Resource-Constrained Scheduling for Maritime Traffic Management

Lucas Agussurja, Akshat Kumar, Hoong Chuin Lau

{lagussurja, akshatkumar, hclau}@smu.edu.sg
School of Information Systems
Singapore Management University

Abstract

We address the problem of mitigating congestion and preventing hotspots in busy water areas such as Singapore Straits and port waters. Increasing maritime traffic coupled with narrow waterways makes vessel schedule coordination for just-in-time arrival critical for navigational safety. Our contributions are: 1) We formulate the maritime traffic management problem based on the real case study of Singapore waters; 2) We model the problem as a variant of the resource-constrained project scheduling problem (RCPSP), and formulate mixed-integer and constraint programming (MIP/CP) formulations; 3) To improve the scalability, we develop a combinatorial Benders (CB) approach that is significantly more effective than standard MIP and CP formulations. We also develop symmetry breaking constraints and optimality cuts that further enhance the CB approach's effectiveness; 4) We develop a realistic maritime traffic simulator using electronic navigation charts of Singapore Straits. Our scheduling approach on synthetic problems and a real 55-day AIS dataset results in significant reduction of the traffic density while incurring minimal delays.

1   Introduction

The Straits of Malacca and Singapore is one of the busiest shipping areas in the world providing the shortest route between the Indian Ocean and the South China Sea. The vessel traffic in Straits has been consistently increasing, with very large container vessels (VLCC) one of the fastest growing segments (Hand 2017). Congestion around busy port waters critically affects safety of navigation as it limits manoeuvrability for large vessels, leads to increased cross traffic, and requires vessels to frequently change course to avoid other vessels (Segar 2015). Recently, several vessel collisions causing oil spill in Straits have also threatened the environment (Lim 2017; Tan 2017). Therefore, our goal is to study and develop maritime traffic coordination techniques to mitigate congestion and prevent hotspots for safety of navigation in busy waterways such as Singapore Straits. The framework and techniques we develop are fairly general and are also applicable to other heavily trafficked ports.

The water area used as the case study, shown in figure 1, is based on the e-navigation charts of the Straits. Different features in this area include the *traffic separation scheme* (TSS)

which is a set of mandatory one-direction routes designed to reduce collision risk among vessels transitioning through the Straits, and *fairway* where vessels may travel in any direction. Other features include *berths*, *anchorages*, and pilot boarding grounds. As fairway is relatively narrow, vessels entering it require a local *pilot* onboard from pilot boarding grounds. The water area is also divided into smaller zones by dividing the TSS, fairway, and anchorages into smaller parts. Such zones, also shown in figure 1, form the basis of our scheduling approach.

We model each vessel as performing a sequence of activities in TSS and fairway. Each such activity requires certain shared resources (e.g., physical sea space, pilots, berths among others). The activity that directly affects congestion is the navigation activity. For a vessel, the traversing of a zone in the Straits/port waters is considered an activity. E.g., the second activity of vessel 1 is navigating the zone 10EB in figure 1. Each vessel is modeled as performing a sequence of such navigation activities with no precedence relation with other vessels. To improve safety of navigation, for each zone, we limit the maximum number of vessels that can be simultaneously present in the zone. Such a limit can be directly derived from data (e.g., by finding the maximum number of vessels present at any instant in the zone from historical data), or provided as parameter from domain experts. Our goal is to optimize the schedule of vessels (when should each activity start and end) such that different resource limits are respected and the resulting schedule has maximum efficiency (or least possible delay).

The above problem can be considered as a restricted variant of RCPSP (Kolisch and Hartmann 1999), with dependency graph consisting of a set of linear dependencies per vessel. Nevertheless, the problem is still challenging. There is no lag/slack time between any two consecutive activities as vessels cannot stop mid way during navigation. This can seen as a special case of RCPSP/max with time lags set to zero (Bartusch, Möhring, and Radermacher 1988). In contrast to *fixed* activity durations in RCPSPs, the vessels are allowed to speed up or slow down on the TSS/fairway up to certain limits depending on the zones. This translates to activities having minimum and maximum durations, where the actual durations are part of the *decision variables*—duration of an activity directly translates into an average speed as the length of the zone is known. This is the crucial difference
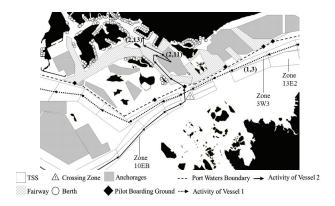
Figure 1: Features within Singapore Straits and Port Waters (only partial Straits/port waters shown). Solid black regions are landmass

from an RCPSP where activity durations are either fixed or follow some given distribution, whereas in our case, activity durations are output of the scheduler.

Our contributions are as follows. *First*, we formalize the maritime traffic management (MTM) problem based on the real case study of Singapore waters and with inputs from the maritime port authority (MPA) of Singapore. *Second*, We model the problem as a variant of RCPSP, and formulate mixed-integer and constraint programming (MIP/CP) formulations for the MTM problem. The MIP formulation is an extension of the event-based continuous-time formulation of Koné et al. (2011) for RCPSP. However, extending this formulation for the MTM problem requires using Big M constraints, which make the LP relaxation weak, and thus, the resulting MIP is not scalable.

