EURO J Comput Optim (2014) 2:147–194
DOI 10.1007/s13675-014-0022-7

ORIGINAL PAPER

CP methods for scheduling and routing with time-dependent task costs

Elena Kelareva · Kevin Tierney · Philip Kilby

Received: 7 July 2013 / Accepted: 28 May 2014 / Published online: 3 July 2014

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 such problems. 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 the area of 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 problems with time-dependent task costs and are worth investigating for other scheduling and routing problems that are currently being solved using MIP or solve-and-improve approaches,

E. Kelareva · P. Kilby
Australian National University, Canberra, Australia

E. Kelareva (✉) · P. Kilby
NICTA, Canberra, Australia
e-mail: elena.kelareva@nicta.com.au

K. Tierney
University of Paderborn, Paderborn, Germany
e-mail: kevin.tierney@upb.de

K. Tierney
IT University of Copenhagen, Copenhagen, Denmark

even when customized global constraints are not available. We also investigate a novel approach to solving the BPCTOP—converting the problem into a vehicle routing problem (VRP) and solving using an existing VRP solver.

Keywords Constraint programming · Scheduling · Routing · Maritime logistics

Mathematics Subject Classification 90B06

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 (Smith 2005). 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) (Tierney and Jensen 2012; Tierney et al. 2012b) and the bulk port cargo throughput optimisation problem (BPCTOP) (Kelareva et al. 2012a,c), the objective is a sum of time-varying costs or values for each task. Additional problems include net present value maximization in project scheduling (Russell 1970); satellite imaging scheduling (Lin et al. 2005; Wolfe and Sorensen 2000); vehicle routing with soft time windows (Sexton and Choi 1985; Figliozzi 2012); ship routing and scheduling with soft time windows (Fagerholt 2001; Christiansen and Fagerholt 2002); and ship speed optimisation (Fagerholt et al. 2010; Norstad et al. 2011).

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.

However, for some problems, such as the BPCTOP (Kelareva et al. 2012a), Constraint programming (CP) (see, e.g., Rossi et al. 2006) has been shown to be more effective than using a 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 large neighbourhood search for vehicle routing (Kilby and Verden 2011), SAT (Ohrimenko et al. 2009) and MIP (Achterberg 2009) 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) (Ohrimenko et al. 2009)—a method for solving CP problems which allows the solver to learn where the previous search failed. A CP solver that

uses LCG was found by Schutt et al. (2012) to be more efficient than traditional finite domain CP solvers for a number of scheduling problems (Schutt et al. 2012; Feydy and Stuckey 2009).

When faced with a new problem with time-dependent task costs, or when trying to add such costs to an existing problem, it is not clear which approaches will yield good results. In this paper, we try to provide guidance for researchers and practitioners dealing with problems with such costs. We do this through an investigation of several key problems in the literature and the introduction of models for two maritime transportation problems. To this end, this paper contains the following novel components: (1) a review of methods for solving problems with time-dependent task costs, (2) a CP model of the LSFRP that outperforms all previous approaches, (3) a comparison of LCG and solve-and-improve approaches on the LSFRP, (4) a solve-and-improve approach for the BPCTOP, (5) and a vehicle routing based modelling of the BPCTOP that finds high-quality solutions even on large BPCTOP instances. We note that this article is an expanded version of the conference paper by Kelareva et al. (2013).

In "Background", we review a number of scheduling and routing problems with time-varying action costs, present a summary of problem characteristics and solution approaches, and discuss techniques that may be generalisable between applications, and problems that may benefit from being extended with time-varying action costs.

In the rest of this paper, we adapt several techniques from other applications to two recent scheduling and routing problems with time-varying action costs in the field of maritime transportation. The main contribution of this paper is to compare the effectiveness of CP and LCG against traditional MIP and solve-and-improve techniques for the BPCTOP and LSFRP. "BPCTOP" presents a summary of earlier work on the BPCTOP, including CP and MIP models, as well as a VRP formulation. We compare these models against the new techniques presented in this paper. "LSFRP" summarises earlier work on the LSFRP, including a MIP model and an automated planning approach. This section also presents a novel CP model for the LSFRP and shows that this model is faster than existing approaches. In "Lazy clause generation", we show that a CP solver with LCG is more effective at solving both the BPCTOP and LSFRP than a traditional finite domain CP solver. In "Solve-and-improve", we describe solve-and-improve approaches for the LSFRP and BPCTOP that simplify the time-varying cost function to find an initial solution. Finally, in "Summary", we solve our BPCTOP VRP model using an existing VRP solver, obtaining solutions much more quickly than any previous method, but at the expense of solution optimality.

Background

This section presents a review of a number of scheduling and routing problems with time-dependent action costs, as well as approaches that have been used to solve these problems. A variety of optimal techniques exist for solving routing and scheduling problems, but it is often unclear which technique will offer the best results given a problem with time-dependent task costs. Our goal is to give a detailed overview of the approaches that are available in the literature before we investigate several of these methods in further detail in the rest of this paper. In doing so, we hope to provide some

intuition into which methods perform best on different types of time-dependent task cost problems.

We note that there are other problems where time-dependent task costs are present, e.g., optimization of electricity usage whether in the home (e.g., Agnetis et al. 2013) or in a data center (e.g., Rao et al. 2010). We have selected several key problems where a variety of solution methods have been proposed. We use this information to try to draw conclusions on which directions seem to be the most promising.

Satellite imaging scheduling

One scheduling problem with time-dependent quality functions for each action is the problem of satellite imaging scheduling, where the goal is to schedule a set of observations of an Earth observing satellite. 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 (Wolfe and Sorensen 2000). A satellite scheduling problem also includes resource constraints, as each satellite can only perform one observation at a time, and sequence-dependent setup times, as different observations may require different orientations of the satellite.

Satellite scheduling approaches handle the time-varying image quality objective in a number of ways. One approach used by Yao et al. (2010) and Wang et al. (2007) is to simplify the image quality function by converting it to hard time windows when a "good enough" image could be obtained. This simplifies the problem significantly, but does not account for the fact that images scheduled closer to the centre of the time window will be of higher quality.

Another approach presented by Lin et al. (2005) is to first find a feasible solution by decomposing the problem into independent subproblems using Lagrangian relaxation and then to use heuristics to improve the solution quality by minor changes in the schedule. This approach is suboptimal, but produces better quality solutions than just using feasible time windows, with solutions being within 2 % of optimality.

The quality functions in satellite imaging scheduling may be further complicated by the fact that it is possible to partially satisfy an imaging request, by taking images that cover only a portion of the requested area. The possibility of partial rewards is handled by an objective function which gives the highest weighting to complete requests and only a low weighting to partial requests (Lemaître et al. 2002; Lin et al. 2005).

Wolfe and Sorensen (2000) compared three algorithms for satellite imaging scheduling with time-dependent task quality functions: priority dispatch, look ahead, and genetic algorithms (GA). The priority dispatch approach also used a solve-and-improve method, first building a schedule by allocating jobs in order of priority, then improving the schedule by adjusting job start and end times using a hill-climbing search to improve schedule quality. The look ahead algorithm produced a 12 % average improvement in schedule quality over the priority dispatch algorithm, by using the quality function earlier in the process, to allocate jobs immediately to the peak of their quality window, rather than using the quality function only to improve the initial schedule. The GA approach produced further improvements in schedule quality, but at the expense of a significant reduction in speed.

Simplifying the quality function to hard time windows is the simplest and most efficient approach for decreasing the solution time, but does not account for quality at all. Improving the quality with small local changes to the schedule can produce schedules that are within 2 % of optimality, with only a small time cost (Lin et al. 2005). More complex quality optimisation approaches produce better schedules, but with increasingly slower calculation times (Wolfe and Sorensen 2000) and an approach that finds an exact optimal solution would introduce a much larger time cost.

Optimal approaches may be worthwhile for problems where even a small deviation from the optimum has a significant impact, such as bulk port cargo throughput optimisation (Kelareva et al. 2012a), but close-to-optimal approaches such as that presented by Lin et al. (2005) suffice for applications such as satellite scheduling where a small deviation from the optimum has less impact.

Project scheduling with net present value

Another scheduling problem with time-dependent action costs is the problem of maximising the net present value (discounted cash flow) in project scheduling, first introduced by in Russell (1970). In standard project scheduling, the objective is usually to minimise makespan (total time). For the alternative objective of maximising net present value, each activity has an associated positive or negative cash flow, which is discounted over time according to a given discount rate.

Project scheduling may also include resource constraints and precedence constraints between actions. The objective function component for each action is monotonic, unlike the satellite scheduling image quality function, which has a plateau of peak quality, and drops off before and after.

The problem of maximising net present value in project scheduling without resource constraints can be solved in polynomial time by transforming to a linear program (Grinold 1972). However, the resource-constrained project scheduling problem with net present value (RCPSPNPV) is NP-complete.

The RCPSPNPV can be solved by relaxing the problem to remove resource constraints and using the resource-unconstrained solution as an upper bound in the search. A review of both the resource-constrained and unconstrained max-nvp problem is presented by Vanhoucke et al. (1999).

Vanhoucke et al. (2001) solved the RCPSPNPV using a depth-first branch-and-bound algorithm with the resource-unconstrained solution as the upper bound. The search algorithm scheduled positive-cashflow activities as early as possible in the process, and negative-cashflow activities as late as possible, thus focussing the search on good quality solutions.

Another technique that has been used to solve this problem is CP with lazy clause generation (LCG) (Ohrimenko et al. 2009), which combines constraint programming with learning nogoods from failed search results. LCG was found by (Schutt et al. 2012) to be more efficient at solving this problem than basic constraint programming. CP and LCG are very general approaches that can be used for any constrained optimisation problem, which makes CP and LCG promising candidates for generalising to other scheduling or routing problems with time-varying task costs.

Bulk port cargo throughput optimisation with time-varying draft

A recently introduced maritime scheduling problem with time-dependent action values is the bulk port cargo throughput optimisation problem (BPCTOP) with time-varying draft restrictions. Many ports have safety restrictions on the draft (distance between waterline and keel) of ships sailing through the port. At most ports, these restrictions vary with the height of the tide, as well as with other environmental conditions. Most maritime scheduling problems either ignore draft constraints entirely (Fagerholt 2004), or do not consider time variation in draft restrictions (Rakke et al. 2011; Song and Furman 2010). 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 5 min. 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.

Approaches for dealing with the time-varying quality function in the BPCTOP are similar to satellite scheduling—traditional ship scheduling problems with constant draft are similar to satellite scheduling with time windows of "good enough" quality, whereas optimisation with time-varying draft is similar to the approach of optimising a weighted sum of time-varying satellite images qualities used by Lin et al. (2005).

As with satellite scheduling, ship scheduling involves a tradeoff between solution quality and scalability. Traditional ship scheduling techniques with constant draft are the most scalable, but do not consider time variation in the allowable amount of cargo.

Ship scheduling with time windows, such as the approaches presented by Christiansen and Fagerholt (2002), Fagerholt (2001) could be used to consider time windows of "good enough" draft, at the cost of increased complexity. This would allow ships to be scheduled even if they cannot sail at all phases of the tidal cycle. However, this approach would still produce schedules with less than the maximum loading for ships that are large enough to be able to load extra cargo to the peak of the tide. Optimal approaches, such as those discussed by Kelareva et al. (2012a), produce higher quality schedules, but are significantly more complex and less scalable to large problems.

Kelareva et al. (2012a) 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 (Kelareva et al. 2012c). Each extra centimetre of draft can increase profit for the shipper by up to $10,000 (Kelareva et al. 2012a), so there is a high incentive to find optimal solutions, and non-optimal approaches for this problem have not yet been investigated.

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) Tierney et al. (2012b). The LSFRP concerns itself with the repositioning of container ships in a liner shipping network when routes in the network are added or changed. We describe the LSFRP in detail in "LSFRP".

The LSFRP is a relatively new problem to the optimisation community, having not been mentioned in any of the most comprehensive surveys of maritime research (Christiansen et al. 2004, 2007, 2013).

