CP Methods for Scheduling and Routing with Time-Dependent Task Costs

Elena Kelareva[1 2], Kevin Tierney[3], and Philip Kilby[1 2]

[1] Australian National University, Canberra, Australia
[2] NICTA, Canberra, Australia
elena.kelareva@nicta.com.au
[3] IT University of Copenhagen, Copenhagen, Denmark

Abstract. A particularly difficult class of scheduling and routing problems involves an objective that is a sum of time-varying action costs, which increases the size and complexity of the problem. Solve-and-improve approaches, which find an initial solution for a simplified model and improve it using a cost function, and Mixed Integer Programming (MIP) are often used for solving such problems. However, Constraint Programming (CP), particularly with Lazy Clause Generation (LCG), has been found to be faster than MIP for some scheduling problems with time-varying action costs. In this paper, we compare CP and LCG against a solve-and-improve approach for two recently introduced problems in maritime logistics with time-varying action costs: the Liner Shipping Fleet Repositioning Problem (LSFRP) and the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP). We present a novel CP model for the LSFRP, which is faster than all previous methods and outperforms a simplified automated planning model without time-varying costs. We show that a LCG solver is faster for solving the BPCTOP than a standard finite domain CP solver with a simplified model. We find that CP and LCG are effective methods for solving scheduling problems, and are worth investigating for other scheduling and routing problems that are currently being solved using MIP or solve-and-improve approaches.

1 Introduction

Scheduling problems typically aim to select times for a set of tasks so as to optimise some cost or value function, subject to problem-specific constraints. Traditional scheduling problems usually aim to minimise the makespan, or total time, of the resulting schedule. More complex objective functions, such as minimising the total weighted tardiness, may vary with time [30]. Routing problems have many similarities with scheduling – both may have resource constraints and setup time constraints, both have actions that need to be scheduled in time, and both may have complex time-dependent cost functions for actions.

In a number of important, real-world scheduling problems, such as the Liner Shipping Fleet Repositioning Problem (LSFRP) [32, 33] and the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP) [14, 15], the objective is a sum of time-varying costs or values for each task. Additional problems include net present value maximization in project scheduling [26]; satellite imaging scheduling [17, 38]; vehicle routing with soft time windows [29, 12]; ship routing and scheduling with soft time windows [8, 2]; and ship speed optimisation [10, 20].

Mixed Integer Programming (MIP) is a standard approach used to solve many scheduling and routing problems. Solve-and-improve approaches are also commonly used to solve scheduling and routing problems with complex constraints or complex objective functions, such as objective functions that are the sum of time-dependent task costs. Solve-and-improve approaches initially solve a simplified problem, then improve the solution using the objective function and constraints of the full problem.

For example, in the satellite image scheduling problem, each observation may only be performed for a specified period of time during the satellite's orbit, and the quality of the observation drops off to zero for times before and after the peak quality window [38]. Yao *et al* [39] and Wang *et al* [37] solved this problem by converting the image quality function to hard time windows when a "good enough" image could be obtained. Lin *et al* [17] found solutions within 2% of optimality by first converting the quality function to hard time windows to find feasible solutions, and then heuristically improving the solution quality by minor changes in the schedule.

Solve-and-improve approaches have also often been used in other applications, such as the vehicle routing problem with soft time windows (VRPSTW), which has time-varying penalties for early and late arrival outside each customer's preferred time window. Soft time windows allow better utilisation of vehicles, thus reducing transportation costs compared to hard time windows, while still servicing most customers within their preferred times. However, soft time windows result in a more complex objective function, making the problem significantly more difficult to solve [24]. Some VRPSTW approaches optimise first for the number of vehicles (routes), then for minimal travel time and distance, and finally improving solutions by minimising cost with time window penalties included [12]. Solve-and-improve approaches are a standard technique which have been discussed in a recent review of vehicle routing with time windows [7].

However, for some problems such as [14], Constraint Programming (CP) has been shown to be more effective than MIP. CP is also a very flexible method that can be used to model a wider variety of constraints than MIP, which is limited to linear constraints. A number of recent approaches have combined CP with other techniques such as vehicle routing [16], SAT [21] and MIP [1] in order to combine the flexibility of CP with fast algorithms for specific problems. CP approaches may be worth investigating for other problems that have traditionally been modelled with MIP.

One CP technique in particular which has been found to be effective on a number of scheduling problems is Lazy Clause Generation (LCG) [21] – a method for solving CP problems which allows the solver to learn where the previous search failed. LCG combines a finite domain CP solver with a SAT solver, by lazily adding clauses to the SAT solver as each finite domain propagator is executed. This approach benefits from efficient SAT solving techniques such as nogood learning and backjumping, while maintaining the flexible modelling of a CP solver and enabling efficient propagation of complex constraints [21].

A CP solver that uses LCG was found by Schutt *et al* [27] to be more efficient than traditional finite domain CP solvers for the Resource Constrained Project Scheduling Problem with Net Present Value – another scheduling problem with time-dependent action costs, where each activity has an associated positive or negative cash flow that is discounted over time. This problem has previously been solved by relaxing the problem to remove resource constraints, and using the resource-unconstrained solution as an upper bound in the search [35]; however, a CP solver with LCG was able to find better solutions than any previous state-of-the-art complete method [27].

LCG has also been found to be faster than finite domain solvers for a number of other scheduling problems, including the Concert Hall Scheduling problem [5], project scheduling with generalised precedence constraints [28] and a number of other scheduling problems [11]. LCG solvers can be used for any problem that is modelled as a constraint programming problem, which makes LCG a highly generalisable technique that is worth investigating for other scheduling and routing problems with time-varying task costs as an alternative to MIP or model simplification approaches.

The contribution of this paper is to compare the effectiveness of CP and LCG against traditional MIP and solve-and-improve techniques for two recent scheduling and routing problems with time-varying action costs in the field of maritime transportation. We present a novel CP model for the LSFRP and show that this model is faster than existing approaches including MIP and automated planning, even when a simplified automated planning model with no time-varying costs is used. We show that a CP solver with LCG is more effective at solving the BPCTOP with time-varying draft than a traditional Finite Domain CP solver (CP was found to be faster than MIP for the BPCTOP in an earlier paper [14]). We also show that the LCG solver scales better to large BPCTOP problems than the first step of a solve-and-improve approach that simplifies the time-varying cost function to find an initial solution.

2 Background

2.1 Bulk Port Cargo Throughput Optimisation with Time-Varying Draft

Many ports have safety restrictions on the draft (distance between waterline and keel) of ships sailing through the port, which vary with the height of the tide. Most maritime scheduling problems either ignore draft constraints entirely [9], or do not consider time variation in draft restrictions [25, 31]. This simplifies the problem and improves scalability, but may miss solutions which allow ships to sail with higher draft (and thus more cargo) close to high tide.

