TR-08-08-08-08-08-08-08-08-08-08-08-08-TR-01-RE-RE-RE-01-TR-08-08-08-
13. TR-01-RE-RE-RE-01-TR-09-09-09-09-09-09-09-09-09-09-09-09-TR-01-RE-RE-RE-01-TR-09-09-09-
14. 13-13-13-13-13-13-TR-02-RE-RE-RE-02-TR-10-10-10-10-10-10-10-10-10-10-10-10-TR-02-RE-RE-
15. 12-12-12-12-12-12-TR-TR-02-RE-RE-RE-02-TR-TR-12-12-12-12-12-12-12-12-12-12-TR-TR-02-RE-
16. 14-14-14-14-14-14-14-14-TR-TR-02-RE-RE-RE-02-TR-TR-14-14-14-14-14-14-14-14-14-14-TR-TR-
17. 11-11-11-11-11-11-11-11-11-11-TR-02-RE-RE-RE-02-TR-13-13-13-13-13-13-13-13-13-13-13-13-
18. TR-10-10-10-10-10-10-10-10-10-10-10-10-TR-02-RE-RE-RE-02-TR-11-11-11-11-11-11-11-11-11-
19. RE-RE-02-TR-TR-12-12-12-12-12-12-12-12-12-12-TR-TR-02-RE-RE-RE-02-TR-TR-12-12-12-12-12-
20. RE-RE-RE-02-TR-13-13-13-13-13-13-13-13-13-13-13-13-TR-02-RE-RE-RE-02-TR-10-10-10-10-10-
21. 02-RE-RE-RE-02-TR-TR-14-14-14-14-14-14-14-14-14-14-TR-TR-02-RE-RE-RE-02-TR-TR-14-14-14-
22. 10-10-TR-02-RE-RE-RE-02-TR-11-11-11-11-11-11-11-11-11-11-11-11-TR-02-RE-RE-RE-02-TR-13-
```

Fig. 13 An optimal solution for Instance 10c, with $T_E = 16$ and $T_R = 4$. The solution was obtained by clustering two portions of the network, then duplicating the solution to (i) by symmetry. In this case, each patrol vessel schedule is self-cyclic over a planning horizon $T = 60$

nomenon occurred with regularity when the rolling horizon approach was applied to Instances 4b* and 10d*. We surmise that this is due to a *total unimodularity* property of the final primal linear programs upon termination of the column generation procedure. (Refer to Chapter III.1 of Nemhauser and Wolsey (1988) for a detailed treatment of total unimodularity.) Further



t = 0 – 28:

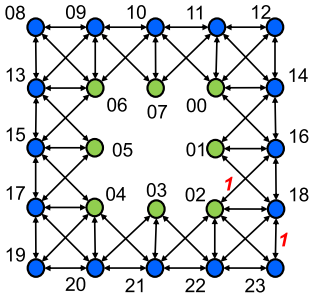
```
1.  10-10-10-10-03-RE-RE-RE-03-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-
2.  13-13-13-13-TR-06-RE-RE-RE-06-TR-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-
3.  09-09-09-09-09-TR-02-RE-RE-RE-02-TR-09-09-09-09-09-09-09-09-09-09-09-09-09-09-09-09-09-09-09-
4.  15-15-15-15-15-15-15-TR-TR-07-RE-RE-RE-07-TR-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-
5.  17-17-17-17-17-17-17-17-TR-00-RE-RE-RE-00-16-16-TR-15-15-15-15-15-15-15-15-15-15-15-15-15-15-
6.  11-11-11-11-11-11-11-11-11-11-11-11-04-RE-RE-RE-04-11-11-11-11-11-11-11-11-11-11-11-11-11-11-
7.  12-TR-13-13-13-13-13-13-13-13-13-13-TR-06-RE-RE-RE-06-TR-12-12-12-12-12-12-12-TR-13-
8.  08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-01-RE-RE-RE-01-08-08-08-08-08-08-08-
9.  12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-05-RE-RE-RE-05-12-12-12-
10. TR-TR-09-09-09-09-09-09-09-09-09-09-TR-TR-08-08-08-08-08-08-08-08-01-RE-RE-RE-01-
11. 11-10-10-10-10-10-10-10-10-10-11-11-11-11-11-11-11-11-11-11-11-11-11-11-04-RE-RE-RE-04-