Our next main contribution is developing decomposition approaches such as Benders (Benders 1962) and combinatorial Benders (CB) (Codato and Fischetti 2006) for the MTM problem. Benders decomposition is a classical technique applied to several real world problems such as airline scheduling (Merciera, Cordeau, and Soumis 2005) and network design (Marín and Jaramillo 2009). Empirically, it does not work well for MTM due to big-M constraints in the subproblem. Therefore, we adopt the CB technique that removes dependence on big-M constraints. The CB approach has its roots in logic-based Benders decomposition (Hooker and Ottosson 2003). The vanilla cuts generated by the CB approach are relatively weak. To address this limitation, we make several new contributions. We develop multiple types of symmetry breaking constraints for the MTM problem that significantly reduce the search space. We also develop additional optimality cuts for the CB's subproblem to generate much tighter combinatorial cuts by enhancing information flow between the master and subproblems.

As a baseline approach, we also developed a CP formulation of the MTM problem (illustrated in the extended version of the paper). CP has been used in several scheduling applications such as for rail and air traffic management (Rodriguez 2007; Allignol et al. 2012). A key limitation of CP is that it is unable to provide optimality gap when stopped prematurely,

which was indeed the case for several large MTM instances where CP fails to terminate in 1 hr. In contrast, our CB-based approach augmented with strong optimality cuts and symmetry breaking constraints is able to *provably* generate near optimal solutions for several such problems. Furthermore, as the problem size increases, the CB approach provides better quality solutions than CP, showing the effectiveness of our approach.

Our final contribution is a realistic maritime traffic simulator using electronic navigation charts of Singapore Straits. We use historical 55-day AIS data for Singapore Straits that records the location of each ship within Straits and port waters at every 5 second interval over 55-days. We process this data and extract several real MTM problem instances. Using our scheduling technique, we show that we can significantly reduce traffic density (to about 60% of current traffic) at the expense of incurring marginal ($\approx$10%) average delay to ships. Thus, our work develops new computational approaches for the MTM problem and demonstrates their significant practical impact in the real world setting.

2 Problem Formulation

The MTM problem is a bi-objective optimization problem, where the first objective is minimizing congestion, and the second is minimizing delay. These two objectives are contradictory where minimizing one will increase the other. Taking the route of turning one objective into a constraint, the problem can be formulated in two ways: (1) minimize delay subject to maximum congestion constraints (modeled as resource capacities); (2) minimize congestion (resource consumption) subject to delay constraints. We use the first because it suits the application domain better, where safety (or congestion) comes first (encoded as hard constraint) before efficiency (objective). Empirically, we evaluate the tradeoff between the two to better help regulators in setting the appropriate safety level. Next, we detail the MIP formulation.

Let $\mathcal{R}$ be the set of *resources*. Examples of a resource are an area of traversable sea space, an anchorage area, or available pilots among others. Each resource $r \in \mathcal{R}$ has a capacity $K_r \in \mathbb{N}$. Let $\mathcal{N}$ be the set of n *vessels*. Each vessel $i \in \mathcal{N}$ requires m activities to be carried out sequentially without time lags between two consecutive activities (wlog we assume m is the same for each vessel). Each vessel i also has a release time $s_i \geq 0$ which is the earliest start time of its first activity. Let (i, j) denote the j-th activity of vessel i, and $\mathcal{A}$ the set of all activities ($|\mathcal{A}| = nm$). The amount of resource r required by activity (i, j) is denoted using a_r^{ij}. Furthermore, each activity (i, j) has a minimum and a maximum processing time, denoted by $T_{\min}^{ij}$ and $T_{\max}^{ij}$ respectively.

Let $\mathcal{E} = \{1, \ldots, nm\}$ be the set of events. Intuitively, each event corresponds to the start or the end of at least one activity (Koné et al. 2011). The first set of decision variables is $t = \{t_e | e \in \mathcal{E}\}$, where t_e denotes the time of event e. The times of events are ordered sequentially, that is, $t_1 \leq \ldots \leq t_{nm}$. The second set of decision variables is $z = \{z_e^{ij} | (i, j) \in \mathcal{A}, e \in \mathcal{E}\}$ where $z_e^{ij} = 1$ iff activity (i, j) is being carried out at t_e, and $z_e^{ij} = 0$ otherwise. When expressing a constraint, the dummy variable z_0^{ij} or z_{nm+1}^{ij} might

$$[P] \quad \min \sum_{i=1}^{n} S_{im} \tag{2}$$

subject to:

$$\forall (i,j) \in \mathcal{A}: \sum_{e=1}^{nm} z_e^{ij} \geq 1 \tag{3}$$

$$\forall (i,j) \in \mathcal{A}, e \in \mathcal{E} \setminus \{1\}: \sum_{e'=1}^{e-1} z_{e'}^{ij} \leq (e-1)\left(1 - (z_e^{ij} - z_{e-1}^{ij})\right) \tag{4}$$

$$\forall (i,j) \in \mathcal{A}, e \in \mathcal{E} \setminus \{1\}: \sum_{e'=e}^{nm} z_{e'}^{ij} \leq (nm-e+1)\left(1 + (z_e^{ij} - z_{e-1}^{ij})\right) \tag{5}$$

$$\forall (i,j) \in \mathcal{A} | j \neq m, e \in \mathcal{E}: \sum_{e'=1}^{e} z_{e'}^{ij+1} \leq (1 - z_e^{ij})e \tag{6}$$

$$\forall (i,j) \in \mathcal{A} | j \neq m, e \in \mathcal{E} \setminus \{1\}: z_e^{ij+1} \geq z_{e-1}^{ij} - z_e^{ij} \tag{7}$$