The LSFRP was first solved by Tierney et al. (2012b), who compared two planning approaches and a MIP model for solving the problem. First, the LSFRP was modelled using PDDL 2.1, a modelling language for automated planning problems (see Fox and Long 2003) and solved using the POPF planner (Coles et al. 2010). Although POPF found solutions, it could not solve the LSFRP to optimality. The authors further introduced a MIP based on an activity-flow approach, in which the various activities that can be undertaken by vessels are modelled as nodes, and arcs represent which activities can be ordered after one another. The authors then introduced linear temporal optimisation planning (LTOP), which uses temporal planning to build optimisation models and solves these using an optimisation version of partial-order planning based on branch-and-bound. LTOP outperformed both the traditional planning approach and the MIP on a limited dataset of instances.

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 (Tierney et al. 2012b).

Tierney and Jensen (2012, 2013) and Tierney (2013) solve a version of the LSFRP including cargo flows, which the version we address in this paper lacks. In these two papers, the authors sacrifice the flexibility of choosing amongst several phase-in and phase-out opportunities to lower the overall problem size enough to include cargo flows. Therefore, the problem we address in this work is an equally important, parallel problem to the LSFRP with cargo flows. We note that the version of the LSFRP presented in this work is based on activities that avoid disrupting cargo flows, even though they are not taken explicitly into account.

Ship scheduling with soft time windows

Other ship scheduling and routing problems have also involved time-varying action costs. Fagerholt (2000, 2001) modelled a ship routing and scheduling problem as a multi-port pickup-and-delivery problem with soft time windows. Soft time window penalties were used to find better schedules with lower shipping costs by allowing deliveries outside a customer's preferred time window. Different penalty functions for deliveries outside the preferred window were compared, which could be used for different situations, e.g., when a customer gets a fixed discount for a delivery outside

the preferred time, or where the discount depends on the lateness of the delivery. This problem was inspired by similar approaches to vehicle routing with soft time windows, which were found to produce better schedules by allowing controlled violation of delivery time windows. This problem was solved by generating feasible shipping routes, then generating possible schedules (with corresponding costs) for each route, and finally using the schedules as input to a set partitioning problem. This approach resulted in a 15 % reduction in shipping costs by shipping 12 % of cargo outside the preferred delivery times (Fagerholt 2000).

Another ship scheduling problem with time-dependent action costs was presented by Christiansen and Fagerholt (2002), who considered penalties for ships arriving at port close to weekends, to avoid the risk of long delays caused by some ports closing on weekends. This problem included multiple soft time windows—a situation rarely explored in the literature, with the exception of a few vehicle routing problems. The penalty function peaked before weekends and dropped to 0 at the start of each week. The objective function was a weighted sum of the transportation costs and the penalty costs for "risky" arrival times. This problem was also solved using a set partitioning approach.

Another type of ship scheduling and routing problem with time-dependent action costs is the problem of ship routing and scheduling with speed optimisation, investigated by Norstad et al. (2011). The cost function for each ship sailing is monotonic and increases with sailing speed, similarly to the LSFRP. This problem is solved by discretising ship arrival times at each port, generating feasible solutions, and iteratively improving them using local search. Problems for a single shipping route were solved successfully, and significant profit increases were found due to ships being able to carry extra cargoes by sailing faster, or use less fuel by sailing slower.

Vehicle routing with soft time windows

Time-varying action costs are also often found in the domain of vehicle routing, particularly in the vehicle routing problem with soft time windows (VRPSTW), which includes penalties for early and late arrival outside each customer's preferred time window. In vehicle routing, time windows allow the modelling of real-world constraints on delivery deadlines. However, hard time windows often result in tightly constrained problems which require a larger fleet of vehicles to meet deadlines. Soft time windows allow better utilisation of vehicles, thus reducing transportation costs, while still servicing most customers within their preferred time windows. However, soft time windows result in a more complex objective function, making the problem significantly more difficult to solve (Qureshi et al. 2009).

Some VRPSTW approaches optimise first for the number of vehicles (routes), then for minimal travel time and distance, and then for minimal cost only as a final objective with time window penalties included (Figliozzi 2012). Others optimise for a combined weighted objective that includes all of the above costs (Qureshi et al. 2009; Fan et al. 2011). Some VRPSTW approaches include penalties for both late and early arrival (Figliozzi 2010; Fan et al. 2011), whereas others only include penalties for lateness, with no penalties for arriving early (Qureshi et al. 2009).

Figliozzi (2010) solved the VRPSTW using an iterative route construction and improvement algorithm. First, a set of feasible routes are constructed sequentially by adding unrouted customers to partial solutions; then the initial routes are improved iteratively. The first improvement step aims to reduce fleet size and routing costs by merging routes with underutilised vehicles. The second route improvement step aims to reduce total route duration and time window penalties by improving customer service times.

While this method is not optimal, it resulted in improved solutions for a number of VRPSTW benchmarks and showed a clear trade-off between solution quality and calculation time. This approach was further extended by Figliozzi (2012) to be able to take into account time-dependent travel times, which result in much more complex constraints between tasks.

Qureshi et al. (2009) investigated an exact solution approach for the VRPSTW based on column generation and compared results of different levels of time window relaxation against solutions with hard time windows. Penalties were only applied for late arrival, not for early arrival, leading to monotonic cost functions for each task.

Kilby and Verden (2011) investigated an approach to vehicle routing that combined large neighbourhood search (Shaw 1997) with CP. Their Indigo vehicle routing solver not only implements some constraints natively, for faster calculation, but also integrates with a CP solver to allow flexible side constraints such as a "blood bank" constraint requiring that deliveries be made within 20 min of pickup. Indigo implements time window constraints natively and can consider soft time windows with linear early and late penalties. The CP system uses backtracking search with Indigo acting as the variable/value choice heuristic: if a variable/value assignment made by Indigo leads to a CP constraint being violated, this information gets propagated back to Indigo, to prevent the Large Neighbourhood Search from making the same assignment in future. The Large Neighbourhood Search destroys part of the solution in each iteration and uses insertion heuristics to repair the partial solution.

The VRPSTW has received more attention in the literature than other problems with time-varying costs, so a number of efficient VRP solvers exist that can handle soft time windows. As a result, one of the new approaches we investigate in this paper is modelling a problem with time-varying costs as a VRPSTW to see if VRP solvers can be used to efficiently solve scheduling problems in other fields.

Summary

Each of the problems we reviewed in this section involves time-dependent action costs; however, other constraints as well as the shapes of these objective functions vary between problems. Most problem types include resource constraints and setup times between tasks; however, not all problems include sequence-dependent setup times. Some problems such as ship scheduling and vehicle routing also include optional tasks, whereas for other problems such as project scheduling, the set of tasks is fixed. Finally, some of the problems also include routing constraints. Table 1 summarises additional constraints and objective function shapes for problems reviewed in this paper.

Table 1 Characteristics of problems reviewed in the literature

Paper	Problem	Objective function	Seq. dep. setups	Routing	Optional
Wolfe and Sorensen (2000)	Satellite	Peaked	Yes	No	Yes
Lin et al. (2005)	Satellite	Peaked	Yes	No	Yes
Wang et al. (2007)	Satellite	TimeWindows	Yes	No	Yes
Yao et al. (2010)	Satellite	TimeWindows	Yes	No	Yes
Schutt et al. (2012)	Project	Monotonic	No	No	No
Grinold (1972)	Project	Monotonic	No	No	No
Vanhoucke et al. (2001)	Project	Monotonic	No	No	No
Kelareva et al. (2012b)	BPCTOP	Peaked	Yes	Yes	Yes
Fagerholt (2001)	ShipSTW	Peaked	No	Yes	No
Christiansen and Fagerholt (2002)	ShipSTW	Monotonic, Multiple windows	Yes	Yes	No
Norstad et al. (2011)	ShipSpeedOpt	Monotonic	No	Yes	No
Bausch et al. (1998)	ShipSpeedOpt	Monotonic	No	Yes	No
Brown et al. (1987)	ShipSpeedOpt	Monotonic	No	Yes	No
Tierney et al. (2012b)	LSFRP	Monotonic	No	Yes	Yes
Tierney and Jensen (2011)	LSFRP	Monotonic	No	Yes	Yes
Figliozzi (2010)	VRPSTW	Peaked	Yes	Yes	No
Figliozzi (2012)	VRPSTW	Peaked	Yes + time dep.	Yes	No
Qureshi et al. (2009)	VRPSTW	Monotonic (late penalties only)	Yes	Yes	No
Fan et al. (2011)	VRPSTW	Peaked	Yes	Yes	No
Kilby and Verden (2011)	VRPSTW	Peaked	Yes	Yes	Yes

Some approaches to dealing with time-dependent action costs, such as (Rakke et al. 2011), involve simplifying the objective function by modelling it as constant over time or by reducing the time-varying action costs to hard "good enough" time windows (Yao et al. 2010). However, in this paper, we will focus on approaches that do consider time-dependent objectives, whether by optimising for the time-varying objective directly, such as Kelareva et al.'s (2012b) solution to the bulk cargo port throughput optimisation problem or by solving a simplified constant-time problem initially and then improving the solution with respect to the time-varying objective, such as Lin et al.'s (2005) approach to satellite imaging scheduling.

The choice of an appropriate algorithm needs to be informed by how important it is to get a good quality solution for the given application. In optimising the cargo throughput for a bulk export port, problem sizes are small—there may only be up to 10 ships sailing on a tide—but solution quality is very important, as every centimetre of extra draft for a ship results in a $10,000 increase in profit (Kelareva et al. 2012b). In the LSFRP, varying the speed of a single vessel can result in savings of tens or even hundreds of thousands of dollars. In satellite image scheduling, however, problems sizes may be very large with thousands of tasks to be scheduled (Wolfe and Sorensen 2000), but the consequences of a suboptimal solution may be less expensive, so a method that produces solutions within 2 % of optimality such as that presented by Lin et al. (2005) is sufficient.

We conclude from our overview of the literature that time-dependent task costs can often be tackled directly without being simplified or abstracted out of the problem. Such models can provide practitioners with more accurate decision support over discretised or otherwise simplified approaches. In particular, we note the effectiveness of constraint programming approaches for solving such problems, even in the absence of customized propagators, a theme we will further explore in this work.

As there are many similarities between scheduling and routing problems involving cost functions that are a sum of time-varying task costs, some techniques may be generalisable from one problem type to others. In the next three sections, we discuss techniques that have been successfully used for one of the problem types discussed in this section that may be generalisable to other problems and apply these technique to ship scheduling with time-varying draft constraints and to the liner shipping fleet repositioning problem.

In particular, we present comparisons of CP versus MIP models for both the BPCTOP and LSFRP; we compare a CP solver with lazy clause generation (LCG) against a traditional finite domain solver for both problems; we present simplified models without time-varying costs and investigate the speedup obtained by removing these costs for both problems; and we model the BPCTOP as a vehicle routing problem, to investigate the effectiveness of state-of-the-art VRP solvers for scheduling problems in related fields.

BPCTOP

Kelareva et al. (2012a) 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 (Kelareva et al. 2012a,c). This section briefly summarises the results presented in (Kelareva et al. 2012a,c), which are then compared against new approaches for the BPCTOP in the rest of the paper.

Bulk port cargo throughput optimisation

The allowable sailing times for a ship arriving or departing a port depend on the draft and other dimensions of the ship, the times when other ships are scheduled to sail, and a complex set of factors affecting the under-keel clearance (amount of water under the ship's keel). Factors affecting under-keel clearance include the tide, waves, currents, and wind; other environmental conditions such as water salinity may also need to be taken into account at some ports. O'Brien (2002) presents a detailed analysis of under-keel clearance calculation at a port; however, for the purposes of this paper, we simplify these calculations down to a function that specifies the maximum allowable sailing draft for each ship at a range of discretised times.

Ports may also 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 and the order in which they sail. If a ship departing from a berth at the back of the port sails first, a second ship sailing from a berth close to the channel exit will have to wait for it to pass. If the ships sail in the opposite order, however, they may be far enough apart at their starting locations to be able to start sailing simultaneously.

Some ports may also have resource constraints on the availability of tugs—small boats used to assist large ships in entering and leaving a port. At Port Hedland, tugs are required to assist every ship over a given size, which includes all draft-restricted ships. When there are many ships sailing on a single tide, the number of tugs available may form a bottleneck in the ship scheduling problem, so tug constraints need to be included in the model.