12. RE-RE-RE-07-TR-TR-15-15-15-15-15-15-15-15-15-15-15-15-TR-16-16-16-16-16-16-16-16-TR-07-
13. RE-RE-RE-00-TR-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-TR-00-
14. 16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-00-RE-RE-RE-00-16-16-16-
15. 14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-TR-TR-00-RE-RE-RE-00-TR-17-17-17-
```

t = 28 – 56:

```
1.  10-10-10-10-03-RE-RE-RE-03-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-10-
2.  13-13-13-13-TR-06-RE-RE-RE-06-TR-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-13-
3.  09-09-09-09-09-TR-02-RE-RE-RE-02-02-02-TR-09-09-09-09-09-09-09-09-09-09-09-09-09-09-09-09-09-
4.  14-14-14-14-14-14-14-14-TR-07-RE-RE-RE-07-TR-TR-15-15-15-15-15-15-15-15-15-15-15-15-15-15-15-
5.  15-15-15-15-15-15-15-TR-TR-00-RE-RE-RE-00-TR-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-
6.  11-11-11-11-11-11-11-11-11-11-11-04-RE-RE-RE-04-11-11-11-11-11-11-11-11-11-11-11-11-11-11-
7.  13-13-13-13-13-13-13-13-13-13-13-13-TR-06-RE-RE-RE-06-TR-12-12-12-12-12-12-12-12-12-
8.  08-TR-TR-09-09-09-09-09-09-09-09-09-09-09-TR-02-RE-RE-RE-02-TR-08-08-08-08-08-08-
9.  12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-12-05-RE-RE-RE-05-12-12-12-
10. 08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-08-01-RE-RE-RE-01-
11. 11-11-11-10-10-10-10-10-10-10-11-11-11-11-11-11-11-11-11-11-11-11-11-11-04-RE-RE-RE-04-
12. RE-RE-RE-07-TR-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-14-TR-07-
13. RE-RE-RE-00-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-16-00-
14. 16-16-16-16-16-TR-15-15-15-15-15-15-15-15-15-15-15-TR-TR-07-RE-RE-RE-07-TR-14-14-14-
15. 17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-17-TR-00-RE-RE-RE-00-00-00-16-16-
```

Fig. 14 A solution for Instance 9b, with $T_E = 24$ and $T_R = 4$. The solution is not provably optimal and was obtained by partitioning the network into two clusters as shown. In this case, patrol vessel schedules 1–13 are self-cyclic over a planning horizon $T = 56$, while there is a small sub-cycle on schedules 14 and 15 over the same horizon

investigation is required to determine if (and how) this structure is induced from the underlying network topology, planning horizon length, endurance and replenishment break duration. On the other hand, there are cases like Instances 4a* and 19d* that delivered root node solutions about half the time, while Instance 12a* did not exhibit the integrality property at all (Fig. 15 shows the partial structure of the solution to Instance 12a*). On the occasions that branch-and-price was invoked, the rolling horizon heuristic performed well in the majority of cases. The worst result was run 6 of Instance 19d*, where 15 failed attempts to extend the solution occurred.