$$\forall e \in \mathcal{E}, r \in \mathcal{R}: \sum_{i=1}^{n}\sum_{j=1}^{m} a_r^{ij} z_e^{ij} \leq K_r \tag{8}$$

$$[\text{auxiliary constraints (1)}] \tag{9}$$

$$\forall i \in \mathcal{N}, e \in \mathcal{E}: \text{start}_e^{i1} = 1 \Rightarrow t_e \geq s_i \tag{10}$$

$$\forall i \in \mathcal{N}, e \in \mathcal{E}: \text{start}_e^{im} = 1 \Rightarrow S_{im} \geq t_e \tag{11}$$

$$\forall (i,j) \in \mathcal{A}, e,f \in \mathcal{E} | f > e: \text{span}_{ef}^{ij} = 1 \Rightarrow t_f \geq t_e + T_{\min}^{ij} \tag{12}$$

$$\forall (i,j) \in \mathcal{A} | j \neq m, e,f \in \mathcal{E} | f > e: \text{span}_{ef}^{ij} = 1 \Rightarrow t_f \leq t_e + T_{\max}^{ij} \tag{13}$$

$$\forall e \in \mathcal{E} \setminus \{nm\}: t_{e+1} \geq t_e \tag{14}$$

$$\forall (i,j) \in \mathcal{A}, e \in \mathcal{E}, i \in \mathcal{N}: z_e^{ij} \in \{0,1\}, t_e \in \mathbb{R}_{\geq 0}, S_{im} \in \mathbb{R}_{\geq 0} \tag{15}$$

Table 1: MIP for the MTM problem

$$[SP] \quad \min \sum_{i=1}^{n} S_{im} \tag{16}$$

subject to:
Constraints (10)-(14)

$$\sum_{i=1}^{n} S_{im} \leq \text{UB}(1-\epsilon) \tag{17}$$

$$\forall e \in \mathcal{N}, (i,j) \in \mathcal{A}: t_e \in \mathbb{R}_{\geq 0}, S_{im} \in \mathbb{R}_{\geq 0}$$

Table 2: Subproblem LP for combinatorial Benders

be used, in which case, its value is always 0 for any activity (i,j). We also define the following two sets of auxiliary binary variables. The first set is $\{\text{start}_e^{ij} \,|\, (i,j) \in \mathcal{A}, e \in \mathcal{E}\}$ where $\text{start}_e^{ij} = 1$ if (i,j) starts at e, otherwise zero. The second set is $\{\text{span}_{ef}^{ij} \,|\, (i,j) \in \mathcal{A}, e \in \mathcal{E}, f \in \mathcal{E}, e < f\}$ where $\text{span}_{ef}^{ij} = 1$ if (i,j) starts at e and ends at f, otherwise zero. We denote by $\mathcal{X}$ the set of all the auxiliary variables.

Auxiliary constraints:

$$\forall (i,j) \in \mathcal{A}, e \in \mathcal{E}, f \in \mathcal{E}, e < f,$$

$$\begin{array}{ll}
\text{start}_e^{ij} \leq z_e^{ij} & \text{span}_{ef}^{ij} \leq \text{start}_e^{ij} \\
\text{start}_e^{ij} \leq 1 - z_{e-1}^{ij} & \text{span}_{ef}^{ij} \leq z_{f-1}^{ij} \\
\text{start}_e^{ij} \geq z_e^{ij} - z_{e-1}^{ij} & \text{span}_{ef}^{ij} \leq 1 - z_f^{ij} \\
\text{start}_e^{ij}, \text{span}_{ef}^{ij} \in \{0,1\} & \text{span}_{ef}^{ij} \geq \text{start}_e^{ij} + z_{f-1}^{ij} - z_f^{ij} - 1
\end{array} \tag{1}$$

The above constraint set (1) links the auxiliary $\mathcal{X}$ variables with z. The variable start_e^{ij} is 1 iff z_{e-1}^{ij} is 0 and z_e^{ij} is 1. The span_{ef}^{ij} is 1 iff activity (i,j) starts at event e and ends at the event f. Let S_{ij} be the start time of activity (i,j). We have the following property of the objective function (proof provided in the extended version):

Proposition 1. *Minimizing the total delay over all the vessels is equivalent to minimizing the sum of start times of the last activity (S_{im}) of each vessel or $\sum_{i=1}^{n} S_{im}$.*

We next describe constraints in the MIP of table 1. Constraint (3) ensures that all activities are assigned to some events. Constraints (4) and (5) are the contiguity constraints which make sure that an activity is assigned to a continuous block of events. Constraint (6) ensures that a vessel's activities are scheduled sequentially, while constraint (7) ensures that there is no slack time between two consecutive activities. And the capacity constraint is given by (8). Conditional constraint (10) ensures that the first activity of each vessel starts after its release time. Constraint (11) defines the start times of the vessels' last activities. Constraints (12) and (13) enforce the minimum and maximum processing times of activities respectively. Finally, constraint (14) ensures that the event times are nondecreasing. Notice that the set of constraints can be divided into three categories: Constraints (3)-(8) together with the auxiliary constraints involve only binary variables, Constraints (10)-(13) are conditional constraints where the conditions consist of a single auxiliary variable, and the implications are linear inequalities in the continuous variables, and Constraint (14) involves only continuous variables.

One way to handle the conditional constraints (10)-(13) is to linearize them using the big-M method. Big-M formulations, however, are notoriously difficult to handle because they typically have very loose LP relaxations. One may use the classical Benders decomposition to remove the big-M in the subproblem, but the resulting cuts still contain it which make them very weak.