Tug constraints are complex and depend on the order in which ships sail, the origins and destinations of successive ships, and the safety rules in place at the port specifying the number of tugs required for each type of ship. Kelareva et al. (2012a) investigated several ways of modelling the tug constraints, and the best approach found in Kelareva et al. (2012a) is summarised here.

Our approach to modelling tugs uses features of the problem to split the tug constraints into four scenarios, each of which can be considered separately. The port is assumed to have one channel, which ships can traverse in only one direction at a time, inbound or outbound—the channel is not wide enough for ships travelling in different directions to pass each other safely. The berths are close together compared to the length of the channel, so the travel time for tugs assisting a ship moving from sea to berth can be considered to be independent of the location of the berth. (Note: the tug travel time we allow for in our scheduling problem must in any case be conservative

enough to ensure tugs do not end up running late to a second ship after assisting their first ship).

These assumptions are valid for Port Hedland, the port we considered in our case study in Kelareva et al. (2012a), but for some ports worldwide this may not be the case, so the BPCTOP model based on these assumptions may need to be extended to be usable at these ports.

These assumptions enable us to split the ship schedule into four scenarios—a sequence of outbound ships, a sequence of inbound ships, an inbound ship followed by an outbound ship, or an outbound ship followed by an inbound ship. The tug constraints can consider each scenario independently, as only one scenario can be occurring at any given time in the schedule. This greatly simplifies the tug constraints compared to any approach investigated in Kelareva et al. (2012a) that considers all four scenarios at once and allows realistic sized problems to be solved to optimality. Both the CP and MIP model presented below use this method of modelling tug constraints.

CP model for BPCTOP

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. (1).

$$sb(v_i, v_j) = 1 \leftrightarrow (t(v_i) < t(v_j) \wedge sb(v_j, v_i) = 0), \quad \forall\, v_i,\ v_j \in V; v_i \neq v_j \quad (1)$$

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 versus 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.

Parameters The following tables summarise the parameters and variables used in our CP model for easy reference. The model uses the following parameters:

V	Set of vessels
$[1, T_{\max}]$	Range of discretised time indices
$E(v)$	Earliest departure time for vessel $v \in V$
$D(v, t)$	Maximum allowable draft for vessel $v \in V$ at time t
$C(v)$	Tonnage per centimetre of draft for vessel $v \in V$
$ST(v_i, v_j)$	Minimum separation time between vessels $v_i, v_j \in V$
B	Set of pairs of incoming and outgoing ships using the same berth
$B_i(b)$, $B_o(b)$	Incoming and outgoing ships in pair $b \in B$
$L(b)$	Minimum delay between the sailing times of vessels $B_i(b)$, $B_o(b)$ for $b \in B$
U_{max}	Number of tugs available
O	Set of outgoing ships
$G(v)$	Number of groups of tugs required for vessel $v \in V$
$H(v, g)$	Number of tugs in group g of vessel v
G_{max}	Maximum number of groups of tugs required for any vessel $v \in V$
$R(v, g)$	"Turnaround time"—the time taken for tugs in group g of vessel $v \in V$ to become available for another ship in the same direction (incoming versus outgoing)
$X(v_i, v_j)$	Extra time required for tugs from incoming vessel $v_i \in I$ to become available for outgoing vessel $v_o \in O$

Variables The variables used in the model are as follows:

$s(v)$	Binary variable specifying whether vessel $v \in V$ is included in the schedule
$t_v \in [1, T_{\max}]$	The scheduled sailing time for vessel $v \in V$
$sb(v_i, v_j)$	SailsBefore(v_i, v_j)—true iff vessel v_i sails before vessel v_j, where $v_i, v_j \in V$
$u(v, t, g)$	Number of tugs busy in tug group g of vessel $v \in V$ at time t, assuming the next ship for these tugs is in the same direction
$x(v, t)$	Number of extra tugs busy at time t for an outgoing vessel $v \in O$, due to still being in transit from the destination of an earlier incoming ship
$l(v, t)$	Binary variable specifying that the extra tug delay time for incoming vessel $v \in I$ overlaps with the sailing time of an outbound vessel at time t

Objective and constraints The model uses the following objectives and constraints:

$$\text{maximise} \sum_{v \in V} s(v) \,.\, C(v) \,.\, D(v, t_v) \tag{2}$$

$$\text{subject to} \quad s(v) = 1 \Rightarrow t_v \geq E(v), \quad \forall\, v \in V \tag{3}$$

$$s(B_i(b)) = 1 \Rightarrow \tag{4}$$
$$s(B_o(b)) = 1 \wedge t(B_o(b)) \leq t(B_i(b)) - L(b), \quad \forall\, b \in B$$

$$sb(v_i, v_i) = 0, \quad \forall\, v_i \in V; v_i \neq v_j \tag{5}$$

$$s(v_i) = 1 \ \wedge s(v_j) = 1 \Rightarrow \tag{6}$$
$$(sb(v_i, v_j) = 1 \Rightarrow t(v_j) - t(v_i) \geq ST(v_i, v_j)), \quad \forall\, v_i,\ v_j \in V$$

$$s(v) = 1 \wedge t \geq t_v \wedge t < t_v + R(v, g) \Rightarrow u(v, t, g) = H(v, g), \tag{7}$$
$$\forall\, v \in V,\ t \in [1, T_{\max}],\ g \in [1, G_{\max}]$$

$$s(v) = 0 \vee t < t_v \vee t \geq t_v + R(v, g) \Rightarrow u(v, t, g) = 0, \tag{8}$$
$$\forall\, v \in V,\ t \in [1, T_{\max}],\ g \in [1, G_{\max}]$$

$$x(v_o, t) = 0, \quad \forall\, v_o \in O,\ t \in [1, T_{\max}] \tag{9}$$

$$l(v_i, t) \iff \exists\, v_o \in O \ \ s.t. \tag{10}$$
$$t = t(v_o) \wedge t(v_i) \leq t(v_o) \wedge$$
$$t(v_i) + \max_{g \in [1, G(v_i)]} R(v_i, g) + X(v_i, v_o) > t(v_o),$$
$$\forall\, v_i \in I,\ t \in [1, T_{\max}]$$

$$x(v_i, t) = \text{bool2int}(l(v_i, t)). \sum_{g \in [1, G(v_i)]} H(v_i, g), \tag{11}$$
$$\forall\, v_i \in I,\ t \in [1, T_{\max}]$$

$$\sum_{v \in I} \sum_{g \in G(v)} u(v, t, g) \leq U_{\max}, \quad \forall\, t \in [1, T_{\max}] \tag{12}$$

$$\sum_{v_o \in O} \sum_{g \in G(v_o)} u(v_o, t, g) + \sum_{v_i \in I} x(v_i, t) \leq U_{\max}, \tag{13}$$
$$\forall\, t \in [1, T_{\max}]$$

The objective function (2) maximises total cargo throughput for the set of ships. Constraint (3) specifies the earliest departure time for each vessel. Equation (4) ensures that the berths for any incoming ships are empty before the ship arrives. Equation (5) specifies that no ship sails before itself. Equation (6) ensures that there is sufficient separation time between successive ships to meet port safety requirements.

Equations (7)–(13) 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 four independent scenarios. Equations (7) and (8) specify that the number of tugs used by any one-directional sequence of ships (incoming/outgoing) at any time must not exceed the number of tugs available. Equations (10) and (11) specify that any tugs used for an incoming ship are not available for an outgoing ship until the extra delay required for tugs to travel between berths has passed. Equation (9) specifies that there is no extra delay needed for tugs moving from an outgoing ship to an incoming ship. Finally, Eqs. (12) and (13) specify that the total number of tugs in use at any time by both incoming and outgoing ships must be no greater than the number of tugs available at the port.

The tug constraints for each scenario are similar to a *cumulative* constraint, and in fact Eq. (12) ensuring that there are enough tugs for incoming ships can be represented by a *cumulative* constraint. However, the constraints for ensuring sufficient tugs for outgoing ships are more complex. In particular, if these constraints were to be represented by a *cumulative* constraint, the starting time and duration of each outgoing ship task would depend on the starting times of all incoming ship tasks, thus making the set of constraints much more complex than a standard *cumulative* constraint.

MIP model for BPCTOP

While the CP model was able to solve realistic size problems to optimality in a reasonable time frame, we also implemented a mixed integer programming (MIP) model for the BPCTOP, to investigate whether MIP or CP would scale better to larger problems. The MIP model introduced in Kelareva et al. (2012a) is similar to the CP model, but with non-linear constraints converted to linear forms.

Variables The MIP model uses the same parameters and variables as the CP model, but adds the following additional variables:

$s(v,t)$	Binary variable which specifies whether vessel $v \in V$ is scheduled to sail at time $t \in [1, T_{\max}]$
t_v	Time slot when vessel $v \in V$ is scheduled to sail, given by Eq. (14)

$$t_v = \sum_{t \in [1, T_{\max}]} s(v,t) \,.\, t, \quad \forall\, v \in V \tag{14}$$

Objective and constraints The objective function and Eqs. (9), (12) and (13) are unchanged from the CP model. Other CP constraints need to be converted to linear form for the MIP model. Modified constraints are shown below.

$$\sum_{t \in [1, T_{\max}]} s(v,t) \leq 1, \quad \forall\, v \in V \tag{15}$$

$$t_v \geq E(v), \quad \forall\, v \in V \tag{16}$$

$$t(B_o(b)) \leq t(B_i(b)) - L(b) \tag{17}$$
$$\sum_{t \in [1, T_{\max}]} s(B_o(b), t) \geq \sum_{t \in [1, T_{\max}]} s(B_i(b), t), \quad \forall\, b \in B$$

$$s(v_i, t) + \sum_{t' \in [t, \min(T_{\max}, t + ST(v_i, v_j) - 1)]} s(v_j, t') \leq 1 \tag{18}$$
$$s(v_j, t) + \sum_{t' \in [t, \min(T_{\max}, t + ST(v_j, v_i) - 1)]} s(v_i, t') \leq 1$$
$$\forall\, v_i,\ v_j \in V,\ t \in [1, T_{\max}]$$

$$u(v,t,g) = H(v,g). \sum_{t' \in [\max(1, t-R(v,g)+1), t]} s(v,t') \tag{19}$$
$$\forall\, v \in V,\ t \in [1, T_{\max}],\ g \in [1, G_{\max}]$$

$$x(v_i,t) = l(v_i,t). \sum_{g \in [1, G(v)]} H(v_i,g) \tag{20}$$
$$\forall\, v_i \in I,\ t \in [1, T_{\max}]$$

$$l(v_i,t) \geq s(v_o,t) + \sum_{t' \in [\max(1, t-t_{\text{range}}), t]]} s(v_i,t) - 1 \tag{21}$$
$$\forall\, v_i \in I,\ v_o \in O,\ t \in [1, T_{\max}], \text{ where}$$
$$t_{\text{range}} = t - X(v_i, v_o) - \max_{g \in [1, G(v_i)]} (R(v_i,g)) + 1$$

Equation (15) specifies that each ship can sail at most once in the time range. Constraint (16) specifies the earliest departure time for each vessel. Equation (17) ensures that the berths for any incoming ships are empty before the ship arrives. Equation (18) ensures that there is sufficient separation time between successive ships to meet port safety requirements.

Equations (19)–(21) describe the modified tug constraints. Equation (19) specifies that the number of tugs used by any one-directional sequence of ships (incoming/outgoing) at any time must not exceed the number of tugs available. Equations (20) and (21) specify that any tugs used for an incoming ship are not available for an outgoing ship until the extra delay required for tugs to travel between berths has passed.

CP versus MIP

These models were formulated in MiniZinc 1.4 and solved with the G12 2.0 finite domain CP solver (Nethercote et al. 2010, 2007) and MIP OSI CBC solver (Lougee-Heimer 2003).

Kelareva et al. present a comparison of runtimes for both solvers, tested on a set of problems 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 (University of Melbourne 2011). The CP model had significantly shorter runtimes than the MIP model for all problems, and the CP solver was able to solve larger problem sizes within the cutoff time for every problem type, with up to 4 more ships able to be scheduled.

The G12 FD solver 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 (OMC International 2009).