t = 0 – 30:

```
1.  20-20-20-20-20-20-20-20-20-20-03-21-21-21-21-21-21-21-21-21-02-23-23-23-23-23-23-23-23-23-
2.  15-15-15-15-15-15-15-04-19-19-19-19-19-19-19-19-19-19-19-04-21-21-21-21-21-21-21-21-21-21-
3.  11-11-11-11-11-11-11-11-00-14-14-14-14-14-14-14-14-14-01-16-16-16-16-16-16-16-16-16-16-16-
4.  13-13-13-13-13-13-06-09-09-09-09-09-09-09-09-09-09-07-10-10-10-10-10-10-10-10-10-10-10-10-
5.  17-17-17-17-05-13-13-13-13-13-13-13-13-13-13-06-08-08-08-08-08-08-08-08-08-08-08-08-06-15-
6.  02-21-21-21-21-21-21-21-21-21-21-04-17-17-17-17-17-17-17-17-17-05-13-13-13-13-13-13-13-13-
7.  09-09-09-09-09-09-09-09-06-08-08-08-08-08-08-08-08-08-08-06-15-15-15-15-15-15-15-15-15-15-
8.  18-18-18-18-18-18-18-18-18-18-18-02-22-22-22-22-22-22-22-22-22-22-22-03-20-20-20-20-20-20-20-
9.  08-08-08-08-08-08-08-08-08-08-08-06-10-10-10-10-10-10-10-10-10-10-00-12-12-12-12-12-12-12-12-
10. 14-14-14-14-14-14-14-14-14-14-01-16-16-16-16-16-16-16-16-16-16-02-22-22-22-22-22-22-22-22-22-
11. 22-22-02-23-23-23-23-23-23-23-23-23-23-02-18-18-18-18-18-18-18-18-18-18-18-01-14-14-14-14-
12. 23-23-23-23-23-23-23-02-18-18-18-18-18-18-18-18-01-14-14-14-14-14-14-14-14-14-14-14-01-16-
13. 01-16-16-16-16-16-16-16-16-16-16-16-02-23-23-23-23-23-23-23-23-23-23-02-18-18-18-18-18-18-
14. 06-10-10-10-10-10-10-10-10-10-10-10-00-12-12-12-12-12-12-12-12-12-12-12-00-11-11-11-11-11-
15. 21-21-04-17-17-17-17-17-17-17-17-17-17-05-13-13-13-13-13-13-13-13-13-13-13-06-08-08-08-
16. 03-22-22-22-22-22-22-22-22-22-22-22-03-20-20-20-20-20-20-20-20-20-20-20-04-19-19-19-19-19-
17. 19-19-19-19-19-19-19-19-19-19-04-20-20-20-20-20-20-04-15-15-15-15-15-15-05-17-17-17-17-17-17-17-17-
18. 10-10-10-10-06-15-15-15-15-15-15-15-15-15-15-15-04-19-19-19-19-19-19-19-19-19-19-19-04-21-
19. 12-12-12-12-12-00-11-11-11-11-11-11-11-11-11-11-11-07-09-09-09-09-09-09-09-09-09-09-07-10-
20. 16-16-16-00-12-12-12-12-12-12-12-12-12-12-12-00-11-11-11-11-11-11-11-11-11-11-11-11-07-09-09-09-
```

t = 30 – 60:

```
1.  23-23-02-18-18-18-18-18-18-18-18-18-18-18-18-01-14-14-14-14-14-14-14-14-14-01-16-16-16-16-16-
2.  02-23-23-23-23-23-23-23-23-23-23-23-02-18-18-18-18-18-18-18-18-18-18-18-01-14-14-14-14-14-14-14-
3.  02-22-22-22-22-22-22-22-22-22-22-03-20-20-20-20-20-20-20-20-20-20-20-04-19-19-19-19-19-19-19-
4.  00-12-12-12-12-12-12-12-12-12-12-12-12-00-11-11-11-11-11-11-11-11-11-11-11-11-07-09-09-09-09-09-09-09-
5.  15-15-15-15-15-15-15-15-15-05-17-17-17-17-17-17-17-17-17-17-17-17-05-13-13-13-13-13-05-15-15-15-
6.  13-13-13-13-06-08-08-08-08-08-08-08-08-08-08-08-06-10-10-10-10-10-10-10-10-10-00-12-12-12-
7.  05-17-17-17-17-17-17-17-17-17-17-17-17-05-13-13-13-13-13-13-13-13-13-13-13-06-08-08-08-08-08-08-08-08-
8.  20-20-20-20-04-19-19-19-19-19-19-19-19-19-19-19-19-04-21-21-21-21-21-21-21-21-21-04-17-17-17-
9.  12-12-00-11-11-11-11-11-11-11-11-11-11-07-09-09-09-09-09-09-09-09-09-09-09-07-10-10-10-10-10-
10. 22-22-03-20-20-20-20-20-20-20-20-20-20-04-19-19-19-19-19-19-19-19-19-19-19-04-21-21-21-21-21-
11. 14-14-14-14-14-14-01-16-16-16-16-16-16-16-16-16-16-16-02-22-22-22-22-22-22-22-22-22-22-22-
12. 16-16-16-16-16-16-16-16-02-22-22-22-22-22-22-22-22-22-22-22-03-20-20-20-20-20-20-20-20-20-
13. 18-18-18-18-01-14-14-14-14-14-14-14-14-14-14-14-01-16-16-16-16-16-16-16-16-16-16-02-23-23-23-
14. 11-11-11-11-07-09-09-09-09-09-09-09-09-09-09-09-06-15-15-15-15-15-15-15-15-15-15-15-06-08-
15. 08-08-08-08-08-08-06-10-10-10-10-10-10-10-10-10-10-10-00-12-12-12-12-12-12-12-12-12-12-12-00-11-
16. 19-19-19-19-19-19-04-21-21-21-21-21-21-21-21-21-21-21-02-23-23-23-23-23-23-23-23-23-23-02-22-
17. 17-17-05-13-13-13-13-13-13-13-13-13-13-13-06-08-08-08-08-08-08-08-08-08-06-13-13-13-13-13-