3 Combinatorial Benders Cuts

Similar to the classical Benders method, the problem [P] is decomposed into the master problem [MP] and the sub problem [SP]. In our case, [MP] is a constraint satisfaction problem of finding a consistent activity-to-event assignment z with respect to constraints (3)-(9) and the cuts generated so far. It is a constraint satisfaction problem because the objective function in [P] does not contain any binary variable (S_{im} is continuous). Given an activity-to-event assignment, specifically the values of the auxiliary variables $\mathcal{X}$, the subproblem [SP] (the LP in table 2) is to assign times to the events while minimizing the objective function. When the left hand side of a conditional constraint in (10)-(13) is satisfied then the right hand side inequality becomes active in the subproblem. Constraint (17) is added to [SP] to incorporate optimality requirements into the generated cuts, where UB is the objective value of the current best solution, and ϵ is a sufficiently small positive value less than 1.

A combinatorial cut is generated from an infeasible subproblem. For example, given an activity-to-event assignment z for which [SP] is infeasible, a trivial cut can be generated

by taking the variables on the left hand sides of all the active conditional constraints and appending them into a single constraint. Let $\mathcal{X}(z) \subseteq \mathcal{X}$ denotes the set of auxiliary variables that become true (are assigned the value 1) given z. The cut is given by $\sum_{x \in \mathcal{X}(z)} x \leq |\mathcal{X}(z)| - 1$. This trivial cut says that the auxiliary variables $\mathcal{X}(z)$ can not be all true at the same time. This cut is very weak because it cuts very few possible assignments from the solution space of [MP] (if any at all beside z). The idea of the combinatorial Benders cuts is to generate a much *shorter* cut using the concept of *irreducible inconsistent subsystems* (IISs). Given an infeasible system of linear inequalities, an IIS is a subset of the inequalities such that: (a) this subset is also infeasible and (b) removing any inequality from the subset makes it feasible. Let ϕ be an IIS of [SP], we denote by $\mathcal{X}_\phi(z) \subseteq \mathcal{X}(z)$ the set of auxiliary variables which activate some conditional constraints in ϕ. A combinatorial cut is of the form:

$$\sum_{x \in \mathcal{X}_\phi(z)} x \leq |\mathcal{X}_\phi(z)| - 1,$$

where in general, smaller ϕ generates stronger cut. Notice that when [P] is feasible, ϕ must contain some conditional constraints, otherwise, it implies that no assignment to the continuous variables will make [SP] feasible regardless of z, which is a contradiction. And since there can be many IISs given an infeasible system, we can generate multiple cuts from a single z.

The IIS generation method is based on the work of (Gleeson and Ryan 1990) which state that given an infeasible system of linear inequalities $By \geq d$, the indices of its IISs are exactly the supports of the vertices of the polyhedron $\{\pi | \pi B = 0, \pi d = 1, \pi \geq 0\}$, where π is the vector of dual variables, each corresponding to one inequality in $By \geq d$. To generate an IIS, one can therefore solve the following LP:

$$\min w\pi \quad \text{s.t.} \quad \pi \in \{\pi | \pi B = 0, \pi d = 1, \pi \geq 0\}, \quad (18)$$

where w is the weight parameter and multiple IISs may be generated by varying the value of w. Furthermore, w can be used to derive an IIS with certain properties. We follow the proposed method of (Parker and Ryan 1996) and (Bai and Rubin 2009) of weighting variables corresponding to previous found IISs more heavily to encourage the detection of IISs with small overlaps. Initially, we start with the value $w = 1$, in the hope of finding an IIS with a small cardinality. Note that finding an IIS with the smallest cardinality is an MIP and therefore is impractical for our purpose. Subsequently, we set $w_c = 1 + \Delta q_c$ where $\Delta > 0$ and q_c is the number of previously generated IISs containing the constraint associated with the dual π_c. Suitable values for Δ as well as the number of cuts to be generated are found using trial and error.

The full procedure for solving the decomposed problem is given in Algorithm 1. In the initialization phase (line 1), the following are performed using heuristic solutions: (1) determining the suitable number of events $|\mathcal{E}|$, (2) determining an initial upper bound, and (3) generating the initial set of cuts. Although in the worst case, nm number of events are needed to schedule nm activities (where the optimal solution is scheduling all vessels sequentially), in a typical instance, the number of events needed is much smaller. To determine

---

Algorithm 1: CB Decomposition for MTM

> **in** : $\Gamma = \langle \mathcal{N}, \mathcal{A}, \mathcal{R}, \{s_i\}, \{T_{\min}^{ij}\}, \{T_{\max}^{ij}\}, \{K_r\}, \{a_r^{ij}\} \rangle$
> **out** : z, t
> **param** : ϵ, Δ, k