Introducing time-varying draft restrictions requires a problem to be modelled with a very fine time resolution, as the draft can change every five minutes. This greatly increases the size of the problem, so time-varying draft restrictions have thus far only been applied to the problem of optimising cargo throughput at a single bulk export port – the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP). The objective function for the BPCTOP with time-varying draft restrictions is the sum of maximum drafts for each ship at their scheduled sailing times. The shape of the objective function is similar to satellite image scheduling, as the maximum draft for each ship peaks around high tide, and drops off before and after.

The BPCTOP also includes resource constraints on the availability of tugs, berths or shipping channels, and sequence-dependent setup times between successive ships.

Ports may have safety restrictions on the minimum separation times between ships, as ships sailing too close together in a narrow channel may pose a safety risk. Minimum separation times between ships may also depend on which berths the ships sail from.

Kelareva *et al* [14] compared CP and MIP approaches for the BPCTOP, and found that CP with a good choice of search strategy was able to find optimal solutions faster than MIP, and was also able to solve problems with more ships. These approaches also produced significantly better solutions compared to human schedulers [15]. Each extra centimetre of draft can increase profit for the shipper by up to $10,000 [14], so there is a high incentive to find optimal solutions, and non-optimal approaches for this problem have not yet been investigated.

The CP model was able to solve problems with up to 6 ships for even the most tightly constrained problems, where all ships are very large and can continue loading extra cargo right up to the peak of the high tide, and with 10 or more ships for less tightly constrained problems. These are realistic problem sizes – Port Hedland, Australia's largest bulk export port recently set a record of 5 draft-constrained ships sailing on the same high tide [22].

2.2 Liner Shipping Fleet Repositioning

Another recently introduced scheduling and routing problem with time-varying action costs in maritime transportation is the Liner Shipping Fleet Repositioning Problem (LSFRP) [32]. In liner shipping, vessels are assigned to services that operate like a bus timetable, and vessels regularly need to be repositioned between services in response to changes in the world economy. A vessel begins repositioning when it *phases out* from its original service, and ends repositioning when it *phases in* to its new service.

The total cost of the repositioning depends on fuel usage, and a fixed hourly *hotel cost* paid for the time between the phase-out and phase-in. The cost function for a repositioning action is therefore monotonically increasing with the duration of sailing, as the fuel cost is much lower for vessels sailing at low speed. There may also be opportunities for a vessel to carry empty containers (*sail-equipment* (SE)), or replace another vessel in a regular service (*sail-on-service* (SOS)), which significantly reduces the cost of the repositioning. The LSFRP is not a pure scheduling problem, as there is a choice of actions and action durations.

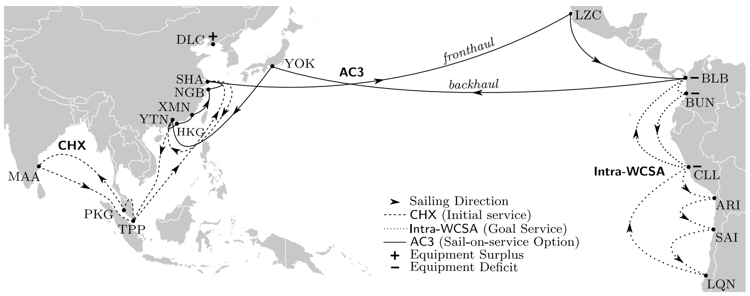

Fig. 1: A subset of a real-world repositioning scenario, from [32].

Figure 1 shows a subset of a repositioning scenario in which the new Intra-WCSA service requires three vessels that must be repositioned from their services in South-East Asia. One of the vessels was originally on the CHX service, and the other two were on other services not shown in this figure. The cost of repositioning in this scenario can be reduced by carrying equipment from China to South America (e.g. DLC to BLB), or using the AC3 service as an SOS opportunity.

There has been little prior research on the LSFRP. In fact, the LSFRP is not even mentioned in the two most comprehensive surveys of maritime research [4, 3]. The LS-FRP was first solved by Tierney *et. al.* [32], who compared three methods for solving the LSFRP. First, the LSFRP was modelled using PDDL 2.1, a modeling language for automated planning problems (see [13]), and solved using the POPF planner [6]. Although POPF found solutions, it could not solve the LSFRP to optimality. This method was therefore compared against a MIP model, and against a Linear Temporal Optimisation Planning (LTOP) approach, which uses temporal planning to build optimisation models, and solves these using an optimisation version of partial-order planning based on branch-and-bound.

Both LTOP and MIP were able to find optimal solutions for problem sizes with up to three ships. These are realistic problem sizes similar to those used by shipping lines. As with the BPCTOP, there is a high motivation to find the optimal solutions, even if solving the problem to optimality is more difficult, since the cost of repositioning a single ship can be hundreds of thousands of dollars [32].

3 CP vs MIP

3.1 CP Model for Liner Shipping Fleet Repositioning

Tierney, *et al* [32] presented MIP and automated planning models for the LSFRP. In this section, we present a novel CP model for the LSFRP, and compare its solution times against the earlier MIP and automated planning models.

Let O_v be the set of possible phase-out actions for the vessel v, and let P be the be the set of possible phase-in ports for the new service. The decision variable $\rho \in P$ is the phase-in port for all vessels. The decision variables $w_v \in \{1, \ldots, W\}$ represent the phase-in week for each vessel v, where W is the number of weeks considered in the problem. For each vessel v we also define a decision variable $q_v \in O_v$ specifying the phase-out action (port and time) used for that vessel.

For each vessel v and phase-out action o, the function $t(v, o)$ specifies the phase-out time for that action. Similarly, $t(p, w)$ specifies the phase-in time for a vessel phasing in at port p in week w. The function $C(v, o, p, w)$ specifies the cost for vessel v using the phase-out action o, and phasing in at port p in week w, with -1 as a flag that indicates vessel v cannot phase in at port p in week w if it phased out using action q (for example, if action q starts too late for vessel v to reach port p in time). The dependent variable c_v specifies the cost for vessel v when the vessel sails directly from the phase-out port to the phase-in port. For each vessel v, $C_H(v)$ specifies its hourly hotel cost, and h_v is the duration of the hotel cost time period (from the phase-out to the phase-in).