18. 21-21-21-21-21-21-21-21-21-21-02-23-23-23-23-23-23-23-23-23-23-23-02-18-18-18-18-18-18-18-18-18-
19. 10-10-10-10-10-10-10-10-00-12-12-12-12-12-12-12-12-12-12-12-00-11-11-11-11-11-11-11-11-11-11-
20. 09-09-09-09-09-09-06-15-15-15-15-15-15-15-15-15-15-15-05-17-17-17-17-17-17-17-17-17-17-17-
```

Fig. 15 An optimal solution for Instance 12a* using 20 patrol boats, where $T_E = 12$ and $T_R = 0$. The problem was solved with the rolling horizon approach over a period of 192 time units, but only the first 60 time units are shown

The rolling horizon approach proved to be effective in establishing an optimal solution to Instance 12a*. Note from Table 5 that this problem shares the same inputs ($T_E = 12, T_R = 0$) with Instance 12a, and from Table 7, that the cyclic scheduling solution obtained for Instance 12a using 21 vessels was not provably optimal. With the rolling horizon approach, it was found that a fleet of size 20 could satisfy the complete coverage requirements in the long term ($T_{\text{sol}} = 192$). A depiction of Patrol Network 12 and a portion of the solution to Instance 12a* obtained with the rolling horizon approach can be found in Fig. 15. The solution structure is interesting in that there is no discernible pattern to the patrol vessel movements throughout the network. This raises the question of whether cyclic schedules will always exist for an

optimal fleet size, or if there exists a minimum planning horizon length over which an optimal cyclic schedule can be found. A further noteworthy feature of the rolling horizon solution to Instance 12a* is that patrol vessels sometimes return to port for replenishment at half resource capacity. For example, by inspection of vessel schedule 17, Regions 20 and 15 are each patrolled for 4 time units before the vessel undergoes replenishment at port. Given that the solution contains a small amount of overlapping patrol, this feature is clearly necessary to ensure complete coverage.

6 Concluding remarks

In this paper, we have presented new modeling and solution approaches to the Patrol Boat Scheduling Problem with Complete Coverage (PBSPCC). Using a resource-space-time (RST) network, we defined a path based linear program for the application of an enhanced branch-and-price solution approach. Temporal and spatial reduction techniques were introduced to alleviate the slower runtime performance of the branch-and-price approach on large-scale problem instances of the PBSPCC. These techniques included cyclic scheduling and rolling horizon approaches, along with spatial clustering. The full suite of techniques was applied to a range of test problem instances to arrive at a set of 60 computational benchmarks. Of the solutions obtained, 70% were provably optimal for either an indefinite or long-term planning horizon. Of the remaining 30% that were not provably optimal, all but two solutions used one surplus patrol vessel compared to the rounded-up lower bound derived from the linear programming relaxation. There are multiple factors that influence the overall runtime performance of the solution algorithms. However, the most decisive factors affecting efficiency appear to be the planning horizon length and the number of patrol regions. This observation, at the very least, highlights the utility of applying the problem reduction techniques introduced in this paper. By scaling down the temporal and spatial dimensions, larger problems can be solved to optimality in a piecemeal fashion much more efficiently. In the temporal domain, the decomposition techniques deliver an additional feature which proves to be quite useful, namely, the ability to find solutions which can be extended indefinitely or satisfied in the long term.