1 $\mathcal{E}, \text{UB}, \mathcal{C} \leftarrow \text{INIT}(\Gamma)$
2 $z', \mathcal{X}(z') \leftarrow \text{SOLVEMASTER}([\text{MP}] \cup \mathcal{C})$
3 **while** $z' \wedge \mathcal{X}(z') \neq \{\}$ **do**
4   // [MP] $\cup \mathcal{C}$ is feasible
5 $t', \text{obj} \leftarrow \text{SOLVESUB}([\text{RSP}], \mathcal{X}(z'))$
6   **if** $t' = \{\}$ **then**
7   // generate k feasibility cuts
8 $\Phi \leftarrow \{\}$
9   **for** k *times* **do**
10 $\phi \leftarrow \text{SOLVEIIS}([\text{RSP}], \mathcal{X}(z'), \Phi, \Delta)$
11 $\Phi \leftarrow \Phi \cup \left\{ \sum_{x \in \mathcal{X}_\phi(z')} x \leq |\mathcal{X}_\phi(z')| - 1 \right\}$
12 $\mathcal{C} \leftarrow \mathcal{C} \cup \Phi$
13   **else**
14   **if** $\text{obj} \leq \text{UB}(1 - \epsilon)$ **then**
15 $\text{UB} \leftarrow \text{obj}$
16 $(z, t) \leftarrow (z', t')$
17   // generate k optimality cuts
18 $\Phi \leftarrow \{\}$
19   **for** k *times* **do**
20 $\phi \leftarrow \text{SOLVEIIS}([\text{SP}], \mathcal{X}(z'), \Phi, \Delta)$
21 $\Phi \leftarrow \Phi \cup \left\{ \sum_{x \in \mathcal{X}_\phi(z')} x \leq |\mathcal{X}_\phi(z')| - 1 \right\}$
22 $\mathcal{C} \leftarrow \mathcal{C} \cup \Phi$
23 $z', \mathcal{X}(z') \leftarrow \text{SOLVEMASTER}([\text{MP}] \cup \mathcal{C})$
24 // [MP] $\cup \mathcal{C}$ is infeasible
25 **return** z, t

---

the number of events to be used, we generate a set of heuristic solutions, and use the maximum number of events used among these solutions as $|\mathcal{E}|$. The best solution amongst these is also used to set the initial value of UB, and finally, from all the other solutions, optimality cuts are generated and added to [MP] before starting the iteration phase.

In each iteration, we start by solving [MP] together with the generated cuts so far (line 2 & 23). Given an activity-to-event assignment z, we generate cuts by solving the IIS problem defined by (18). The IIS problem is different depending on whether we are generating a feasibility cut or an optimality cut. We first solve the restricted subproblem [RSP] (line 5), which is [SP] with only constraints (10)-(14). If it is infeasible, which means that there is no feasible assignment of event times, we add feasibility cuts by generating the IISs of [RSP] (line 8-12). On the other hand, if [RSP] is feasible we check whether constraint (17) is violated. If it is, optimality cuts are generated by finding IISs of the full [SP] (line 18-22). If (17) is not violated, then a new current best solution is found, in which case, the value of UB is updated (line 15 & 16). This will make [SP] infeasible since constraint (17) will now become false, and optimality cuts are generated (line 18-22). This process continues until [MP] becomes infeasible, in which case, the current best found solution is within $\epsilon \times 100\%$ optimal.

In our implementation, rather than resolving [MP] repeat-

edly, we solve a single [MP] and incorporate the cut generations within its branch-and-bound framework using CPLEX callback functions. Whenever a feasible integer solution is found, the search is interrupted and a process is run to solve the corresponding IIS problems and generate cuts that are then added back to [MP] before resuming the search. The nodes that have been invalidated so far will remain invalid since the added cuts only make the problem more constrained by invalidating current node and potentially future nodes. This has the benefit of cutting down the running time significantly by not revisiting nodes that have already been processed in previous passes. (Bai and Rubin 2009) report savings between 3 to 6.5 times as much CPU time as compared to resolving the master problem per iteration.

3.1 Strengthening Optimality Cuts

To further strengthen the optimality cuts generated by the subproblem, we add the following constraints, which are specific to the MTM problem, to [SP]:

$$\forall i \in \mathcal{N}, j \in [1, m-1], e \in \mathcal{E}: \text{start}_e^{ij} = 1 \Rightarrow S_{ij} \geq t_e \quad (19)$$

$$\forall (i,j) \in \mathcal{A}: S_{ij} \geq s_i + \sum_{j'=1}^{j-1} T_{\min}^{ij'} \quad (20)$$

$$\forall (i,j), (i',j') \in \mathcal{A}; e, f \in \mathcal{E} | e \geq f:$$
$$\text{start}_e^{ij} = 1 \wedge \text{start}_f^{i'j'} = 1 \Rightarrow S_{ij} \geq S_{i'j'} \quad (21)$$

$$\forall (i,j) \in \mathcal{A}: S_{ij} \in \mathbb{R}_{\geq 0}$$

Constraint (19) defines the start time of activities. Notice that in the original formulation [P], only the start times of the last activities are defined. Here, we define the same for all the other activities. Constraint (20) defines the minimum start time of activities regardless of assignments to events. And constraint (21) is a conditional constraint between any two activities (i,j) and (i',j') that says that if (i,j) starts at a later event than (i',j') then its start time should be later as well. To illustrate how these constraints improve the optimality cuts, consider the activity-to-event assignments given in Figure 2(a). Suppose that this assignment is feasible, and activity (5,2) has an earliest start time of 10 and activity (2,2) has an earliest start time of 5. Given this, vessel 2 will incur a delay of at least 5. Suppose that this is the only delay and it violates the optimality constraint (17), then the following optimality cuts can be generated from the additional constraints:

$$\left\{ \text{start}_e^{5,2} + \text{start}_f^{2,2} \leq 1 | e, f \in \mathcal{E}, e < f \right\}.$$

Notice that these cuts can not be generated from the original constraints in [SP], since these auxiliary variables do not appear in any of the original conditional constraints.