Conversion to vehicle routing

Of all scheduling and routing problems with time-varying task costs, the vehicle routing problem with soft time windows (VRPSTW) is one of the most thoroughly researched, comprising about 90 % of the literature on scheduling and routing problems with soft time windows. Since the vehicle routing domain has been investigated more extensively than other scheduling and routing problems with time-varying task costs, there are several fast, efficient solvers for the VRPSTW in existence.

Many scheduling problems, particularly those without complex side constraints, can be modelled as a VRP, which brings us to the idea that for some scheduling problems, converting them to VRPs and solving using an existing VRP solver may be more efficient than using a generic CP or MIP solver.

The VRPSTW consists of a number of jobs that must be completed by a set of vehicles.[1] Each job is located at a node in a graph and must be serviced within a given time window, with early or late arrival at the job being penalized proportionally to the time window violation. Jobs are additionally associated with a profit value that varies depending on when the job is completed. Arcs connect graph nodes and have a fixed travel time and cost. The goal of the problem is to find a tour for each vehicle such that every job is assigned to a vehicle and the sum of the earned job values minus the costs of the tours is maximal.

We now investigate converting the BPCTOP costs to a VRPSTW. The basic BPCTOP without tug availability constraints can be converted easily to a VRPSTW; however, the complex tug constraints are more difficult to convert to vehicle routing constraints. Thus, we only show how to convert the BPCTOP to the VRPSTW when ignoring tug constraints. The model is as follows:

- Each ship is modelled as a job.
- The channel is modelled as a single vehicle that must be used to complete all jobs.
- The sequence-dependent separation times between ships are modelled as direction-dependent travel times between successive jobs.
- Each job has an earliest time when it can be completed, corresponding to the ship's earliest sailing time.
- The value of each job request corresponds to the weighted draft of the ship.
- For each request, soft time window cost penalties are specified corresponding to the reduction in draft if the ship sails outside its maximum draft window.
- The objective of the VRP is to maximise the total value of all fulfilled requests.

One of the advantages of this VRP conversion is that the VRP graph size is relatively small. Each node in the graph represents a ship, meaning the routing aspect of the problem is not very hard to solve. The complication comes mainly in the soft time window penalties, which must be overcome with specialized VRPSTW heuristics.

[1] Some versions of the VRPSTW minimize the number of vehicles used; here, we assume this value is fixed.

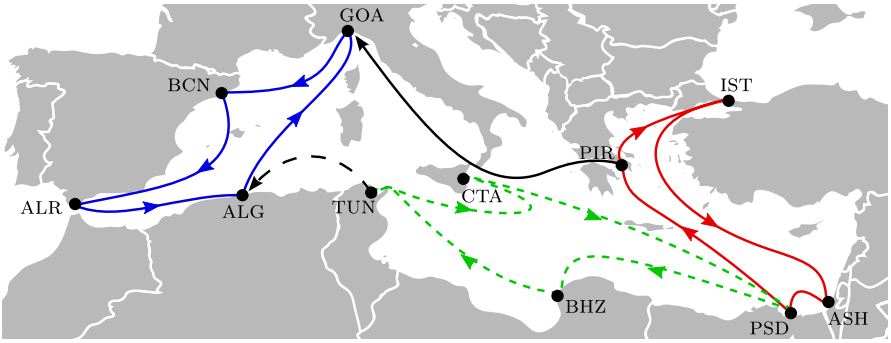

Fig. 1 An example repositioning scenario with two vessels (*solid and dashed black lines*) from their initial services to the goal service

LSFRP

In this section, we describe the LSFRP and present a novel CP model. We compare the CP model against the MIP and automated planning models introduced in Tierney et al. (2012b) on a dataset of 39 instances based on data from our industrial collaborator. We show that CP greatly outperforms previous approaches. We begin the section with an overview of the previous approaches, followed by our CP model and experimental results.

Fleet repositioning

Liner shipping networks must be regularly updated to adjust them to seasonal demands and shifts in the world economy. In order to do this, shipping lines *reposition* ships between *services* in the network. A service is a cyclical route consisting of a sequence of ports with fixed visitation times. Services can be thought of like bus routes, in that each port is visited on a regular (i.e., periodic) frequency. Many shipping lines set this frequency to be one or two weeks, meaning each port on a service is visited by a vessel every week or every other week. This regularity allows liner shippers, who are the customers of shipping lines, to more easily integrate cargo shipments into their international supply chains.

Given a set of vessels and a goal service, where each vessel is assigned an initial service, the aim of the LSFRP is to reposition each vessel to the goal service within a given time period at minimal cost. Figure 1 shows an example repositioning at an abstract level. Two vessels (black lines) are repositioned to their goal service (blue, left).

A vessel begins repositioning when it *phases out* from its original service and ends repositioning when it *phases in* to its new service. Vessels must phase in to their goal service such that the liner shipping structure of the service is maintained. That is, a periodic (often, but not always, weekly) schedule must be created. To ensure this is the case, vessels all phase-in in the same port in a sequential, week-by-week fashion.[2]

[2] Using the same port for a phase-in is an assumption of this model and is not always necessary in practice.

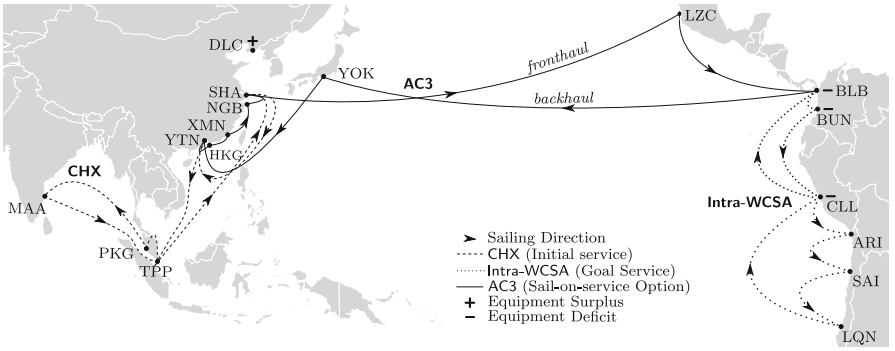

Fig. 2 A subset of a real-world repositioning scenario, from Tierney et al. (2012b)

The total cost of the repositioning depends on the fuel usage from sailing the vessel between ports and a fixed hourly cost, called the *hotel cost*, that is paid for the time between the phase-out and phase-in. The hotel cost represents the baseline costs for keeping a ship running, such as a paying the crew, purchasing food and parts, etc. Minimizing the hotel cost ensures that the sailing costs are balanced against the costs of running the vessel, i.e., not too much time is spent in a non-revenue generating repositioning.

The cost function for a sailing action decreases as the duration of a sailing increases because less fuel is needed to sail at slow speeds than at fast speeds. We use a linear approximation of vessel fuel consumption in this work. The cost of sailing also depends on whether the vessel sails empty or carries empty containers. Carrying empty containers is called a *sail-equipment* (SE) opportunity. Another option for repositioning vessels is to replace another vessel in a regular service, which is called a *sail-on-service* (SoS) opportunity. SoS opportunities significantly reduce the cost of a repositioning, but may increase sailing duration due to the delay in loading and unloading cargo. The times at which these sailings may be performed may also be limited. Our model also includes *cabotage restrictions*, which prevent ships from violating laws in certain countries that protect domestic shipping routes from international competition. The LSFRP is not a pure scheduling problem, as there is a choice of actions and action durations. Note that in this work we do not take a detailed view of cargo flows as in Tierney et al. (2014), Tierney and Jensen (2012), however, the modelling of the problem does attempt to avoid significant disruptions to the overall flows of the network.

Figure 2 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.

Automated planning and linear temporal optimization planning (LTOP)

Automated planning is used to model problems where it is difficult to select and sequence activities to achieve specific goals starting from an initial state. This is done

through a state-based search in which real-world activities are modelled as *planning actions* that can be applied only if a set of preconditions hold and when applied change the state according to a set of effects. We use a state variable representation of planning (Fikes and Nilsson 1971) in which variables with finite domains hold information about the current status of all agents or actors in a planning problem, e.g., the position of a liner shipping vessel or what is loaded on the vessel. We refer the reader to (Nau et al. 2004) for the details of automated planning. Automated planning offers a distinct advantage over other modelling approaches in the way actions correspond directly to real-world activities, making models easy to understand and interpret for domain experts.

LTOP

LTOP uses a type of automated planning called partial-order planning (Penberthy and Weld 1992) to build and search through optimization models that involve continuous time, metric quantities, and a complex mixture of action choices and ordering constraints. LTOP fundamentally diverges from classical automated planning approaches by introducing two sets of modelling variables that decouple the planning problem from the optimization model. Thus, the optimization model is not tightly bound to the semantics of actions. Actions are merely used as handles to optimization components that are combined to complete optimization models using partial-order planning. In other words, LTOP builds optimization models with the help of automated planning, i.e., the focus of LTOP is on the optimization model of a problem, whereas classical planning approaches include optimization components within actions merely as a byproduct of the overall planning process.

As mentioned, LTOP is built on a state variable representation of propositional STRIPS planning. LTOP utilizes partial-order planning (Penberthy and Weld 1992) and extends it in several ways. First, an optimization model is associated with each action in the planning domain. This allows for complex objectives and cost interactions that are common in real-world optimization problems to be easily modelled. Second, instead of focusing on simply achieving feasibility, LTOP minimizes a cost function. Finally, begin and end times can be associated with actions, making them durative. Such actions can have variable durations that are coupled with a cost function. A branch-and-bound search is used to find the plan with the minimal (or maximal) cost.

We briefly describe how the LTOP approach works, but refer to Tierney et al. (2012b) for a formalization of the approach. Algorithm 1 shows the branch-and-bound procedure of LTOP. The LTOP function is initialized with an initial and goal state, $\mathcal{I}$ and $\mathcal{G}$, respectively. The initial state is a complete assignment of state variables to values in their domain, whereas the goal state consists of an assignment of a subset of all the variables. An empty plan is generated on line 2, and the best plan, π_{best} is initialized to null. The algorithm then enters a loop which ensures that every possible plan must be explored. On line 5 a plan is selected from the set of plans that must be explored, Π. The mechanism behind this selection is irrelevant to the completeness of the algorithm, i.e., it can use a lowest cost heuristic or other mechanism. On line 7 the plan is checked to see if it has any *flaws*, which are problems in the plan that must be fixed in order for the plan to connect the initial state to the goal state in the state

Algorithm 1 Linear Temporal optimization planning algorithm, from Tierney et al. (2012b).

```
1: function LTOP(I, G)
2:     Π ← {INITIALLTOP(I, G)}                          ▷ Empty plan initialization
3:     π_best ← null
4:     while Π ≠ ∅ do
5:         π ← SELECTPLAN(Π)
6:         Π ← Π\{π}
7:         if NUMFLAWS(π) = 0 and COST(π) < COST(π_best) then      ▷ Incumbent update
8:             π_best ← π
9:         else if ESTIMATECOST(π) < COST(π_best) then             ▷ Bounding step
10:            f ← SELECTFLAW(π)
11:            Π ← Π ∪ REPAIRFLAW(π, f)
12:    return π_best
```

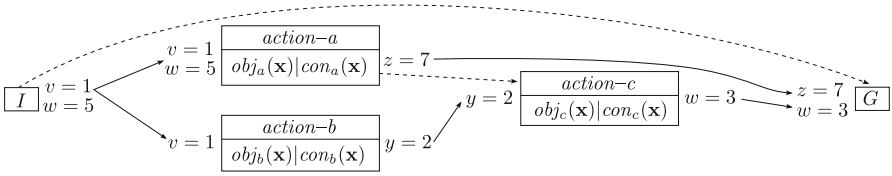

Fig. 3 An example LTOP plan

space. For example, a plan that does not satisfy a goal condition or a plan in which two actions simultaneously set the value of a state variable to different values is considered flawed. If the current plan, π, has one or more flaws, its cost is estimated on line 9 by solving a linear program and using a heuristic to provide a lower bound on the plan cost. This lower bound is compared with the current incumbent to prune plans from the search that will never lead to the optimal solution. If the plan cannot be pruned, a flaw is selected (line 10) and repaired (line 11), which consists of branching on all of the possible ways of repairing the flaw.