We split each SOS opportunity into several *SOS actions*, where each SOS action represents starting the SOS at a different port on the SOS service. SOS opportunities save money by allowing vessels to sail for free between two ports, however a cost for transshipping cargo at each side of the SOS is incurred. Let S be the set of available SOS actions and S' be the set of SOS opportunities. The decision variable $s_v \in S$ specifies the SOS action used for each vessel v, with 0 being a flag indicating that vessel v does not use an SOS action. For each SOS action $s \in S$, the function $y : S \to S'$ specifies which SOS opportunity each SOS action belongs to, with $y(s) = 0$ being a flag that specifies that the vessel is not using any SOS opportunity.

In order to use an SOS opportunity, a vessel must sail to the starting port of the SOS opportunity before a deadline, and after using the SOS, it sails from the end port at a pre-determined time to the phase-in port. The function $C^{to}(v, s, o)$ specifies the cost of vessel v using SOS action s, phasing out at phase-out o going to the SOS action, and $C^{from}(v, s, p, w)$ is the cost of vessel v to sail from SOS action s to phase in port p in week w, with -1 as a flag that indicates that this combination of vessel, SOS action, phase-in port and week is infeasible. The dependent variables σ_v^{to} and σ_v^{from} specify the SOS costs for vessel v for sailing to and from the SOS, respectively. The function $A(v, s)$ specifies the cost savings of vessel v using SOS action s, and the dependent variable σ_v^{dur} specifies the SOS cost savings for vessel v on the SOS.

Let Q be the set of *sail-equipment* (SE) opportunities, which are pairs of ports in which one port has an excess of a type of equipment, e.g. empty containers, and the other port has a deficit. Since we do not include a detailed view of cargo flows in this version of the LSFRP, SE opportunities save money by allowing vessels to sail for free between two ports as long as the vessel sails at its slowest speed. The cost then increases linearly as the vessel sails faster. Let the decision variable $e_v \in E$ be the SE opportunity undertaken by vessel v, with $e_v = 0$ indicating that no SE opportunity is used. Let the decision variables d_v^{to}, d_v^{dur} and d_v^{from} be the duration of vessel v sailing to, during, and from an SE opportunity.

The functions $C^{to}(v, e, o)$, $C^{dur}(v, e)$ and $C^{from}(v, e, p, w)$ specify the fixed costs of sailing to, utilizing, and then sailing from SE opportunity e, where v is the vessel, o is the phase-out port/time, p is the phase-in port and w is the phase-in week. Together with the constant α_v, which is the variable sailing cost per hour of vessel v, the hourly cost of sailing can be computed. This is necessary since SE opportunities are not fixed in time and, thus, must be scheduled. Let the dependent variables λ_v^{to}, λ_v^{dur} and λ_v^{from} be the fixed costs sailing to, on and from an SE opportunity. Additionally, let $\Delta_{min}^{to}(v, e, o)$, $\Delta_{min}^{dur}(v, e)$ and $\Delta_{min}^{from}(v, e, p, w)$ be the minimum sailing time of v before, during and after the SE opportunity and $\Delta_{max}^{to}(v, e, o)$, $\Delta_{max}^{dur}(v, e)$ and $\Delta_{max}^{from}(v, e, p, w)$ be the maximum sailing time of v before, during and after the SE opportunity.

In this version of the LSFRP, the chaining of SOS and SE opportunities is not allowed, meaning each vessel has the choice of either sailing directly from the phase-out to the phase-in, undertaking an SOS, or performing an SE. The decision variable $r_v \in \{\texttt{SOS}, \texttt{SE}, \texttt{SAIL}\}$ specifies the type of repositioning for each vessel v, where v utilizes an SOS opportunity, SE opportunity, or sails directly from the phase-out to the phase-in, respectively. The CP model is formulated as follows:

$$\begin{aligned}
\min \sum_{v \in V} \big(& C_H(v)\,(t(\rho, w_v) - t(v, q_v)) + c_v + \sigma_v^{from} + \sigma_v^{dur} + \sigma_v^{to} \\
& + \lambda_v^{to} + \lambda_v^{dur} + \lambda_v^{from} + \alpha_v (d_v^{to} + d_v^{dur} + d_v^{from}) \big) && (1) \\
\text{s.t.} \quad & \texttt{alldifferent}(w_v), v \in V && (2)
\end{aligned}$$

$$\max_{v \in V} w_v - \min_{v \in V} w_v = |V| - 1 \tag{3}$$

$$\texttt{alldifferent\_except\_0}(s_v), v \in V \tag{4}$$

$$\texttt{alldifferent\_except\_0}(y(s_v)), v \in V \tag{5}$$

$$r_v = \texttt{SAIL} \rightarrow c_v = C(v, q_v, \rho, w_v), \ \ \forall v \in V \tag{6}$$

$$\begin{aligned} r_v = \texttt{SAIL} \rightarrow \big(& s_v = 0 \wedge y(s_v) = 0 \wedge \sigma_v^{dur} = 0 \wedge \sigma_v^{from} = 0 \wedge \sigma_v^{to} = 0 \\ & \wedge e_v = 0 \wedge \lambda_v^{to} = 0 \wedge \lambda_v^{dur} = 0 \wedge \lambda_v^{from} = 0\big), \ \ \forall v \in V \end{aligned} \tag{7}$$

$$\begin{aligned} r_v = \texttt{SOS} \rightarrow s_v > 0 \wedge y(s_v) > 0 \wedge \sigma_v^{dur} = -A(v, s_v) \\ \wedge \sigma_v^{from} = C^{from}(v, s_v, \rho, w_v) \wedge \sigma_v^{to} = C^{to}(v, s_v, q_v), \ \ \forall v \in V \end{aligned} \tag{8}$$

$$r_v = \texttt{SOS} \rightarrow c_v = 0 \wedge e_v = 0 \wedge \lambda_v^{to} = 0 \wedge \lambda_v^{dur} = 0 \wedge \lambda_v^{from} = 0, \ \ \forall v \in V \tag{9}$$

$$s_v > 0 \vee y(s_v) > 0 \rightarrow r_v = \texttt{SOS}, \ \ \forall v \in V \tag{10}$$

$$\texttt{alldifferent\_except\_0}(e_v), \ \ \forall v \in V \tag{11}$$

$$\begin{aligned} r_v = \texttt{SE} \rightarrow e_v > 0 \wedge \lambda_v^{to} = C^{to}(v, e_v, q_v) \wedge \lambda_v^{dur} = C^{dur}(v, e_v) \\ \wedge \lambda_v^{from} = C^{from}(v, e_v, \rho, w_v), \forall v \in V \end{aligned} \tag{12}$$