3.2 Symmetry-Breaking Constraints

Similar to many combinatorial optimization problems, symmetry occurs in our problem. Specifically, we refer to the assignments of activities to events in the master problem [MP], where two assignments are symmetric if both return the same objective value when passed to the subproblem [PSP]. Symmetries are problematic because they increase the search space and algorithms like branch-and-bound/cut are

(a) base assignment   (b) shift operations

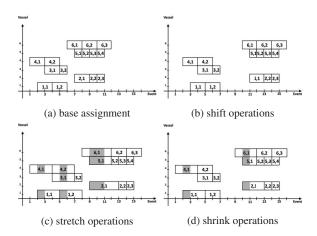(c) stretch operations   (d) shrink operations

Figure 2: Example of equivalent assignments

particularly susceptible because a lot of efforts will be spent on visiting symmetric states in the search tree. For recent surveys on the subject, see (Gent, Petrie, and Puget 2006; Margot 2010; Walsh 2012). Checking for and removing all symmetric solutions are a hard problem in itself and at times, might not be the best thing to do. In this section, we present some static inequalities that are added to [MP] to remove a large number of symmetric solutions. We start by defining a set of operations on assignments, under which the optimal value is preserved. We then present the inequalities that break the symmetry defined by these operations. And finally, we present a related constraint to strengthen the LP relaxation of [MP].

Given an assignment z, we define the following operations on z: *shift*, *stretch* and *shrink*. The shift operation takes a non-overlapping subassignment and shifts it left or right without creating new overlaps. The stretch operation takes the assignment to an event and duplicates it to the right (left) and shifts subsequent assignments to the right (left). The shrink operation, which is the inverse of stretch, removes a duplicate assignment from two consecutive events and "shrink" the rest of the assignments. Figure 2 gives some examples of these operations. Assignment 2(a) is used as the base. It has two non-overlapping subassignments, one involving vessels 1, 3 and 4, and another, 2, 5, and 6. Assignment 2(b) is obtained by shifting these subassignments around. Assignment 2(c) is obtained by taking 2(b) and duplicating some events. Event 11 is duplicated twice to the left, event 5 once to the right and event 3 once to the left. And finally, assignment 2(d) is obtained by removing duplicate events in 2(c). Note that these operations define equivalent classes, where two assignments are in the same class if one can be reached from the other with a series of these operations. We have the following property.

Proposition 2. *Let z and z' be two assignments of activities to events. If z and z' belong to the same equivalent class defined by the operations: shift, stretch and shrink, then the subproblem* [PSP] *returns the same objective value for both.*

Since assignments in the same class are symmetric to each other, we only need to consider one assignment from each

class during branch-and-bound. To do this, we add the following two global constraints to [MP]:

$$\forall e \in \mathcal{E} \setminus \{nm\}: \sum_{i=1}^{n}\sum_{j=1}^{m} z_{e+1}^{ij} \leq n \cdot \sum_{i=1}^{n}\sum_{j=1}^{m} z_{e}^{ij}, \qquad (22)$$

$$\forall e \in \{0,\ldots,nm\}: \sum_{i=1}^{n}\sum_{j=1}^{m} z_{e}^{ij} \oplus z_{e+1}^{ij} \geq \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} z_{e}^{ij}, \quad (23)$$

where $\oplus$ is the exclusive-or operation. Constraint (22) ensures a left-shifted assignment, where an event has zero assignment only if subsequent events have zero assignment as well. The same effect can be obtained by ordering the variables during search, which is a common technique in symmetry breaking. Constraint (23) ensures that no consecutive events have the same assignment except the zero assignment. Together, they make only the "compact" assignment from each class feasible. The compact assignment is one that is left-shifted and cannot be shrunk further. In the examples of Figure 2, only assignment 2(a) is feasible given these constraints.

The left term of constraint (23) has an interesting interpretation. For any event e, its value denotes the number of activities that start or end at $e + 1$. Given this, we can add the following constraint to [MP] to strengthen its LP relaxation:

$$\sum_{e=0}^{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} z_{e}^{ij} \oplus z_{e+1}^{ij} = 2nm, \qquad (24)$$

which ensures that the total number of activity-start and end in all events equals to twice the number of activities. It is a valid equality cut because it doesn't change the feasible region of [MP], but it reduces that of its LP relaxation.

Proposition 3. *Equality* (24) *is a valid and useful cut for the master problem* [MP].

4 Experimental Results

We perform experiments on both synthetic and real-world instances. Synthetic instances are used to compare the performance of different methods and formulations, while the real-world instances are used to measure the effectiveness and the tradeoff of using scheduling approach to mitigate congestions within Singapore waters.

For each synthetic instance, we generate a semi-random connected undirected graph where the edges of the graph represent activities. For each vessel, we then generate a random path of a fixed length through the graph. The length of the path represents the number of activities. With each edge in the path is associated a minimum and a maximum time required to traverse the edge, which represent the minimum and maximum processing time of the activity (which may be different for every vessel). To each vessel is also associated a release time. The base resources are the edge capacities which denote the number of vessels that can be at any edge at a given point of time. Additional resources are generated by grouping a set of edges and setting a collective capacity for the set (the maximum number of vessels that can be on these edges at any point of time).