6.1 Suggestions for future work

For future studies of the PBSPCC, alternative modeling and solution approaches could be developed to improve the scaling performance of the column generation subproblem. Candidate modeling and algorithmic approaches include the *resource-constrained shortest path problem with replenishment* examined by Smith et al. (2012), and the *multi-trip elementary shortest path problem with resource constraints* which has been studied by Akca et al. (2010). (The resource-constrained shortest path problem was first proffered by Desrochers (1986) for a column generation approach to bus driver scheduling. For a directed acyclic graph with negative arc costs, the resource-constrained shortest path problem can be solved using an extended label-correcting or label-setting algorithm (Desrochers and Soumis 1988). See Irnich and Desaulniers (2005) for a comprehensive survey of solution approaches to resource-constrained shortest path problems.) Another possibility for future work is to investigate the applicability of constraint programming (CP) for the subproblem solver, as this paradigm has shown considerable promise on certain combinatorial problems (Easton et al. 2004; Gualandi and Malucelli 2009; He and Qu 2012).

A natural extension of the PBSPCC is to consider a heterogeneous fleet of patrol boats, where each boat is a member of a vessel class, say $k \in \mathcal{K}$. Under this schema, there are $|\mathcal{K}|$ separate column generation subproblems to be solved, and we propose branching on the *patrol arcs* of the underlying RST networks. If $(u, v) \in A_{\mathrm{P}}^k$ is fractional and selected for branching, then branch (a) is enforced by removing (u, v) from the network and deleting all path variables λ_p^k from the master problem that use arc (u, v). For branch (b), all arcs (x, y) such that $\phi_k(x, y) = \phi_k(u, v)$ and $(x, y) \neq (u, v)$ are removed from the network, along with all arcs in the sets $A_+^k(y)$ and $A_-^k(x)$. In addition, all path variables that do not use (u, v) are removed from the master problem. Another natural extension of the PBSPCC is the incorporation of a crew scheduling component. For example, this might be approached in a similar manner to that of Fischetti et al. (2001), who examined crew scheduling in the vehicle routing context. Furthermore, incorporating a regular maintenance cycle (Hahn and Newman 2008) for the patrol boats would be an interesting research question to address in the future.

On the problem of finding cyclic schedules, future work could consider the subproblem pricing strategy introduced by Andersen et al. (2011) for service network design with asset management. In this case, the pricing subproblem seeks path cycles over a time-space network by implementing a label-correcting dynamic programming algorithm. The algorithm uses two labels: the first label corresponds to the reduced cost of a path and the second ensures that the first vertex visited along the path matches the final vertex visited. Such a method could be implemented over the RST network design, where the labeling procedure ensures that if $v = (i, \hat{a}, 0)$ is the first vertex visited along a path for some $i \in \mathcal{V}_{\mathrm{pat}}$ and $\hat{a} \in D_i$, then the last vertex must be $u = (i, \hat{a}, T)$, where $f(s, v) = (u, \tau)$. Implementing the labeling algorithm of Andersen et al. (2011) would require an overhaul of the shortest path approach that we have proffered for the pricing subproblem. It is important to note, however, that a cyclic scheduling solution over a planning horizon does not necessarily imply that each path has the same initial and termination conditions.

Acknowledgements Paul Chircop would like to acknowledge his co-authors for their expert supervision of his doctoral studies from 2010–2016 at the University of New South Wales. Paul Chircop and Timothy Surendonk wish to thank Dr Maria Athanassenas (Group Leader Maritime Mathematical Sciences, Joint and Operations Analysis Division) of the Defence Science and Technology Group for her generous encouragement and support. Toby Walsh was funded by the European Research Council under the Horizon 2020 Programme via AMPLify 670077.

References

Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network flows: Theory, algorithms, and applications*. Prentice Hall.