Figure 3 shows an example plan from LTOP. The plan consists of an initial and goal state and three actions, which are represented by boxes with preconditions on the left side and effects on the right. There are four state variables: v, w, y, z. Each action contains an optimization model consisting of an objective and constraints, which in the case of LTOP form a linear program. Solid lines represent *causal links*, which connect the effects of actions to the conditions they satisfy. Dashed lines are ordering constraints that prevent, for example, actions from invalidating the precondition required by another action. This is the case for actions a and c, which must be ordered to prevent action c from setting $w = 3$, thereby invalidating the precondition of action a, which requires $w = 5$. As this plan has no flaws, its cost can be computed using a linear programming solver with the objective $obj_a + obj_b + obj_c$, subject to the constraints $con_a \wedge con_b \wedge con_c$.

LSFRP LTOP model

We describe the LTOP model found in Tierney et al. (2012b) and refer readers to Tierney et al. (2012a) or Tierney et al. (2014) for more details about

the automated planning PDDL domain that was compared against LTOP, since LTOP greatly outperforms this approach on the LSFRP. The LTOP model consists of several core actions: `phase-out`, `phase-in`, `sail`, `sail-on-service`, `sail-with-equipment` that are based around keeping track of the *state* of each vessel being repositioned. A vessel can be in one of three states: *initial*, *transit*, or *goal*, which describe a vessel being on its initial service, performing a repositioning, or having reached its goal service, respectively.

We create phase-out actions at every port call along each vessel's initial service. Phase-out actions may only be applied for a particular vessel when it is in its initial state, meaning it has not yet begun its repositioning. Applying a phase-out action transitions a vessel to a state of transit. Phase-in actions are created at each port on the goal service in each week of the planning period. Vessels must be in a transit state to use a phase-in action. The effect of a phase-in action indicates that a vessel has reached the goal service, which satisfies the goal state of the model.

While a vessel is in transit (i.e., repositioning), it may use any sailing, SoS or sail equipment action to move between ports. Sailing actions are created between all ports in the model, except for sailings with phase-out ports as the destination, as once a vessel leaves the phase-out service it is not allowed to go back. We create SoS actions for each SoS specified by the repositioning coordinator, and equipment sailing opportunities between all ports with equipment surpluses and those with equipment demands.

Our actions describing sailing, SoS opportunities, and equipment sailings are *durative* actions, meaning they take place over a time period that is specified through the preconditions of the action. In the case of SoS actions, the amount of time they require is fixed based on the start port and end port of the action. However, sailing with and without equipment has a variable duration that must be between the minimum and maximum speed of a vessel. Phase-out and phase-in actions are instantaneous actions that have no duration.

We model sailing costs by setting the fixed cost of sailing (and sailing with equipment) actions to the maximum possible cost of sailing between two ports, i.e., sailing at maximum speed. We then subtract an amount from the action cost based on the duration of the sailing, meaning longer sailing subtract more from the cost, making them cheaper.

Hotel costs, i.e., the fixed hourly cost of operating a vessel, are modelled by associating each vessel with two optimization variables representing the begin and end time of the hotel period for that vessel. The hotel period is constrained to be the duration of the vessel between its phase-out and phase-in, minus the duration of any SoS opportunities used. In this way, no extra actions are needed to model the hotel cost, which is a key time-dependent task cost of the LSFRP. Each action that is added to an LTOP plan updates the bounds of the hotel cost, assisting LTOP in its search for the optimal solution.

All repositioning plans contain, at the very least, a phase-out and a phase-in action for each vessel. Sailings, SoS opportunities, and sail equipment actions can be ordered between the phase-out and phase-in to bring the vessel to the goal service.

Figure 4 presents an example LTOP plan for the real-world scenario in Fig. 2. Actions are shown in boxes with state variable preconditions to the left and effects to the right. The objective and constraints over optimization variables for each action

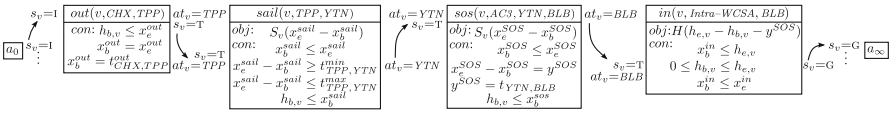

Fig. 4 An LTOP plan solving the LSFRP scenario for a single ship presented in Fig. 2

are contained inside their boxes. Causal links connecting preconditions and effects are shown with solid lines. The state variable s_v indicates whether vessel v is on its initial service (I), performing its repositioning (T), or on its goal service (G), while at_v points to the geographical location of v. Vessel v begins its repositioning by phasing out from the CHX service at TPP and sails to the port YTN where it picks up an SoS opportunity on the AC3 across the Pacific. Once arriving at BLB the vessel joins its goal service, the Intra-WCSA.

MIP model

Despite the success of LTOP in solving LSFRP instances, the question remains as to how it performs versus more traditional combinatorial optimization techniques. To find out, we create a MIP model of the LSFRP that considers the activities that a vessel may undertake and connects activities based on which ones can feasibly follow one another temporally. The structure of the LSFRP is embedded directly into the graph of the MIP, meaning that it is unable to model general automated planning problems as in Kautz and Walser (1999) and Van Den Briel et al. (2005). Note that, unlike LTOP, the MIP is capable of handling negative activity costs.

Since the vessel state in fleet repositioning is relatively simple, encompassing where a vessel is and whether it has begun its repositioning or not, there is not an exponential growth in the number of graph nodes as there would be in many planning problems if they were modeled using a graph.

Graph and MIP description

Given a graph $G = (A, T)$, where A is the set of actions (nodes), and T is the set of transitions, with $(a, b) \in T$ iff action b may follow action a, let the decision variable $y_{a,b} \in \{0, 1\}$ indicate whether the transition $(a, b) \in T$ is used or not. The auxiliary variable $w_a = \sum_{(a,b)\in T} y_{a,b}$ indicates whether action a is chosen by the model and $x_a^s, x_a^e \in \mathbb{R}^+$ are action a's start and end time, respectively. Finally, the variables h_v^s and h_v^e are the start and end time of the hotel cost period for vessel v.

Each action $a \in A$ is associated with a fixed cost, $c_a \in \mathbb{R}$, a variable (hourly) cost, $\alpha_a \in \mathbb{R}$, and a minimum and maximum action duration, $d_a^{\min}$ and $d_a^{\max}$. The set $A^t \subseteq A$ specifies actions that must begin at a specific time, t_a. The use of a particular action may exclude the use of other actions. These exclusions are specified by $\eta : A \to 2^{|A|}$. There are also n sets of mutually exclusive actions, given by $\mu : \{1, \ldots, n\} \to 2^{|A|}$. We differentiate between phase-out and phase-in actions for each vessel using the sets A_v^{po}, $A_v^{\text{pi}} \subseteq A^t$, respectively, and let $A' = A \setminus \cup_{v\in V} (A_v^{\text{po}} \cup A_v^{\text{pi}})$ be all actions that are

not related to the phase-out or phase-in. Finally, let $c_v^H \in \mathbb{R}^+$ represent each vessel's hourly hotel cost.

The upper bound on the difference between the end and start of two actions is given by $M_{a,b}^y$. The upper bound on the start of a vessel's hotel period and the lower bound on the end of the vessel's hotel period are given by M_v^s and m_v^e, respectively.

Parameters We now summarize the parameters and variables used in our MIP model for easy reference. The model uses the following parameters:

$G = (A, T)$	Graph of nodes A (actions) and arcs T (transitions)
V	Set of vessels
A_v^{po}	Set of phase-out actions for vessel $v \in V$
A_v^{pi}	Set of phase-in actions for vessel $v \in V$
A^t	Set of actions with fixed start times
t_a	Specifies the time that action $a \in A^t$
c_a	Fixed cost of action a
α_a	Variable cost of action a
c_v^H	Hotel cost of vessel $v \in V$ per hour
$d_a^{\min}, d_a^{\max}$	Minimum and maximum duration of action a
$M_{a,b}^y$	Maximum time difference between the start of actions a and b
M_v^s, m_v^e	Upper and lower bound of the start and end of vessel $v \in V$'s hotel period
$\mu(a)$	Set of actions that are mutually exclusive with a
$\eta(a)$	Set of actions that are excluded by a (but not necessarily mutually exclusive)

Variables The model uses the following variables:

$y_{a,b} \in \{0, 1\}$	Indicates whether transition $(a, b) \in T$ is used
$w_a = \sum_{(a,b)\in T} y_{a,b}$	Auxiliary variable indicating whether action a is used or not
$x_a^s, x_a^e \in \mathbb{R}^+$	Start and end time of action $a \in A$, respectively
$h_v^e, h_v^s \in \mathbb{R}^+$	Start and end of the hotel period for vessel $v \in V$, respectively

Objective and constraints The objective and constraints are as follows:

$$\min \sum_{a\in A} \left(c_a w_a + \alpha_a (x_a^e - x_a^s)\right) + \sum_{v\in V} c_v^H (h_v^e - h_v^s) \tag{22}$$

$$\text{s.t.} \sum_{\{(a,b)\in T \mid b\in A\setminus A_v^{po}\}} y_{a,b} = 1 \qquad \forall a \in \bigcup_{v\in V} A_v^{po} \tag{23}$$

$$\sum_{(a,b)\in T} y_{a,b} = \sum_{(b,c)\in T} y_{b,c} \qquad \forall b \in A' \tag{24}$$

$$\sum_{(a,b)\in T} y_{a,b} \le 1 \qquad \forall b \in A' \tag{25}$$

$$x_a^e - x_b^s \le M_{a,b}^y (1 - y_{a,b}) \qquad \forall (a, b) \in T \tag{26}$$

$$x_a^s \leq x_a^e \qquad \forall a \in A \qquad (27)$$

$$d_a^{\min} w_a \leq x_a^e - x_a^s \leq d_a^{\max} w_a \qquad \forall a \in A \qquad (28)$$

$$x_a^s = t_a w_a \qquad \forall a \in A^t \qquad (29)$$

$$h_v^s + M_v^s w_a \leq M_v^s + t_a \qquad \forall a \in \bigcup_{v \in V} A_v^{\text{po}} \qquad (30)$$

$$h_v^e + m_v^e w_a \geq m_v^e + t_a \qquad \forall a \in \bigcup_{v \in V} A_v^{\text{pi}} \qquad (31)$$

$$\sum_{a \in \mu(i)} w_a \leq 1 \qquad \text{for } i = 1, 2, \ldots n \qquad (32)$$

$$|\eta(a)| w_a + \sum_{b \in \eta(b)} w_b \leq |\eta(a)| \qquad \forall a \in A \qquad (33)$$

The objective, (22), sums the fixed and variable costs of each action that is used and adds this to the hotel cost for each vessel. The single unit flow (i.e., node disjoint) structure of the graph is enforced in constraints (23) and starts the flow of vessels through the graph, ensuring that every vessel transitions out of its phase-out. Constraints (24) are flow balance constraints that ensure vessels sail traverse path in the graph. Constraints (25) limit the incoming number of vessels to any activity to one, meaning that only a single vessel may undertake any particular activity. Constraints (26) enforce the ordering of transitions between actions, preventing the end of one action from coming after the start of another if the edge between them is turned on. Action start and end times are ordered by (27), and the duration of each action is limited by (28). Actions with fixed start times are bound to this time in (29). Constraints (30) and (31) connect the hotel start and end times to the time of the first and last action, respectively. Note that the objective forces h_v^s and h_v^e as close together as possible and that the big/little-Ms are required because each action has a different start time. The mutual exclusivity of certain sets of actions is enforced in constraints (32). Finally, constraints (33) prevent actions from being included in the plan if they are excluded by an action that was chosen. These constraints are primarily used to ensure the block phase in structure necessary to have a weekly temporal spacing of vessels. When a phase-in is chosen for a vessel, phase-ins that could not possibly be used with it are disabled. For example, in a 3-vessel problem, any phase in more than 2 weeks later from a particular phase-in is disabled if a vessel uses that phase-in.

A novel CP model

In contrast to our MIP model for the LSFRP, which is based on an activity graph, our CP model exploits the fact that SoS and SE activities cannot be chained together. The CP model uses a number of variables to store the state of each vessel in each stage of its repositioning and connects these stages through logical constraints.