Using these synthetic instances, we compare the performance of the combinatorial Benders approach (implemented

(a) Plot for the busiest day (03/13)

Date	Average Delay (%)		
(2014)	0.6C	0.5C	0.4C
03/02	5	14	39
03/03	7	20	37
03/04	6	14	35
03/05	4	17	35
03/06	5	19	36
03/07	6	14	32
03/08	7	19	39
03/09	5	16	38
03/10	6	19	39
03/11	4	16	37

(b) Results for the first 10 days

Figure 3: Capacity vs delay tradeoff
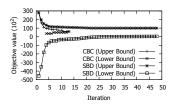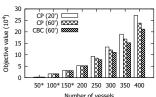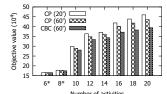
using CPLEX 12.7) against classical Benders decomposition and the baseline CP approach (implemented using CP Optimizer 12.7). As described in the previous section, classical Benders suffers from the existence of big-M, which makes the cuts very weak, and this is further confirmed in the experiments. Even for medium size instances, the classical Benders exhibits a degenerate behavior. Figure 4(a) shows the movements of the upper and lower bounds of both combinatorial Benders (CBC) and classical Benders (SBD) on a medium size instance. It shows that while CBC manages to achieve convergence within a few iterations, classical Benders fails to converge even after more than 100 iterations. For comparisons with the baseline CP, three sets of synthetic instances are generated by varying the number of vessels, the number of activities and the number of additional resources.
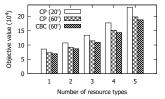
The settings for the three sets of instances are as follows. For all instances, the capacity of each edge is uniformly generated between 1-3, and the minimum time to traverse an edge is sampled uniformly from [5sec, 10sec]. The maximum is set to twice the minimum. The release time of each vessel is uniformly generated between 0-20 seconds, and each activity consumes 1 unit of resources. For each setting, 10 instances are generated and the average values are taken. Figures 4(b-c) show the comparison results for the three sets– x-axis shows varying parameters, y-axis is the objective (2) (lower is better). The CP solver is run for both 20 minutes and 1 hour on each instance, while CBC is run for 1 hour. The results show that on medium-size instances, CBC and the baseline CP perform equally well. However, while the baseline CP does not terminate within the time limit, CBC terminates with proven 10% optimality gap (for instances with variables marked by * on the x-axis). On larger instances, CBC achieves around 10-15% improvement over CP, with better quality with increasing number of vessels and number of activities (figures 4(b,c)).

For real-world instances, we use 55-day historical AIS data of vessels moving in Singapore waters from March 2nd to April 25th, 2014, consisting of over 4 millions records. An AIS record contains, among other things, a timestamp, vessel unique ID, lat-long position, speed over ground, direction and navigation status (e.g., at anchor, not under command, etc.) An AIS transponder on-board sends a record every 2-10 seconds while the vessel is underway, and every 3 minutes
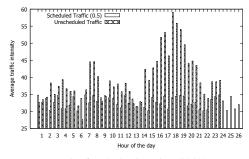
(a) Convergence rate of CBC vs SBD on a small instance

(b) CBC vs CP with varying number of vessels

(c) CBC vs CP with varying number of activities

(d) CBC vs CP with varying number of additional resources

Figure 4: Performance comparison on synthetic data

(a) Plot for the busiest day (03/13)

Date	Scheduled Traffic (0.5C)			Unscheduled Traffic		
(2014)	Min	Max	SD	Min	Max	SD
03/13 - Thu	30.09	34.84	1.57	27.6	55.83	6.53
03/03 - Mon	28.52	34.45	1.61	25.85	55.03	5.8
03/21 - Fri	27.8	34.29	1.78	26.24	53.95	6.15
04/16 - Wed	28.38	32.75	1.25	25.81	54.27	6.7
04/17 - Thu	27.61	30.06	1.54	23.77	52.25	5.82

(b) Results for the top 5 busiest days

Figure 5: Comparison of traffic intensities.

at anchor. In this work, we consider only tankers and cargos, which are the largest vessel types causing hotspots. From the records of each day, we perform the following to create scheduling instances. We first divide the day into 24 1-hour periods, and group the vessels according to these periods. A vessel belongs to a period if its first record on that day falls within that period. Then, from the vessel set in one period, we create an instance, where the activities of a vessel are inferred from its records. The recorded length of an activity becomes the minimum duration in the instance, and the maximum is set to twice the minimum (i.e., a vessel can go twice as slow). The earliest release time of a vessel is the timestamp of its first record on that day. Resource capacity and requirement are computed as follows. We identify the size (diameter) of the smallest tankers/cargos u, and that becomes 1 unit of resource. A vessel with the size of approximately $2u$, for example, will consume 2 units of space resources. A zone's capacity is set to the maximum recorded units, at any timepoint, over all the records. A rolling-horizon approach is then used to reschedule the vessels in a day, where we start by solving the first-period instance. The second-period instance is then solved by fixing the resulting schedule from the first instance and so on.

The results are shown in Figures 3 and 5. In Figure 3, we show the tradeoff between reducing capacities (and thus congestion) and the incurred delay. The capacity parameter determines the reduction in the zones' capacities. E.g., at 0.5C, capacities of all zones are reduced by half (up to a minimum which is required for any vessel to pass through the zones). Figure 3(b) shows the average percentage of delay over 24 periods for different capacity levels for the first 10 days (other 45 days show similar trends and are omitted). Figure 3(a) shows the plot for the busiest day (March 13th). An important observation here is that delays are minimal until 0.6C capacity, which implies that schedule adjustment has the potential to significantly reduce congestion with minimal resulting delays. In Figures 5, we show how rescheduling affects the traffic intensity in a day. Here the traffic intensity is measured as the number of tankers/cargos within the planning area at any point of time. Figure 5(b) shows the min, max and standard deviation (SD) of traffic intensity for the 5 busiest days. The scheduled traffic is with half capacity (0.5C). It shows that by rescheduling, congestion can be significantly reduced. Significantly lower standard deviations for the scheduled traffic shows that peak traffic intensity is flattened out. Figure 5(a) shows the plot of traffic intensities for the busiest day, comparing unscheduled with scheduled traffic. This figure further illustrates how intelligent rescheduling effectively flattens traffic intensity over the day and thereby reduces congestion.