Akca, Z., Ralphs, T. K., & Berger, R. T. (2010). Solution methods for the multi-trip elementary shortest path problem with resource constraints. Optimization Online. http://www.optimization-online.org/DB_HTML/2011/03/2962.html.

Andersen, J., Christiansen, M., Crainic, T. G., & Grønhaug, R. (2011). Branch and price for service network design with asset management constraints. *Transportation Science, 45*(1), 33–49.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Va, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research, 46*(3), 316–329.

Ben Amor, H., & Valério de Carvalho, J. (2005). Cutting stock problems. In G. Desaulniers, J. Desrosiers, & M. M. Solomon (Eds.), *Column generation* (pp. 131–161). Springer.

Brown, G. G., Dell, R. F., & Farmer, R. A. (1996). Scheduling Coast Guard district cutters. *Interfaces, 26*(2), 59–72.

Çapar, İ., Keskin, B. B., & Rubin, P. A. (2015). An improved formulation for the maximum coverage patrol routing problem. *Computers & Operations Research, 59,* 1–10.

Chircop, P. A. (2017). *Column generation approaches to patrol asset scheduling with complete and maximum coverage requirements* (Doctoral dissertation, University of New South Wales). https://doi.org/1959.4/57656

Chircop, P. A., & Surendonk, T. J. (2021). On integer linear programming formulations of a patrol boat scheduling problem with complete coverage requirements. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, 18*(4), 429–439.

Chircop, P. A., Surendonk, T. J., van den Briel, M. H. L., & Walsh, T. (2013). A column generation approach for the scheduling of patrol boats to provide complete patrol coverage. In J. Piantadosi, R. S. Anderssen, & J. Boland (Eds.), Proceedings of the 20th international congress on modelling and simulation (pp. 1110–1116). Modelling and Simulation Society of Australia and New Zealand.

Chircop, P. A., Surendonk, T. J., van den Briel, M. H. L., & Walsh, T. (2021). A branch-and-price framework for the maximum covering and patrol routing problem. In A. T. Ernst, S. Dunstall, R. García-Flores, M. Grobler, & D. Marlow (Eds.), *Data and decision sciences in action 2* (pp. 59–80). Springer.

Christiansen, M., Fagerholt, K., & Ronen, D. (2004). Ship routing and scheduling: Status and perspectives. *Transportation Science, 38*(1), 1–18.

Christiansen, M., & Nygreen, B. (1998). A method for solving ship routing problems with inventory constraints. *Annals of Operations Research, 81*, 357–378.

Chvatal, V. (1983). *Linear programming*. Macmillan.

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Integer programming*. Springer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd edn.). The MIT Press.

Dantzig, G. B., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research, 8*(1), 101–111.

Darby-Dowman, K., Fink, R. K., Mitra, G., & Smith, J. W. (1995). An intelligent system for US Coast Guard cutter scheduling. *European Journal of Operational Research, 87*(3), 574–585.

Desaulniers, G., Desrosiers, J., & Solomon, M. M. (2002). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In *Essays and surveys in metaheuristics* (pp. 309–324). Springer.

Desrochers, M. (1986). *La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes* (Unpublished doctoral dissertation). Centre de Recherche sur les Transports, Université de Montréal.

Desrochers, M., & Soumis, F. (1988). A generalized permanent labeling algorithm for the shortest-path problem with time windows. *Information Systems and Operations Research, 26*(3), 191–212.

Dewil, R., Vansteenwegen, P., Cattrysse, D., & Oudheusden, D. V. (2015). A minimum cost network flow model for the maximum covering and patrol routing problem. *European Journal of Operational Research, 247*(1), 27–36.

Easton, K., Nemhauser, G., & Trick, M. (2004). CP based branch-and-price. In M. Milano (Ed.), *Constraint and integer programming* (pp. 207–231). Springer.

Fang, F., Stone, P., & Tambe, M. (2015). When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *Proceedings of the 24th international joint conference on artificial intelligence* (pp. 2589–2595). AAAI Press.

Fischetti, M., Lodi, A., Martello, S., & Toth, P. (2001). A polyhedral approach to simplified crew scheduling and vehicle scheduling problems. *Management Science, 47*(6), 833–850.