Let O_v be the set of possible phase-out actions for the vessel v and let P 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 $t^{\text{out}}(v, o)$ specifies the phase-out time for that action. Similarly, $t^{\text{in}}(p, w)$ 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 o (for example, if action o starts too late for vessel v to reach port p on 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 $C^{\text{to}}(v, s, o)$ specifies the cost of vessel v using SoS action s, phasing out at phase-out o going to the SoS action, and $C^{\text{from}}(v, s, p, w)$ 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 $C^{\text{to}}(v, e, o)$, $C^{\text{dur}}(v, e)$ and $C^{\text{from}}(v, e, p, w)$ 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 $\Delta_{\min}^{\text{to}}(v, e, o)$, $\Delta_{\min}^{\text{dur}}(v, e)$ and $\Delta_{\min}^{\text{from}}(v, e, p, w)$ be the minimum sailing time of v before, during, and after the SE opportunity and $\Delta_{\max}^{\text{to}}(v, e, o)$, $\Delta_{\max}^{\text{dur}}(v, e)$ and $\Delta_{\max}^{\text{from}}(v, e, p, w)$ 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.

We summarize the model parameters, functions, and decision variables in the following tables:

Parameters and functions

V	Set of vessels
W	Set of weeks in the planning horizon
O_v	Set of phase-out actions for vessel $v \in V$
P	Set of phase-out ports for all vessels
S	Set of SoS actions
S'	Set of SoS opportunities (each of which consists of multiple SoS actions)
Q	Set of SE opportunities
$t^{\text{out}}(v, o) \in \mathbb{R}^+$	Phase-out time for port $v \in V$ in week $o \in O_v$
$t^{\text{in}}(p, w) \in \mathbb{R}^+$	Phase-in time for port $p \in P$ in week $w \in W$
$y(s) \in S'$	Specifies the SoS action that an SoS opportunity, $s \in S$, belongs to
$\Delta_{\{\min,\max\}}^{\text{to}}(v, e, o)$	Minimum and maximum sailing time for vessel $v \in V$ from phase-out $o \in O_v$ to SE $e \in Q$, respectively
$\Delta_{\{\min,\max\}}^{\text{dur}}(v, e)$	Minimum and maximum duration that vessel $v \in V$ may sail on SE $e \in Q$, respectively
$\Delta_{\{\min,\max\}}^{\text{from}}(v, e, p, w)$	Minimum and maximum duration for vessel $v \in V$ to sail from SE $e \in Q$ to phase in at port $p \in P$ in week $w \in W$, respectively
$C(v, o, p, w) \in \mathbb{R}^+$	Cost for vessel $v \in V$ to phase out with action $o \in O_v$ and phase in at port $p \in P$ in week $w \in W$; –1 if the phase-in is not possible for this vessel
$C_H(v) \in \mathbb{R}^+$	Hourly hotel cost for vessel $v \in V$
$C^{\text{to}}(v, s, o) \in \mathbb{R}^+$	Cost for vessel $v \in V$ to phase out with action $o \in O_v$ and sail to the SoS action $s \in S$
$C^{\text{from}}(v, s, p, w)$	Cost for vessel $v \in V$ sail from SoS action $s \in S$ to port $p \in P$ in week $w \in W$ to phase in
$C^{\text{to}}(v, e, o) \in \mathbb{R}^+$	Fixed cost for vessel $v \in V$ to sail from phase-out $o \in O_v$ to SE $e \in Q$
$C^{\text{dur}}(v, e) \in \mathbb{R}^+$	Fixed cost for vessel $v \in V$ to use SE $e \in Q$
$C^{\text{from}}(v, e, p, w) \in \mathbb{R}^+$	Fixed cost for vessel $v \in V$ to phase in at port $p \in P$ in week $w \in W$ from SE $e \in Q$
$\alpha_v \in \mathbb{R}$	Sailing cost coefficient for vessel $v \in V$

Variables We split the variables of the CP model into two categories: decision variables and dependent variables. The dependent variables simply keep track of costs

that are determined by the decision variables to make modelling the problem more simple.

Decision variables

$q_v \in O_v$	Phase-out activity for vessel $v \in V$
$\rho \in P$	Phase-in port
$w_v \in W$	Phase-in week for each vessel $v \in V$
$s_v \in S$	SoS action used by vessel $v \in V$, or the value 0 if no SoS action is used
$e_v \in Q$	SE opportunity used by vessel $v \in V$, or the value 0 if no SE opportunity is used
d_v^{to}, d_v^{dur} and d_v^{from}	Sailing duration of vessel $v \in V$ to the SE, during the SE, and from the SE to the phase-in, if an SE is used
r_v	Specifies one of {SoS, SE, SAIL}, indicating the type of repositioning being undertaken

Dependent variables

$c_v \in \mathbb{R}^+$	Stores the cost for vessel $v \in V$ during a direct sailing from a phase-out to a phase-in activity
$\sigma_v^{\text{to}}, \sigma_v^{\text{dur}}, \sigma_v^{\text{from}} \in \mathbb{R}^+$	Stores the costs to sail to the SoS, sail on the SoS, and sail from the SoS to the phase-in, respectively
λ_v^{to}	Stores the cost for vessel $v \in V$ to sail from its phase-out to an SE opportunity, if it uses one
λ_v^{dur}	Stores the cost for vessel $v \in V$ to sail on an SE opportunity, if it uses one
λ_v^{from}	Stores the cost for vessel $v \in V$ to sail from an SE opportunity to its phase-in, if it uses an SE opportunity

Model The CP model is formulated as follows:

$$\text{minimize} \sum_{v \in V} \Big(C_H(v) \Big(t^{\text{in}}(\rho, w_v) - t^{\text{out}}(v, q_v) \Big) + c_v + \sigma_v^{\text{from}} + \sigma_v^{\text{dur}} + \sigma_v^{\text{to}} + \lambda_v^{\text{to}} + \lambda_v^{\text{dur}} + \lambda_v^{\text{from}} + \alpha_v (d_v^{\text{to}} + d_v^{\text{dur}} + d_v^{\text{from}}) \Big) \tag{34}$$

$$\text{subject to} \quad \texttt{alldifferent}(\{w_v \mid v \in V\}) \tag{35}$$

$$\max_{v \in V} w_v - \min_{v \in V} w_v = |V| - 1 \tag{36}$$

$$\texttt{alldifferent\_except\_0}(\{s_v \mid v \in V\}) \tag{37}$$

$$\texttt{alldifferent\_except\_0}(\{y(s_v) \mid v \in V\}) \tag{38}$$

$$r_v = \texttt{SAIL} \rightarrow c_v = C(v, q_v, \rho, w_v) \qquad \forall v \in V \tag{39}$$

$$\begin{aligned} r_v = \texttt{SAIL} \rightarrow \big(& s_v = 0 \wedge y(s_v) = 0 \wedge \sigma_v^{\text{dur}} = 0 \wedge \sigma_v^{\text{from}} = 0 \wedge \sigma_v^{\text{to}} = 0 \\ & \wedge e_v = 0 \wedge \lambda_v^{\text{to}} = 0 \wedge \lambda_v^{\text{dur}} = 0 \wedge \lambda_v^{\text{from}} = 0 \big) \forall v \in V \\ r_v = \texttt{SoS} \rightarrow & s_v > 0 \wedge y(s_v) > 0 \wedge \sigma_v^{\text{dur}} = -A(v, s_v) \end{aligned} \tag{40}$$

$$\wedge\, \sigma_v^{\text{from}} = C^{\text{from}}(v, s_v, \rho, w_v) \wedge \sigma_v^{\text{to}} = C^{\text{to}}(v, s_v, q_v) \forall v \in V \tag{41}$$

$$r_v = \texttt{SoS} \rightarrow c_v = 0 \wedge e_v = 0 \wedge \lambda_v^{\text{to}} = 0 \wedge \lambda_v^{\text{dur}} = 0 \wedge \lambda_v^{\text{from}} = 0 \forall v \in V \tag{42}$$

$$s_v > 0 \vee y(s_v) > 0 \rightarrow r_v = \texttt{SoS} \forall v \in V \tag{43}$$

$$\texttt{alldifferent\_except\_0}(\{e_v \mid v \in V\}) \tag{44}$$

$$r_v = \texttt{SE} \rightarrow\ e_v > 0 \wedge \lambda_v^{\text{to}} = C^{\text{to}}(v, e_v, q_v) \wedge \lambda_v^{\text{dur}} = C^{\text{dur}}(v, e_v)$$

$$\wedge\, \lambda_v^{\text{from}} = C^{\text{from}}(v, e_v, \rho, w_v), \forall v \in V \tag{45}$$

$$r_v = \texttt{SE} \rightarrow s_v = 0 \wedge y(s_v) = 0 \wedge \sigma_v^{\text{dur}} = 0 \wedge c_v = 0$$

$$\wedge\, \sigma_v^{\text{from}} = 0 \wedge \sigma_v^{\text{to}} = 0 \forall v \in V \tag{46}$$

$$e_v > 0 \rightarrow r_v = \texttt{SE}\ \forall v \in V \tag{47}$$

$$\Delta_{\text{min}}^{\text{to}}(v, e_v, q_v) \leq d_v^{\text{to}} \leq \Delta_{\text{max}}^{\text{to}}(v, e_v, q_v) \forall v \in V \tag{48}$$

$$\Delta_{\text{min}}^{\text{dur}}(v, e_v) \leq d_v^{\text{dur}} \leq \Delta_{\text{max}}^{\text{dur}}(v, e_v) \forall v \in V \tag{49}$$

$$\Delta_{\text{min}}^{\text{from}}(v, e_v, \rho, w_v) \leq d_v^{\text{from}} \leq \Delta_{\text{max}}^{\text{from}}(v, e_v, \rho, w_v) \forall v \in V \tag{50}$$

$$\sigma_v^{\text{to}}, \sigma_v^{\text{from}}, c_v \geq 0 \forall v \in V \tag{51}$$

The objective function (34) minimises the sum of the hotel costs and repositioning action costs minus the cost savings for SoS actions for the set of vessels. Constraints (35) and (36) specify that the vessels must all phase in to the new service on different, successive weeks. Constraints (37) and (38) 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 (39) and (40) 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 (41) and (42) 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 (43) to reinforce that the repositioning type be set correctly when an SoS is chosen.

In constraints (44) we ensure that no two vessels choose the same SE action (unless they choose no SE action), and constraints (45) and (46) 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 (47) 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 (48), (49) and (50). Constraint (51) 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).

We note that our model essentially consists of a number of table constraints for each vessel bound together by the variables preventing multiple vessels from undertaking similar activities (such as phase-outs, phase-ins, etc.). We have chosen this approach instead of an even larger table constraint as the table would grow too big. This is mainly due to the effect of time on the model, as the planning period can be several months, and all of the activities in this period must be handled.

LSFRP CP results

To compare our CP model for the LSFRP against an earlier MIP model and against the LTOP planner (Tierney et al. 2012b), we use the 11 AC3 problem instances from Tierney et al. (2012b) and augment them with 27 new instances based on a real-world scenario provided by an industrial collaborator. The problem instances contain up to 9 vessels, with varying SoS and sail-with-equipment opportunities that may be used to reduce repositioning costs.

The LSFRP CP model was formulated in the MiniZinc 1.6 modelling language (Nethercote et al. 2007, 2010) and solved using the G12 finite domain solver (Wallace 2009). We compare the CP against a MIP model and the LTOP planner (Tierney et al. 2012b), both using CPLEX 12.4. 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, 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. We thereby attempt to first find an SoS option for each vessel, then search 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. This is because 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 2 compares the run times of the CP model against the MIP model and LTOP, all of which solve to optimality.[3] We ran the CP model with search annotations using a

[3] We used Intel Core i7-2600K 3.4GHz processors with a maximum of 4GB of RAM per execution.

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). We label instances with the format $G\_S\_O\_ce$, where G is the name of the goal service in Maersk Line's network, S is the number of ships, O is the number of SoS opportunities, c indicates that there are cabotage restrictions, and e indicates whether the instance contains any equipment.