$$\begin{aligned} r_v = \texttt{SE} \rightarrow s_v = 0 \wedge y(s_v) = 0 \wedge \sigma_v^{dur} = 0 \wedge c_v = 0 \\ \wedge \sigma_v^{from} = 0 \wedge \sigma_v^{to} = 0, \forall v \in V \end{aligned} \tag{13}$$

$$e_v > 0 \rightarrow r_v = \texttt{SE} \tag{14}$$

$$\Delta_{min}^{to}(v, e_v, q_v) \le d_v^{to} \le \Delta_{max}^{to}(v, e_v, q_v), \ \ \forall v \in V \tag{15}$$

$$\Delta_{min}^{dur}(v, e_v) \le d_v^{dur} \le \Delta_{max}^{dur}(v, e_v), \ \ \forall v \in V \tag{16}$$

$$\Delta_{min}^{from}(v, e_v, \rho, w_v) \le d_v^{from} \le \Delta_{max}^{from}(v, e_v, \rho, w_v), \ \ \forall v \in V \tag{17}$$

$$\sigma_v^{to}, \sigma_v^{from}, c_v \ge 0, \ \ \forall v \in V \tag{18}$$

The objective function (1) minimises the sum of the hotel costs and repositioning action costs minus the cost savings for SOS actions for the set of vessels. Constraints (2) and (3) specify that the vessels must all phase in to the new service on different, successive weeks. Constraints (4) and (5) specify that all vessels using SOS actions must use different actions and action types. `alldifferent_except_0` is a global constraint that requires all elements of an array to be different, except those that have the value 0.

Constraints (6) and (7) set the costs for a vessel if it uses a `SAIL` repositioning, and ensures that the SOS/SE actions and costs are set to 0, as they are not being used. Constraints (8) and (9) specify that if vessel v uses an SOS (`SOS`) repositioning action s_v, then its repositioning cost is equal to the costs for sailing to and from that SOS action based on the phase-out action, phase-in port and week, minus the cost savings $A(v, s_v)$ for that SOS action. In addition, the normal repositioning cost c_v and the sail equipment action for that vessel are set to 0. We also add redundant constraints (10) to reinforce that the repositioning type be set correctly when an SOS is chosen.

In constraints (11) we ensure that no two vessels choose the same SE action (unless they choose no SE action), and constraints (12) and (13) bind the costs of the sail equipment action to the dependent variables if an SE is chosen, as well as set the costs of a direct sailing and SOS opportunities for each vessel to 0. The redundant constraints (14) ensure that the repositioning type of vessel v is correctly set if an SE action is chosen. The minimum and maximum durations of the parts of the SE (sailing to the SE from the phase-out, the SE itself, and sailing from the SE to the phase-in) are set in constraints (15), (16) and (17). Constraint (18) requires that all SOS actions and phase-out/phase-in combinations must be valid for each vessel (i.e. transitions with -1 costs must not be used).

3.2 LSFRP CP Results

To compare our CP model for the LSFRP against an earlier MIP model and against the LTOP planner [32], we use 11 problem instances based on a real-world scenario provided by an industrial collaborator. The problem instances contain up to 3 vessels, with varying SOS and sail-with-equipment opportunities that may be used to reduce repositioning costs. Note that this version of the LSFRP lacks the cargo flows present in [33], but is nonetheless a real-world relevant problem.

The LSFRP CP model was formulated in the MiniZinc 1.6 modelling language [19, 18], and solved using the G12 finite domain solver [36]. We compare the CP against a MIP model and the LTOP planner [32], both using CPLEX 12.3. All problems were solved to optimality. Note that in our CP model for MiniZinc we had to add constraints on the maximum duration of SE actions, as well as a constraint on the maximum sum of the objective, in order to prevent integer overflows. These constraints do not cut off any valid solutions from the search tree. Since MiniZinc does not support floating point objective values, the MiniZinc model is a close approximation of the true objective.

We also used several search annotations within MiniZinc to help guide the solver to a solution. The first is to branch on the type of repositioning, r_v, before other variables, attempting at first to find a SOS option for each vessel, then searching through SE options, and finally SAIL options. This search order was the most efficient for the most complex models that include both SOS and SE opportunities, since SE constraints are more complex than SOS constraints, so searching SE options first is more time consuming for models that contain both SOS and SE opportunities.

For instances with SE opportunities, we also add a search annotation to branch on the SE opportunity, e_v, using the "indomain_split" functionality of MiniZinc, which excludes the upper half of a variable's domain. Both annotations use a first failure strategy, meaning the variable the solver branches on is the one with the smallest domain.

Table 1 compares the run times of the CP model against the MIP model and LTOP, all of which solve to optimality[4]. We ran the CP model with search annotations using a search order of SAIL/SOS/SE (CP–A), with search annotations and the SOS/SE/SAIL ordering (CP–AO), as well as with only the redundant constraints and using the solver's default search (CP–R), and using the SOS/SE/SAIL ordering with redundant constraints (CP–AOR). CP–AOR is faster than MIP on all instances, often by an order of magnitude. The main challenge for the CP model are instances with SE, same as for LTOP and the MIP. In fact, the CP model times out on two instances for CP and CP–R, as well as one for CP-A, whereas the MIP and LTOP solve all instances. Of particular note is that the CP model is able to solve AC3_3_3, which is the instance that most closely models the actual scenario faced by our collaborator, in only 10 seconds. Such a quick solution

[4] Experiments used AMD Opteron 2425 HE processors with a maximum of 4GB each.

Problem	LTOP	MIP	CP	CP–A	CP–AO	CP–R	CP–AOR
AC3_1_0	1.1	0.4	0.1	0.2	0.1	0.2	0.1
AC3_2_0	51.0	9.3	0.3	0.3	0.3	0.3	0.3
AC3_3_0	188.3	23.0	0.6	0.6	0.6	0.6	0.6
AC3_1_1e	3.9	3.8	0.7	0.7	0.7	0.8	0.4
AC3_2_2ce	15.2	27.7	-	83.9	25.5	-	11.3
AC3_3_2c	203.2	250.5	6.0	3.1	3.1	6.2	3.0
AC3_3_2e	217.1	228.8	1731.0	15.8	16.2	1742.8	13.2
AC3_3_2ce1	218.2	312.2	32.6	23.0	16.7	31.5	13.9
AC3_3_2ce2	192.4	252.6	64.4	25.4	23.3	63.7	17.1
AC3_3_2ce3	516.9	706.5	-	-	695.4	-	470.3
AC3_3_3	80.0	148.3	18.1	11.0	11.0	18.6	11.2

Table 1: Computation times in seconds to optimality for the CP model with and with annotations (A), repositioning type order (O), and redundant constraints (R) vs LTOP and the MIP.

time allows for interaction and feedback with a repositioning coordinator within a decision support system. These results show that it is critical to choose a search strategy that quickly finds good quality solutions, as well as constraints that propagate well to cut out infeasible areas of the search. Note that no single improvement alone is enough to give the CP model better performance than both LTOP and the MIP.