5 Summary

In this paper, we address the problem of mitigating congestion in a busy water area while minimizing average delay. We formulate the maritime traffic management problem as a variant of RCPSP where activity durations, which translate into average vessel speeds, also become decision variables unlike the classical RCPSP. We explore both continuous-time event-based MIP formulation and discrete-time CP formulation. To solve the former, we apply a combinatorial Benders approach, with stronger cuts and symmetry-breaking constraints. Empirical results on synthetic instances show that combinatorial Benders performs significantly better than classical Benders, and is about 10-15% better than the CP model. On real-world instances, the results confirm that using intelligent scheduling can result in significant peak traffic reduction while incurring minimal delay. We also developed a realistic maritime simulator to analyze and visualize our approach and results.

Acknowledgments

References

Allignol, C.; Barnier, N.; Flener, P.; and Pearson, J. 2012. Constraint programming for air traffic management : a survey. *Knowledge Engineering Review* 27(3):pp 361–392.

Alvarez-Valdés, R., and Tamarit, J. M. 1993. The project scheduling polyhedron: Dimension, facets and lifting theorems. *European Journal of Operational Research* 67(12):204–220.

Bai, L., and Rubin, P. A. 2009. Combinatorial benders cuts for the minimum tollbooth problem. *Operations Research* 57(6):1510–1522.

Bartusch, M.; Möhring, R. H.; and Radermacher, F. J. 1988. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research* 16(1):199–240.

Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4(1):238–252.

Codato, G., and Fischetti, M. 2006. Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research* 54(4):756–766.

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

Gent, I. P.; Petrie, K. E.; and Puget, J.-F. 2006. Symmetry in constraint programming. In Rossi, F.; van Beek, P.; and Walsh, T., eds., *Handbook of Constraint Programming*. Elsevier. 329–376.

Gleeson, J., and Ryan, J. 1990. Identifying minimally infeasible subsystems of inequalities. *ORSA Journal on Computing* 2(1):61–63.

Hand, M. 2017. Malacca and S'pore Straits traffic hits new high in 2016, VLCCs fastest growing segment. http://www.seatrade-maritime.com/news/asia/malacca-and-s-pore-strait-traffic-hits-new-high-in-2016-vlccs-fastest-growing-segment.html.

Hooker, J. N., and Ottosson, G. 2003. Logic-based Benders decomposition. *Mathematical Programming* 96:33–60.

Kolisch, R., and Hartmann, S. 1999. Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In *Project Scheduling*. Springer US. 147–178.

Koné, O.; Artigues, C.; Lopez, P.; and Mongeau, M. 2011. Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research* 38(1):3–13.

Lim, V. 2017. 300 tonnes of oil spilled after Singapore-registered ship collides with vessel off Johor. http://www.channelnewsasia.com/news/singapore/300-tonnes-of-oil-spilled-after-singapore-registered-ship-collid-7537142.

Margot, F. 2010. Symmetry in integer linear programming. In *50 Years of Integer Programming 1959-2008*. Springer Berlin Heidelberg. 647–686.

Marín, Á., and Jaramillo, P. 2009. Urban rapid transit network design: Accelerated benders decomposition. *Annals of Operations Research* 169(1):35–53.

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

Mingozzi, A.; Maniezzo, V.; Ricciardelli, S.; and Bianco, L. 1998. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science* 44(5):714–729.

Papadakos, N. 2009. Integrated airline scheduling. *Computers & Operations Research* 36(1):176–195.

Parker, M., and Ryan, J. 1996. Finding the minimum weight IIS cover of an infeasible system of linear inequalities. *Annals of Mathematics and Artificial Intelligence* 17:107–126.

Qu, X.; Meng, Q.; and Suyi, L. 2011. Ship collision risk assessment for the Singapore Strait. *Accident Analysis & Prevention* 43(6):2030–2036.

Rodriguez, J. 2007. A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological* 41(2):231 – 245.

Romanski, J., and Hentenryck, P. V. 2016. Benders decomposition for large-scale prescriptive evacuations. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 3894–3900.

Schutt, A.; Feydy, T.; Stuckey, P. J.; and Wallace, M. G. 2013. Solving RCPSP/max by lazy clause generation. *Journal of Scheduling* 16(3):273–289.

Schutt, A.; Feydy, T.; and Stuckey, P. J. 2013. Explaining time-table-edge-finding propagation for the cumulative resource constraint. In *Proceedings of the 10th CPAIOR international conference*, 234–250.

Segar, M. 2015. Challenges of Navigating In Congested and Restricted Water. http://www.mpa.gov.sg/web/wcm/connect/www/968fafb8-7582-4091-9bcd-ec0a332f73a6/segar_challenges_of_navigating.pdf.

Tan, A. 2017. Big cleanup of Singapore's north-eastern coast after oil spill. http://www.straitstimes.com/singapore/environment/big-cleanup-of-n-e-coast-after-oil-spill.

Vilím, P. 2011. Timetable edge finding filtering algorithm for discrete cumulative resources. In *Proceedings of the 8th CPAIOR international conference*, 230–245.

Walsh, T. 2012. Symmetry breaking constraints: Recent results. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2192–2198.