Ford, L. R., & Fulkerson, D. R. (1958). A suggested computation for maximal multi-commodity network flows. *Management Science, 5*(1), 97–101.

Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research, 9*(6), 849–859.

Gualandi, S., & Malucelli, F. (2009). Constraint programming-based column generation. *4OR*, *7*(2), 113–137.

Hahn, R. A., & Newman, A. M. (2008). Scheduling United States Coast Guard helicopter deployment and maintenance at Clearwater Air Station, Florida. *Computers & Operations Research, 35*(6), 1829–1843.

He, F., & Qu, R. (2012). A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research, 39*(12), 3331–3343.

Horn, M. E. T., Jiang, H., & Kilby, P. (2006). Scheduling patrol boats and crews for the Royal Australian Navy. *Journal of the Operational Research Society, 58*(10), 1284–1293.

Hsieh, Y. C., You, P. S., Lee, P. J., & Lee, Y. C. (2015). A novel encoding scheme based evolutionary approach for the bi-objective grid patrol routing problem with multiple vehicles. *Scientia Iranica. Transaction B, Mechanical Engineering, 22*(4), 1576–1585.

Irnich, S., & Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, & M. M. Solomon (Eds.), *Column generation* (pp. 33–65). Springer.

Keskin, B. B., Li, S., Steil, D., & Spiller, S. (2012). Analysis of an integrated maximum covering and patrol routing problem. *Transportation Research Part E: Logistics and Transportation Review, 48*(1), 215–232.
Kim, J., Song, B. D., & Morrison, J. R. (2013). On the scheduling of systems of UAVs and fuel service stations for long-term mission fulfillment. *Journal of Intelligent & Robotic Systems, 70*(1–4), 347–359.
Lübbecke, M. E. (2001). *Engine scheduling by column generation* (Doctoral dissertation). Braunschweig University of Technology. https://doi.org/10.24355/dbbs.084-200511080100-720
Lübbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research, 53*(6), 1007–1023.
Millar, H. H., & Russell, S. N. (2012). A model for fisheries patrol dispatch in the Canadian Atlantic offshore fishery. *Ocean & Coastal Management, 60*, 48–55.
Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and combinatorial optimization*. Wiley.
Nulty, W. G., & Ratliff, H. D. (1991). Interactive optimization methodology for fleet scheduling. *Naval Research Logistics, 38*(5), 669–677.
Shieh, E., Jain, M., Jiang, A. X., & Tambe, M. (2013). Efficiently solving joint activity based security games. In *Proceedings of the 23rd international joint conference on artificial intelligence* (pp. 346–352). AAAI Press.
Smith, O. J., Boland, N., & Waterer, H. (2012). Solving shortest path problems with a weight constraint and replenishment arcs. *Computers & Operations Research, 39*(5), 964–984.
Surendonk, T. J., & Chircop, P. A. (2020a). *Detailed complexity proofs for the patrol boat scheduling problem with complete coverage* (Technical note no. DST-Group-TN-2026). Canberra, Australian Capital Territory: Joint and Operations Analysis Division, Defence Science and Technology Group.
Surendonk, T. J., & Chircop, P. A. (2020b). On the computational complexity of the patrol boat scheduling problem with complete coverage. *Naval Research Logistics, 67*(4), 289–299.
Vanderbeck, F. (1994). *Decomposition and column generation for integer programs* (Doctoral dissertation). Universite Catholique de Louvain. https://doi.org/2078.1/205381
Vanderbeck, F. (2000). On Dantzig–Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research, 48*(1), 111–128.
Vanderbeck, F. (2005). Implementing mixed integer column generation. In G. Desaulniers, J. Desrosiers, & M. M. Solomon (Eds.), *Column generation* (pp. 331–358). Springer.
Wagner, M. R., & Radovilsky, Z. (2012). Optimizing boat resources at the US Coast Guard: Deterministic and stochastic models. *Operations Research, 60*(5), 1035–1049.
Zadeh, H. S., Storey, I., & Lenarcic, J. (2009). NaMOS; Scheduling patrol boats and crews for the Royal Australian Navy. In *2009 IEEE aerospace conference* (pp. 1–12).