Our CP model comes with two limitations. The first limitation is the model's flexibility. A natural extension to this model would be to allow for the chaining of SOS and SE opportunities, which is easy to do in both the LTOP and MIP models, due to automated planning's focus on actions, and our MIP model's focus on flows. However, the CP model is structured around exploiting this piece of the problem. Other natural changes, such as allowing vessels to undergo repairs, would also be difficult to implement. The second limitation is that many of the components of the CP model involve pre-computations that multiply the number of phase-out actions with the number of phase-in ports and weeks. Although the model works well on our real world instance, these pre-computations pose an issue for scaling to larger liner shipping services.

Constraint Programming (CP) is a very flexible method that can model many different types of scheduling and routing problems with complex side constraints. Our CP model for the Liner Shipping Fleet Repositioning Problem (LSFRP) was able to solve all of the problem instances faster than Mixed Integer Programming (MIP) or automated planning, and we also found that CP was much faster than MIP for the Bulk Port Cargo Throughput Optimisation Problem [14]. However, while CP can be very efficient at solving scheduling and routing problems, the solution time is highly dependent on having a good choice of model and search strategy, so applying CP to new applications requires a good understanding of both the application and CP modelling techniques.

4 Lazy Clause Generation

Kelareva *et al* [14] presented CP and MIP models for the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP), and found that CP with a good choice of search strategy was much faster than MIP. However, the CP model solution time was highly dependent on the choice of modelling approach and search strategy used – a number of different modelling approaches and search strategies were investigated in [14, 15]. In this paper, we compare a LCG solver against a traditional finite domain CP solver for the BPCTOP model from [14], with all scalability improvements identified in [14, 15]

4.1 CP Model for Bulk Port Optimisation

Let V be the set of vessels to be scheduled. Let $[1, T_{max}]$ be the range of discretised time indices. Each vessel v has an earliest departure time $E(v)$, a maximum allowable draft $D(v,t)$ at each time t, and a tonnage per centimetre of draft $C(v)$. $ST(v_i, v_j)$ specifies the minimum separation time between the sailing times of every ordered pair of ships $v_i,\ v_j \in V$. Let B be the set of pairs of incoming and outgoing ships $B_i(b)$ and $B_o(b)$ indexed by b that use the same berth. For every such pair of ships, $d(b)$ is the minimum delay between their sailing times. The binary decision variable $s(v)$ specifies whether ship v is included in the schedule. Let $T(v) \in [1, T_{max}]$ be the decision variable specifying the scheduled sailing time for vessel v. The binary variable $sb(v_i, v_j)$ – SailsBefore(v_i, v_j) – is true iff vessel v_i sails earlier than vessel v_j, defined by Eq. (19).

$$sb(v_i, v_j) = 1 \leftrightarrow (T(v_i) < T(v_j) \wedge sb(v_j, v_i) = 0),\ \ \forall\, v_i,\ v_j \in V; v_i \neq v_j \qquad (19)$$

Let U_{max} be the number of tugs available at the port. Let I and O be the sets of incoming and outgoing ships. $G(v)$ is the number of groups of tugs required for ship v, and $H(v, g)$ is the number of tugs in group g of vessel v. G_{max} is the maximum number of groups of tugs required for any ship. $r(v, g)$ is the "turnaround time" – the time taken for tugs in group g of ship v to become available for another ship in the same direction (incoming vs outgoing). $X(v_i, v_j)$ is the extra time required for tugs from incoming vessel v_i to become available for outgoing vessel v_o. $U(v, t, g)$ is a dependent variable that specifies the number of tugs busy in tug group g of vessel v at time t, assuming the next ship for these tugs is in the same direction. $x(v, t)$ defines the number of extra tugs that are busy at time t for an outgoing vessel v, due to still being in transit from the destination of an earlier incoming ship. Finally, $L(v, t)$ specifies that the extra tug delay time for incoming vessel v overlaps with the sailing time of an outbound vessel at time t. The model is formulated as follows:

$$\text{maximise} \sum_{v \in V} s(v)\ .\ C(v)\ .\ D(v, T(v)) \qquad (20)$$

$$\text{s.t.} \qquad s(v) = 1 \Rightarrow T(v) \geq E(v),\ \ \forall\, v \in V \qquad (21)$$

$$s(B_i(b)) = 1 \Rightarrow s(B_o(b)) = 1 \wedge T(B_o(b)) \leq T(B_i(b)) - d(b),\ \ \forall\, b \in B \qquad (22)$$

$$sb(v_i, v_i) = 0,\ \ \forall\, v_i \in V; v_i \neq v_j \qquad (23)$$

$$s(v_i) = 1\ \wedge s(v_j) = 1 \Rightarrow \qquad (24)$$
$$(sb(v_i, v_j) = 1 \Rightarrow T(v_j) - T(v_i) \geq ST(v_i, v_j)),\ \ \forall\, v_i,\ v_j \in V$$

$$s(v) = 1 \wedge t \geq T(v) \wedge t < T(v) + r(v, g) \Rightarrow U(v, t, g) = H(v, g), \qquad (25)$$
$$\forall\, v \in V,\ t \in [1, T_{max}],\ g \in [1, G_{max}]$$

$$s(v) = 0 \vee t < T(v) \vee t \geq T(v) + r(v, g) \Rightarrow U(v, t, g) = 0, \qquad (26)$$
$$\forall\, v \in V,\ t \in [1, T_{max}],\ g \in [1, G_{max}]$$

$$x(v_o, t) = 0,\ \ \forall\, v_o \in O,\ t \in [1, T_{max}] \qquad (27)$$

$$L(v_i,t) \iff \exists\, v_o \in O \;\; s.t. \tag{28}$$
$$t = T(v_o) \wedge T(v_i) \le T(v_o) \wedge T(v_i) + \max_{g\in[1,G(v_i)]} r(v_i,g) + X(v_i,v_o) > T(v_o),$$
$$\forall\, v_i \in I,\; t \in [1, T_{max}]$$
$$x(v_i,t) = \text{bool2int}(L(v_i,t)). \sum_{g\in[1,G(v_i)]} H(v_i,g),\;\; \forall\, v_i \in I,\; t \in [1,T_{max}] \tag{29}$$
$$\sum_{v\in I}\sum_{g\in G(v)} U(v,t,g) \le U_{max},\;\; \forall\, t \in [1,T_{max}] \tag{30}$$
$$\sum_{v_o\in O}\sum_{g\in G(v_o)} U(v_o,t,g) + \sum_{v_i\in I} x(v_i,t) \le U_{max},\;\; \forall\, t \in [1,T_{max}] \tag{31}$$

The objective function (20) maximises total cargo throughput for the set of ships. Constraint (21) specifies the earliest departure time for each vessel. Equation (22) ensures that the berths for any incoming ships are empty before the ship arrives. Equation (23) specifies that no ship sails before itself. Equation (24) ensures that there is sufficient separation time between successive ships to meet port safety requirements. Equations (25) to (31) specify that the total number of tugs in use at any time must be no greater than the number of tugs available at the port, by splitting the tug constraints into several scenarios that can be considered independently as discussed in [14]. For a more detailed discussion of this model, including scalability issues, see [14, 15].