CP–AOR outperforms LTOP on all instances in the dataset, and the MIP on all but one instance in the dataset. This contrasts with the results in (Kelareva et al. 2013), where CP–AOR outperformed the MIP on all AC3 instances. This is because we are using CPLEX 12.4, and Kelareva et al. (2013) uses CPLEX 12.3. This shows that using a well-supported solver like CPLEX (or the G12 CP solver) can provide advantages over less-developed techniques, like LTOP, in that new versions can provide significantly faster solution times without any model changes. The average solution time required by CP–AOR is also significantly less than LTOP and the MIP both in terms of the arithmetic and geometric means, with CP–AOR requiring, on average, only 59 % the time of the MIP, and 46 % of the time of LTOP in terms of the arithmetic mean, respectively.

We provide results for the various parameterizations of our CP approach to determine the source of its good performance. No configuration on its own is able to dominate the others across all instances. Combining annotations, ordering and redundant constraints into CP–AOR provides better average performance than any single configuration, as well as finds more optimal solutions than any other configuration. Interestingly, several of the best CPU times are not found by CP–AOR, but by individual configurations, such as on FM3_4_4ce, FM3_6_6e, FM3_6_6ce, and TP7_6_4. In fact, CP-R finds an optimal solution to TP7_6_4 not found by any other solver, indicating that an instance-specific solving approach using machine learning, like in Kadioglu et al. (2010), could be a good approach on problems like the LSFRP. We save the implementation of such a system for future work.

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.

We note that the success of this model is rather promising for other, similar problems, as no custom propagators needed to be written to solve the model. This means that concepts from this model can rather easily be copied into other CP models and extended with problem-specific side constraints. It stands to reason that good results can thus be obtained.

In light of these results, we make the following recommendations for solving problems with time-dependent task costs. First, a traditional network flow model using

Table 2 Computation times to optimality in seconds with a timeout of one hour for the CP model with and without annotations (A), repositioning type order (O), and redundant constraints (R) versus LTOP and the MIP

Problem	LTOP	MIP	CP	CP–A	CP–AO	CP–R	CP–AOR
AC3_1_0	0.6	**0.3**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
AC3_2_0	27.1	2.0	**0.1**	**0.1**	**0.1**	**0.1**	**0.2**
AC3_3_0	107.1	12.0	**0.2**	**0.2**	**0.2**	**0.2**	**0.3**
AC3_1_1e	2.1	0.9	0.3	0.3	0.3	0.4	**0.2**
AC3_2_2ce	8.1	9.3	–	48.3	12.3	–	**5.3**
AC3_3_2c	111.6	54.9	2.8	**1.3**	**1.3**	2.8	**1.4**
AC3_3_2e	112.4	69.7	657.2	6.8	7.0	1,052.5	**6.1**
AC3_3_2ce1	104.4	73.3	999.5	12.2	7.1	83.4	**6.2**
AC3_3_2ce2	99.6	68.0	78.6	15.8	9.7	505.7	**8.2**
AC3_3_2ce3	234.0	**190.1**	–	3,067.5	338.6	–	228.9
AC3_3_3	41.0	43.9	7.0	5.0	**4.6**	7.2	**4.8**
FM3_4_0	3,002.4	1,469.3	**8.2**	9.0	8.5	**8.2**	**8.1**
FM3_4_4	–	712.9	**6.5**	10.8	**6.7**	**6.5**	**6.4**
FM3_4_4c	3,031.9	380.2	**6.5**	7.0	**6.7**	**6.4**	**6.4**
FM3_4_4ce	–	514.6	–	**38.1**	68.6	3,370.0	55.9
FM3_5_0	–	615.4	23.0	27.4	23.5	23.2	**22.6**
FM3_5_4	–	897.3	**20.4**	25.8	21.1	**20.5**	**20.4**
FM3_5_4e	–	1,205.5	–	**120.4**	227.6	–	172.5
FM3_5_6	–	836.4	**21.6**	30.5	25.3	**21.7**	22.6
FM3_6_0	–	–	69.2	80.3	72.4	71.1	**68.6**
FM3_6_1	–	–	89.8	98.3	93.8	93.7	**88.7**
FM3_6_2	–	–	90.3	128.8	93.9	92.7	**89.1**
FM3_6_4	–	–	98.8	111.9	89.5	101.5	**86.1**
FM3_6_6	–	–	90.5	106.3	88.1	94.2	**83.3**
FM3_6_6e	–	–	–	**466.7**	3,063.4	–	735.5
FM3_6_6ce	–	2,290.3	–	**529.6**	3,071.6	–	736.0
TP7_6_0	–	–	468.6	472.7	467.2	469.9	**446.3**
TP7_6_4	–	–	–	–	–	**2,041.2**	–
TP7_6_4e	–	–	–	–	–	–	–
TP7_7_0	–	–	–	–	–	–	–
TP7_7_4	–	–	–	–	–	–	–
TP7_7_4e	–	–	–	–	-	–	–
TP7_8_0	–	–	–	–	–	–	–
TP7_8_6	–	–	–	–	–	–	–
TP7_8_6e	–	–	–	–	–	–	–
TP7_9_0	–	–	–	–	–	–	–
TP7_9_3	–	–	–	–	–	–	–
TP7_9_6	–	–	–	–	–	–	–
TP7_9_6e	–	–	–	–	–	–	–
Mean	2,549.5	1,996.1	1,731.8	1,246.7	1,307.9	1,683.9	1,182.3
Geo. mean	909.6	553.1	188.2	97.3	92.9	186.8	82.1

an arc-routing approach does not necessarily provide the best performance. In particular, the LP-relaxation of the model is not tight. This has a strong influence on the search, as many nodes in the branch-and-bound tree cannot be pruned. We note that time-dependent task costs do not make the relaxation require more time to solve than not having such costs. Second, while automated planning techniques show promise for making modelling tasks easier to perform, they are not yet developed enough to scale past small instances. Finally, CP is able to handle time-dependent task costs even without specialized propagators; thus we can recommend the use of CP as a first-step in modelling problems with time-dependent task costs.

Lazy clause generation

Lazy clause generation (LCG) (Ohrimenko et al. 2009) or CP with learning has been found to be effective on a number of scheduling problems (Schutt et al. 2012; Feydy and Stuckey 2009; Chu et al. 2010). LCG combines a finite domain CP solver with a SAT solver by mapping finite domain propagators to clauses in a SAT solver.

Mapping a complete CP problem to a SAT problem often results in a very large SAT problem which is intractable for even the best modern SAT solvers. LCG gets around this limitation by lazily adding clauses to the SAT solver as each finite domain propagator is executed, precisely at the point when the new clauses are able to trigger unit propagation. 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 (Ohrimenko et al. 2009).

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 scheduling and routing problems with time-varying action costs. In this section, we compare a LCG solver against a finite domain CP solver for the BPCTOP CP model from (Kelareva et al. 2012b) summarised in "BPCTOP", as well as for the LSFRP CP model presented in "LSFRP".

Experimental results for BPCTOP

We used a CP solver with LCG to solve the above CP model for the BPCTOP. The CP model was formulated in the MiniZinc modelling language (Nethercote et al. 2007) and solved using the CPX solver included in G12 2.0 (Wallace 2009; Feydy and Stuckey 2009), which uses LCG. 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 Kelareva et al. (2012a,c). 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.

The CPX solver builds on the G12 finite domain solver and combines it with a SAT solver. The FD solver controls the search, posting an *explanation* clause to the SAT solver every time a propagator is executed which updates a variable domain or causes failure. Explanation clauses explain the reason for a failure or a variable domain change; however, the implementation of a LCG solver may differ in how explanation clauses are generated, which may affect the efficiency of both the CP and

Table 3 CPU time (s) for CPX solver with LCG versus G12 finite domain solver

NShips	No tugs		With tug constraints	
	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	>1,800	**817**
11	**609**	>1,800	–	>1,800
12	>1,800	–	–	–

SAT components. After receiving a new explanation clause, the SAT solver performs unit propagation, which may cause further domain changes to be made, after which control passes back to the FD solver. See Ohrimenko et al. (2009) for an introduction to lazy clause generation, and Feydy and Stuckey (2009) for an in-depth discussion of the CPX solver.

Both solvers were used to solve problems presented in Kelareva et al. (2012a). This data set contains problems of 4 distinct types, with each set containing 17 problems ranging from 4 to 20 ships, for a fictional but realistic port. The problem types vary in terms of how tightly constrained the problems are. The ONEWAY_NARROW (ON) problems are the most tightly constrained, with all ships sailing in the same direction (outbound) with high maximum drafts, leading to narrow windows at the peak of the tide. The MIXED_WIDE (MW) problems are the least constrained, with ships being evenly split between outbound and inbound and with low maximum drafts and wide sailing windows. The MIXED_NARROW (MN) and ONEWAY_WIDE (OW) problem types are moderately constrained, with lower maximum drafts for the ONEWAY_WIDE (OW) problem set and a mix of inbound and outbound ships for the MIXED_NARROW (MN) problem set. Each problem type was also solved both with and without tug constraints, since not every port will have tugs as the bottleneck in ship scheduling. This resulted in $4 \times 17 \times 2 = 136$ different problem instances.

Table 3 presents calculation times in seconds for CPX and the G12 FD solver, for each number of ships that could be solved to optimality within the 30-min cutoff time, for the problems in the most tightly constrained (and thus most difficult) ONEWAY_NARROW (ON) problem set. A dash indicates that the optimal solution was not found within the cutoff time. Problems with 13 or more ships could not be solved within the cutoff time and are thus omitted.

Table 4 shows the number of ships in the largest problem that could be solved within the 30-min cutoff time for all four problem types, with and without tugs, with runtimes in seconds given in brackets. In both tables, bold font indicates the solver that was faster to solve a given problem.

Table 3 shows that CPX scales better than the FD solver for large ON problems with tug constraints. 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

Table 4 CPX solver versus G12 finite domain solver: number of ships in the largest problem solved within the cutoff time, with CPU time in seconds in brackets

Problem	FD	CPX
MIXED_WIDE (MW)	11 (326)	**16 (1,040)**
ONEWAY_WIDE (OW)	11 (1,480)	**11 (273)**
MIXED_NARROW (MN)	11 (370)	**16 (1,100)**
ONEWAY_NARROW (ON)	**11 (609)**	10 (482)
MIXED_WIDE_TUGS (MWT)	10 (11.5)	**13 (1,290)**
ONEWAY_WIDE_TUGS (OWT)	9 (81.5)	**11 (1,680)**
MIXED_NARROW_TUGS (MNT)	10 (202)	**12 (1,350)**
ONEWAY_NARROW_TUGS (ONT)	9 (184)	**10 (817)**

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.

Table 4 shows that the difference between the CPX and FD solvers is even greater for MIXED problems. MIXED problems have a mix of inbound and outbound ships, resulting in a less constrained problem, but one that has more complex tug constraints due to needing to consider tugs moving between inbound and outbound ships. 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 TUGS 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.

Experimental results for LSFRP

We use the Opturion CPX 1.0 optimizer (Opturion Pty Ltd 2013) to solve the LSFRP.[4] Table 5 shows the time to solve the LSFRP to optimality in seconds for all of the instances in our dataset for both using LCG (CPX–AOR) and without LCG (CP–AOR). We omit LTOP, the MIP and other CP configurations from these results due to the dominance of CP-AOR on the LSFRP. Using LCG results in an average time of 138 s less than not using LCG, although the geometric means of both approaches are relatively similar. Of particular note is that LCG is able to solve three instances that timeout when not using LCG, allowing instances with up to 9 ships to be solved. In addition to these three instances, CPX–AOR posts significant runtime gains on

[4] This is essentially the same solver as the CPX solver included with G12 that we use for the BPCTOP except for some bug fixes that allow it to work with the LSFRP.