4.2 Experimental Results for Lazy Clause Generation

We used a CP solver with Lazy Clause Generation (LCG) to solve the above CP model for the BPCTOP. The model was formulated in the MiniZinc modelling language [19], and solved using the CPX solver included in G12 2.0 [36, 11], which uses Lazy Clause Generation. The runtimes were then compared against the G12 2.0 finite domain CP solver, using backtracking search with the fastest variable selection and domain reduction strategies as discussed in [14, 15]. All BPCTOP experiments used an Intel i7-930 quad-core 2.80 GHz processor and 12.0 GB RAM, with a 30-minute cutoff time.

Both solvers were used to solve problems presented in [14], with 4-13 ships sailing on a single tide, based on a fictional but realistic port, similar to the SHIP_SCHEDULING data set used for the 2011 MiniZinc challenge [34]. Table 2 presents calculation times in seconds for CPX and the G12 FD solver, for each problem size that could be solved to optimality within the 30-minute cutoff time, for the problems in the most tightly constrained (and thus most difficult) ONEWAY_NARROW (ON) problem set from [14]. A dash indicates that the optimal solution was not found within the cutoff time. Table 3 shows the largest problem solved for all eight problem types from [14], with runtimes in brackets. Problem types include all combinations of MIXED problems with inbound and outbound ships vs. all ships sailing ONEWAY, NARROW vs WIDE safe sailing windows for each ship, with and without tugs.

Table 2 shows CPX is faster to solve ON problems with tug constraints, and scales better than the FD solver for large problem sizes. However, CPX is slower to solve ON problems without tugs. This may indicate that CPX finds effective nogoods (areas of the search space with no good solutions) for tug constraints, enabling CPX to avoid searching large areas of the search space for the problem with tug constraints. The FD solver, on the other hand, cannot eliminate those areas of the search space, leading to excessive backtracking resulting from the highly oversubscribed tug problem.

	No Tugs		With Tug Constraints	
NShips	G12_FD	G12_CPX	G12_FD	G12_CPX
4	0.34	**0.23**	**0.23**	0.33
5	**0.23**	**0.22**	**0.33**	**0.33**
6	0.44	**0.22**	**0.44**	0.67
7	**0.34**	**0.33**	**2.95**	3.60
8	**0.45**	0.87	47.0	**24.4**
9	**4.70**	23.4	184	**65.7**
10	**99.9**	482	>1800	**817**
11	**609**	>1800	–	>1800
12	>1800	–	–	–

Table 2: Runtime (s) for CPX solver with LCG vs. G12 finite domain solver.

Problem	FD	CPX
MIXED_WIDE (MW)	11 (326)	**16 (1040)**
ONEWAY_WIDE (OW)	11 (1480)	**11 (273)**
MIXED_NARROW (MN)	11 (370)	**16 (1100)**
ONEWAY_NARROW (ON)	**11 (609)**	10 (482)
MIXED_WIDE_TUGS (MWT)	10 (11.5)	**13 (1290)**
ONEWAY_WIDE_TUGS (OWT)	9 (81.5)	**11 (1680)**
MIXED_NARROW_TUGS (MNT)	10 (202)	**12 (1350)**
ONEWAY_NARROW_TUGS (ONT)	9 (184)	**10 (817)**

Table 3: CPX solver vs. G12 finite domain solver: largest problem solved for each type, with runtime in seconds in brackets. Bold indicates faster method for this problem type.

Table 3 shows that the difference between the CPX and FD solvers is even greater for MIXED problems (less constrained problem with more complex constraints). This indicates that CPX is very fast at dealing with complex constraints, but the speed difference decreases when the problem is tightly constrained. CPX is able to solve larger problems faster for all problem types except for ONEWAY_NARROW without tugs – this is the most constrained problem type, using the simplest constraints (no tugs, and no interaction between incoming and outgoing ships).

The slower performance of CPX on the problem without tugs indicates that there may be room for improvement if better explanations are added for constraints that do not involve tugs, such as the sequence-dependent setup times between ships, and the propagation of the objective function itself. As the tug problem is composed of the NoTugs problem with additional constraints, speeding up the solution time of the NoTugs problem would likely further improve the solution time of the Tugs problem.

Lazy Clause Generation (LCG) is a very general method that has been successfully used to speed up calculation times for many different types of scheduling problems. However, as it is a recent method, its effectiveness for many other problem types has not yet been investigated. Like other complete methods, LCG does not scale well to large problems. However, like traditional CP solvers, LCG solvers can be combined with decomposition techniques or large neighbourhood search to improve scalability [27]. LCG may be worth investigating as an approach to dealing with other routing and scheduling problems with time-dependent task costs, and we plan to compare LCG against an FD solver for the LSFRP in future work.

5 Solve and Improve

Many scheduling and routing approaches initially solve a simplified model, and then use the constraints and objective function of the full problem to improve it. For routing and scheduling problems with time-dependent action costs, removing the time dependence of the objective function is one way to simplify the problem for the first step of a solve-and-improve approach, as used by Lin *et al* [17].

In this section, we solve simplified models for both problems which ignore time-varying action costs, and compare the improvement in runtime against improvements obtained by the LTOP for the LSFRP and the LCG solver for the BPCTOP. The runtimes for the simplified models are a lower bound on the runtime of a full solve-and-improve approach, as the improvement step would increase the runtime further.

5.1 Results for Bulk Port Cargo Throughput Optimisation

We implemented a simplified BPCTOP model by replacing the time-varying objective function by feasible time windows, and solving the problem with the objective of maximising the number of ships scheduled to sail. Table 4 compares the runtimes of the normal and simplified models for the largest problem that was successfully solved by all approaches, as well as for the largest problem solved by any approach. Bold font indicates the fastest runtime, (ie. the approach that results in the largest reduction in runtime).

Table 4 shows that, while the simplified model is faster for small problem sizes, for large problem sizes the CPX solver with Lazy Clause Generation is faster than the simplified model to solve 5 of the 8 problem types, indicating that CPX scales better than the simplified model for 5 of the 8 problem types. As seen earlier in Table 3, CPX is particularly effective on large problems with tugs.

One interesting observation from Table 4 is that the simplified model gives the largest runtime improvement for the most tightly constrained ON and ONT problems, allowing problems with one more ship to be solved within the 30-minute cutoff time. The least tightly constrained MW and MWT problems show only a small improvement in runtime from relaxing the time window penalties; and the moderately constrained

Problem	NShips	Runtime (s)			NShips	Runtime (s)		
Type	(small)	FD	CPX	SIMPLIFIED	(large)	FD	CPX	SIMPLIFIED
MW	11	326	**8.19**	188	16	>1800	**1040**	>1800
OW	11	1480	273	**0.56**	12	>1800	>1800	**26.7**
MN	11	370	**7.64**	180	16	>1800	**1100**	>1800
ON	10	99.9	482	**0.34**	12	>1800	>1800	**19.6**
MWT	10	11.5	17.1	**11.4**	13	>1800	**1290**	>1800
OWT	9	81.5	23.9	**4.60**	11	>1800	**1680**	>1800
MNT	10	202	42.8	**8.21**	12	>1800	**1350**	>1800
ONT	9	184	65.7	**0.69**	10	>1800	817	**262**

Table 4: Runtime (s) of CPX vs. the FD solver with a simplified BPCTOP model.

MN, MNT, OW and OWT problems show moderate improvements. This result is similar to the effect of relaxing hard time windows in a vehicle routing problem [24]. Extending the latest delivery time by 10-20 minutes was found to significantly improve schedule costs for tightly constrained problems. For problems with wide time windows, on the other hand, where the time windows did not constrain the problem, relaxing the time windows had little effect on cost and also increased runtime due to increasing the computational complexity of the problem.

5.2 Results for Liner Shipping Fleet Repositioning

We implemented a simplified LSFRP model by fixing all sailing actions to "slow-steaming", i.e. minimum fuel cost with maximum time. This constraint was subsequently loosened for any instances that did not solve to optimality. The problems were then solved using the LTOP planner [32].

Table 5 shows that, while CPU time for most problem instances was reduced, the improved runtimes were still slower than the best CP model (CP-AOR). We can therefore conclude that fixing the length and cost of the sail action is not very effective for the LSFRP, since the problems where the most benefit can be expected from fixing action costs are those in which the optimal answer uses all slow-steaming actions.

A solve-and-improve approach that simplifies away time-varying action costs was found by Lin *et al* [17] to be effective for satellite imaging scheduling. However, they did not compare the calculation speed of the complete problem against the simplified problem, so there is no indication of the improvement in calculation speed produced by simplifying away the time-varying quality function. However, our experiments found that simplifying the LSFRP and BPCTOP models by removing the time-varying cost function did not produce significant speed improvement, and that switching to a Constraint Programming model or a solver with Lazy Clause Generation was more effective.

It is possible that simplifying the cost function was more effective for speeding up solution times for Linear Programming problems such as [17], rather than for CP or automated planning. However, more work would need to be done to identify the speed improvement produced by simplifying the quality function in [17].

Problem	NORMAL LTOP	SIMPLIFIED LTOP	CP-AOR
AC3_1_0	1.1	0.82	0.1
AC3_2_0	51.0	38.4	0.3
AC3_3_0	188.3	118	0.6
AC3_1_1e	3.9	3.72	0.4
AC3_2_2ce	15.2	15.9	11.3
AC3_3_2c	203.2	265	3.0
AC3_3_2e	217.1	233	13.2
AC3_3_2ce1	218.2	234	13.9
AC3_3_2ce2	192.4	210	17.1
AC3_3_2ce3	516.9	548	470.3
AC3_3_3	80.0	86.6	11.2

Table 5: CPU time (s) of CP vs. LTOP vs. simplified LSFRP with optimal windows.

6 Conclusions and Future Work

While scheduling and routing problems have usually been solved using Mixed Integer Programming (MIP) and solve-and-improve approaches, Constraint Programming (CP) with a good choice of model and search strategy, as well as recent techniques such as Lazy Clause Generation (LCG) have been found to be faster for some problem types.

In this paper, we presented a novel CP model for the Liner Shipping Fleet Repositioning Problem (LSFRP) and compared it against existing MIP and planning models, and found that the CP model was faster than both existing approaches, by an order of magnitude for some instances. CP was also found to be faster than MIP for the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP) in an earlier paper [14]. We also compared a CP solver that uses LCG against a traditional finite domain CP solver for the BPCTOP, and found that LCG was faster for 7 out of 8 problem types. We plan to present a more detailed comparison of different CP models and search strategies, as well as an LCG solver for the LSFRP in a future paper.

Both the problems we investigated in this paper have time-varying action costs, which increase the complexity of the problem. Solve-and-improve approaches have previously used simplified models without time-varying action costs to find initial solutions [17]. We compared the LTOP model of the LSFRP and the full CP model for the BPCTOP against simplified models without time-varying action costs, and found that the speed improvement of removing time-varying costs was less than that obtained by converting the LSFRP to a CP model for all problem instances, and that solving the BPCTOP using an LCG solver scaled better than using a finite domain solver for 5 of 8 problem types, particularly for the most challenging problems with complex tug constraints. While our previous approaches were able to solve realistic-sized problems, the long calculation times limited their usefulness, and the speed improvements presented here may open up new applications such as using the LSFRP as a subproblem of fleet redeployment problems.

In our investigation of both the LSFRP and BPCTOP CP models, we found that the CP model solution time was highly dependent on having a good choice of model and search strategy. However, with this caveat, we find that CP and LCG are efficient and flexible methods that are able to handle complex side constraints, and may therefore be worth investigating for other scheduling and routing problems that are currently being solved using MIP or solve-and-improve approaches.

7 Acknowledgements

The authors would like to acknowledge the support of ANU and NICTA at which Elena Kelareva is a PhD student. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. We would also like to thank OMC International for their support for the research into the port optimisation problem. The research into the fleet repositioning problem is sponsored in part by the Danish Council for Strategic Research as part of the ENERPLAN research project.

References

[1] T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] M. Christiansen and K. Fagerholt. Robust ship scheduling with multiple time windows. *Naval Research Logistics*, 49(6):611–625, September 2002.

[3] M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Chapter 4: Maritime transportation. In C. Barnhart and G. Laporte, editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 189–284. Elsevier, 2007.

[4] M. Christiansen, K. Fagerholt, and D. Ronen. Ship routing and scheduling: status and perspectives. *Transportation Science*, 38(1):1–18, January 2004.

[5] G. Chu, M.G. de la Banda, C. Mears, and P.J. Stuckey. Symmetries and lazy clause generation. In *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming (CP'10) Doctoral Programme*, pages 43–48, September 2010.

[6] A. J. Coles, A. I. Coles, M. Fox, and D. Long. Forward-Chaining Partial-Order Planning. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10)*, May 2010.

[7] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis. VRP with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 7, pages 157–194. SIAM, 2002.

[8] K. Fagerholt. Ship scheduling with soft time windows: an optimisation based approach. *European Journal of Operational Research*, 131(3):559–571, June 2001.

[9] K. Fagerholt. A computer-based decision support system for vessel fleet scheduling - experience and future research. *Decision Support Systems*, 37(1):35–47, April 2004.

[10] K. Fagerholt, G. Laporte, and I. Norstad. Reducing fuel emissions by optimizing speed on shipping routes. *Journal of the Operational Research Society*, 61:523–529, 2010.

[11] T. Feydy and P.J. Stuckey. Lazy clause generation reengineered. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming (CP'09), LNCS*, volume 5732, pages 352–366, 2009.

[12] M.A. Figliozzi. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review*, 48(3):616–636, 2012.

[13] M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20(1):61–124, 2003.

[14] E. Kelareva, S. Brand, P. Kilby, S. Thiébaux, and M. Wallace. CP and MIP methods for ship scheduling with time-varying draft. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 110–118, June 2012.

[15] E. Kelareva, P. Kilby, S. Thiébaux, and M. Wallace. Ship scheduling with time-varying draft: Constraint programming and benders decomposition. *Transportation Science (submitted)*, 2012.

[16] P. Kilby and A. Verden. Flexible routing combing constraint programming, large neighbourhood search, and feature-based insertion. In Kerstin Schill, Bernd Scholz-Reiter, and Lutz Frommberger, editors, *Proceedings 2nd Workshop on Artificial Intelligence and Logistics (AILOG'11)*, pages 43–49, 2011.

[17] W.C. Lin, D.Y. Liao, C.Y. Liu, and Y.Y. Lee. Daily imaging scheduling of an earth observation satellite. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions*, 35(2):213–223, 2005.

[18] N. Nethercote, K. Marriott, R. Rafeh, M. Wallace, and M.G. de la Banda. Specification of Zinc and MiniZinc, November 2010.

[19] N. Nethercote, P.J. Stuckey, R. Becket, S. Brand, G.J. Duck, and G. Tack. MiniZinc: Towards a standard CP modelling language. In C. Bessière, editor, *Principles and Practice of Constraint Programming (CP'07)*, volume 4741 of *LNCS*, pages 529–543. Springer, 2007.

[20] I. Norstad, K. Fagerholt, and G. Laporte. Tramp ship routing and scheduling with speed optimization. *Transportation Research*, 19:853–865, 2011.

[21] O. Ohrimenko, P.J. Stuckey, and M. Codish. Propagation via lazy clause generation. *Constraints*, 14(3):357–391, 2009.

[22] OMC International. DUKC helps Port Hedland set ship loading record. `http://www.omc-international.com/images/stories/press/omc-20090810-news-in-wa.pdf`, 2009.

[23] Port Hedland Port Authority. 2009/10 cargo statistics and port information. `http://www.phpa.com.au/docs/CargoStatisticsReport.pdf`, 2011.

[24] A.G. Qureshi, E. Taniguchi, and T. Yamada. An exact solution approach for vehicle routing and scheduling problems with soft time windows. *Transportation Research Part E: Logistics and Transportation Review*, 45(6):960–977, 2009.

[25] J.G. Rakke, M. Christiansen, K. Fagerholt, and G. Laporte. The traveling salesman problem with draft limits. *Computers & Operations Research*, 39:2161–2167, 2011.

[26] A.H. Russell. Cash flows in networks. *Management Science*, 16(5):357–373, 1970.

[27] A. Schutt, G. Chu, P.J. Stuckey, and M.G. Wallace. Maximising the net present value for resource-constrained project scheduling. In *CPAIOR 2006*, volume 7298 of *LNCS*, pages 362–378. Springer, 2012.

[28] A. Schutt, T. Feydy, P.J. Stuckey, and M.G. Wallace. Solving the resource constrained project scheduling problem with generalised precedences by lazy clause generation. http://arxiv.org/abs/1009.0347, 2010.

[29] T.R. Sexton and Y. Choi. Pickup and delivery of partial loads with soft time windows. *American Journal of Mathematical and Management Science*, 6:369–398, 1985.

[30] S. Smith. Is scheduling a solved problem? In *Multidisciplinary Scheduling: Theory and Applications*, pages 3–17, 2005.

[31] J.-H. Song and K.C. Furman. A maritime inventory routing problem: Practical approach. *Computers & Operations Research*, 2010.

[32] K. Tierney, A. Coles, A. Coles, C. Kroer, A.M. Britt, and R.M. Jensen. Automated planning for liner shipping fleet repositioning. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 279–287, June 2012.

[33] K. Tierney and R.M. Jensen. The liner shipping fleet repositioning problem with cargo flows. In Hao Hu, Xiaoning Shi, Robert Stahlbock, and Stefan Voß, editors, *Computational Logistics*, volume 7555 of *LNCS*, pages 1–16. Springer, 2012.

[34] University of Melbourne. MiniZinc Challenge 2011. `http://www.g12.cs.mu.oz.au/minizinc/challenge2011/challenge.html`, 2011.

[35] M. Vanhoucke, E.L. Demeulemeester, and W.S Herroelen. On maximizing the net present value of a project under renewable resource constraints. *Management Science*, 47:1113–1121, 2001.

[36] M. Wallace. G12 - Towards the Separation of Problem Modelling and Problem Solving. In *CPAIOR 2009*, volume 5547 of *LNCS*, pages 8–10. Springer, 2009.

[37] J. Wang, N. Jing, J. Li, and H. Chen. A multi-objective imaging scheduling approach for earth observing satellites. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*, pages 2211–2218, 2007.

[38] W.J. Wolfe and S.E. Sorensen. Three scheduling algorithms applied to the earth observing systems domain. *Management Science*, 46(1):148–168, 2000.

[39] F. Yao, J. Li, B. Bai, and R. He. Earth observation satellites scheduling based on decomposition optimization algorithm. *International Journal of Image, Graphics and Signal Processing*, 1:10–18, 2010.