Table 5 Computation times to optimality in seconds with a timeout of 1 h for the CP model without LCG (CP–AOR) and with LCG (CPX–AOR)

Problem	CP–AOR	CPX–AOR
AC3_1_0	**0.1**	**0.1**
AC3_2_0	**0.2**	**0.3**
AC3_3_0	**0.3**	0.6
AC3_1_1e	**0.2**	**0.2**
AC3_2_2ce	**5.3**	5.8
AC3_3_2c	1.4	**1.0**
AC3_3_2e	**6.1**	11.0
AC3_3_2ce1	**6.2**	14.5
AC3_3_2ce2	**8.2**	14.1
AC3_3_2ce3	228.9	**75.5**
AC3_3_3	4.8	**2.0**
FM3_4_0	**8.1**	13.5
FM3_4_4	**6.4**	8.1
FM3_4_4c	**6.4**	8.3
FM3_4_4ce	**55.9**	125.7
FM3_5_0	**22.6**	25.3
FM3_5_4	**20.4**	28.2
FM3_5_4e	**172.5**	507.7
FM3_5_6	**22.6**	53.9
FM3_6_0	68.6	**44.5**
FM3_6_1	88.7	**80.5**
FM3_6_2	89.1	**82.4**
FM3_6_4	**86.1**	218.6
FM3_6_6	**83.3**	84.4
FM3_6_6e	**735.5**	823.4
FM3_6_6ce	**736.0**	825.2
TP7_6_0	446.3	**63.4**
TP7_6_4	–	–
TP7_6_4e	–	–
TP7_7_0	–	**435.6**
TP7_7_4	–	–
TP7_7_4e	–	–
TP7_8_0	–	**1,303.4**
TP7_8_6	–	–
TP7_8_6e	–	–
TP7_9_0	–	**3,455.5**
TP7_9_3	–	–
TP7_9_6	–	–
TP7_9_6e	–	–
Mean	1,182.3	1,043.9
Geo. mean	82.1	83.1

4 instances, although it is significantly slower than not using LCG on 10 instances, with the rest of the instances resulting in (roughly) a tie. LCG seems to be somewhat slower on problems with equipment, which tend to have more time-dependent task costs in their objective than instances without equipment, which could indicate that LCG is useful for problems with limited numbers of time-dependent tasks or tasks that resemble the hotel costs present in all LSFRP instances.

Discussion

Lazy clause generation (LCG) is a very general technique 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, but shows promise in its scaling behavior on the LSFRP. However, like traditional CP solvers, LCG solvers can be combined with decomposition techniques or large neighbourhood search to improve scalability (Schutt et al. 2012). Given our positive results on both the BPCTOP and LSFRP, LCG may be worth investigating as an approach to dealing with other routing and scheduling problems with time-dependent task costs.

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. (2005).

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.

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 by solving the problem with the objective of maximising the number of ships scheduled to sail. Table 6 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, (i.e., the approach that results in the largest reduction in runtime).

Table 6 shows that, while the simplified model is faster for small problem sizes, for large problem sizes the CPX solver with LCG is faster than the simplified model to

Table 6 Runtime (s) of CPX versus the FD solver with a simplified BPCTOP model

Problem type	NShips (small)	Runtime (s)			NShips (large)	Runtime (s)		
		FD	CPX	SIMPLIFIED		FD	CPX	SIMPLIFIED
MW	11	326	**8.19**	188	16	>1,800	**1,040**	>1,800
OW	11	1,480	273	**0.56**	12	>1,800	>1,800	**26.7**
MN	11	370	**7.64**	180	16	>1,800	**1,100**	>1,800
ON	10	99.9	482	**0.34**	12	>1,800	>1,800	**19.6**
MWT	10	11.5	17.1	**11.4**	13	>1,800	**1,290**	>1,800
OWT	9	81.5	23.9	**4.60**	11	>1,800	**1,680**	>1,800
MNT	10	202	42.8	**8.21**	12	>1,800	**1,350**	>1,800
ONT	9	184	65.7	**0.69**	10	>1,800	817	**262**

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 4, CPX is particularly effective on large problems with tugs.

One interesting observation from Table 6 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-min 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 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 (Qureshi et al. 2009). 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.

Results for liner shipping fleet repositioning

We implemented a simplified LSFRP model by fixing all sailing and sail equipment actions to "slow-steaming", i.e., minimum fuel cost with maximum time using the LTOP planner (Tierney et al. 2012b). We perform a similar simplification for the CP model, in which all sail equipment actions are bound to their maximum length. Sail actions in the CP model are discretised, so we do not need to change them to simplify the model. We note, however, that the hotel cost calculation, an important time-dependent task cost in the LSFRP, cannot be simplified in any reasonable way.

Table 7 provides the runtimes in seconds for LTOP and CP versus simplified LTOP and CP approaches, with a timeout of one hour of CPU time.[5] The results show that

[5] Experiments used Intel Core i7-2600K 3.4GHz processors with a maximum of 4GB per execution.

Table 7 Computation times to optimality in seconds for CP and LTOP versus simplified approaches (S–LTOP and S–CP–AOR) with optimal windows. Note that we have removed all TP7 instances except for TP7_6_0 due to timeouts

Problem	LTOP	S–LTOP	CP–AOR	S–CP–AOR
AC3_1_0	0.6	0.5	**0.1**	**0.1**
AC3_2_0	27.1	20.7	**0.2**	**0.1**
AC3_3_0	107.1	66.1	**0.3**	**0.3**
AC3_1_1e	2.1	2.3	**0.2**	**0.2**
AC3_2_2ce	8.1	8.9	5.3	**2.5**
AC3_3_2c	111.6	142.7	**1.4**	**1.3**
AC3_3_2e	112.4	117.4	6.1	**3.1**
AC3_3_2ce1	104.4	115.7	6.2	**3.2**
AC3_3_2ce2	99.6	104.4	8.2	**3.5**
AC3_3_2ce3	234.0	234.0	228.9	**26.9**
AC3_3_3	41.0	42.8	**4.8**	**4.5**
FM3_4_0	3,002.4	2,931.3	**8.1**	8.8
FM3_4_4	–	–	**6.4**	7.2
FM3_4_4c	3,031.9	2,939.6	**6.4**	6.9
FM3_4_4ce	–	–	55.9	**8.6**
FM3_5_0	–	–	**22.6**	24.7
FM3_5_4	–	–	**20.4**	22.0
FM3_5_4e	–	–	172.5	**24.3**
FM3_5_6	–	–	**22.6**	24.7
FM3_6_0	–	–	**68.6**	73.5
FM3_6_1	–	–	**88.7**	97.2
FM3_6_2	–	–	**89.1**	96.5
FM3_6_4	–	–	**86.1**	92.9
FM3_6_6	–	–	**83.3**	91.8
FM3_6_6e	–	–	735.5	**78.0**
FM3_6_6ce	–	–	736.0	**78.3**
TP7_6_0	–	–	446.3	–
Mean	2,549.5	2,572.5	1,182.3	1,220.0
Geo. mean	909.6	934.5	82.1	61.6

solve-and-improve is not a particularly effective method within the LTOP framework, with only AC3_3_0 showing any speed improvements. We can, therefore, conclude that fixing the length and cost of the sail action is not very effective for the LTOP method on 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.

Using solve-and-improve with CP–AOR leads to more promising results than with LTOP, although it is unable to offer solutions for large LSFRP instances with equipment. S–CP–AOR is able to find a solution quickly on a number of problems with equipment, which are the only problems for CP where it can make a difference. In particular on AC3_3_2ce3, FM3_6_6e, and FM3_6_6ce solve-and-improve requires only around 10 % of the time of CP–AOR. Of course, the improve step still would need to be run.

Discussion

A solve-and-improve approach that simplifies away time-varying action costs was found by Lin et al. (2005) 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 CP model or a solver with LCG 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 Lin et al. (2005), 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 Lin et al. (2005).

Conversion to vehicle routing

We solve the BPCTOP without tug constraints from "Conversion to vehicle routing" using the Indigo VRP solver, which is able to consider a wide variety of side constraints (Kilby and Verden 2011). One limitation of the solver is that it is only able to handle soft time windows with linear penalties, whereas the draft function can vary non-linearly outside the peak draft windows. Using linear penalty functions to approximate our draft functions may lead to schedules that are not entirely optimal with respect to the true draft functions. While conversion to VRP may be useful for finding close-to-optimal solutions for large ship scheduling problems, a further improvement step may be needed to optimise the final schedules. For example, we could use the VRP for an approximate solution for large multi-port problems or long time scales and then find optimal schedules for each high tide at each port by solving the CP model with the exact draft function.

The basic port optimisation problem without tug constraints only has one vehicle and can, therefore, be considered as a travelling salesman problem which aims to minimise soft time window penalties instead of the total distance. However, a TSP approach would not be generalisable to more complex scheduling problems that have multiple resources. Also, any TSP approach would not be able to handle more complex constraints such as the BPCTOP tug constraints. In this paper, we only investigate modelling the problem as a VRP, not as a TSP, since the VRP approach would be generalisable to a larger range of problems.

We converted the most tightly constrained ONEWAY_NARROW ship scheduling problems from Kelareva et al. (2012a) with 4–20 ships sailing on a tide to VRP problems with soft time windows and solved them using the Indigo VRP solver (Kilby and Verden 2011). Indigo does not yet support constraints that require two jobs to be completed at the same time in conjunction with early and late arrival penalties, so we only present results for the problems with no tug constraints for now. Extending

Table 8 Indigo VRP solver versus G12 FD and CPX solvers, without tug constraints, for ONEWAY_NARROW problems

NShips	G12_FD	CPX	VRP: 200	VRP: 1K
4	0.34	**0.23**	0.55	2.18
5	**0.23**	**0.22**	1.14	5.15
6	0.44	**0.22**	2.43	11.7
7	**0.34**	**0.33**	4.70	23.0
8	**0.45**	0.87	8.43	41.6
9	**4.70**	23.4	13.8	68.6
10	99.9	482	**21.3**	108
11	609	>1,800	**30.7**	151
12	>1,800	–	**41.7**	207
13	–	–	**53.6**	276
14	–	–	**65.3**	330
15	–	–	**78.4**	430
16	–	–	**93.5**	480
17	–	–	**118**	548
18	–	–	**130**	668
19	–	–	**151**	737
20	–	–	**170**	851

Indigo to include the constraints required to support tugs and testing the VRP model with tug constraints is left for future work.

All experiments were run on a Windows 7 machine with an Intel i7-930 quad-core 2.80 GHz processor and 12.0 GB RAM, and with a 30-min (1,800-s) cutoff time. Calculation times for Indigo versus the G12 FD and CPX solvers are shown in Table 8.

The Indigo calculations used a large neighbourhood search with two variations of the search parameters: starting from 1,000 iterations and increasing the number of iterations by 200 for every ship; or starting from 200 iterations and increasing the number of iterations by 40 for every ship. The calculation times for both sets of iterations are shown in Table 8. Up to 15 visits are removed in each LNS iteration and then re-inserted.

Indigo uses large neighbourhood search to find the best solution within the given number of iterations, but it does not produce a proof of optimality for the VRP solution, and may return suboptimal solutions. Also, since Indigo only supports linear time-window penalty functions, it cannot model the non-linear task value functions precisely, which contributes to it finding suboptimal solutions for larger problems where not all ships are able to sail with their peak draft.

Figure 5 shows the cost increase of the optimal and VRP solutions compared to an "ideal" situation with all ships sailing with their peak draft, for problems with up to 11 ships for which an optimal solution was found using the CP model. Since the solver's large neighbourhood search uses randomisation, the solver was run five times for each problem, and the best of the five results is shown in Fig. 5. Note: the calculation times in Table 8 show the mean runtime for the five calculations. The solver failed to find an optimal solution for problems with 10 or more ships, as well as for 8 ships with the smaller number iterations of Large Neighbourhood Search.

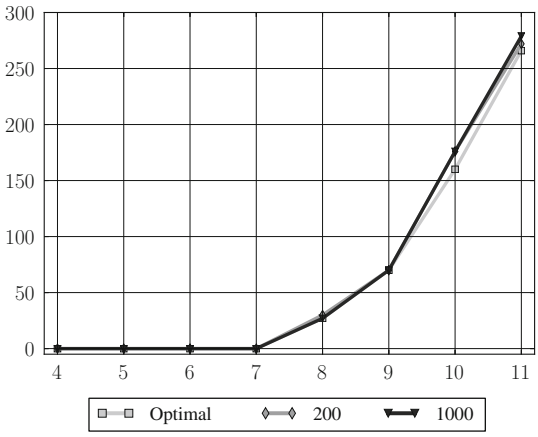

Fig. 5 Cost increase of optimal and VRP solutions compared to all ships sailing with peak draft

Tug constraints Modelling the tug constraints as a VRP requires a VRP solver that can handle constraints that require two jobs to be completed at the same time, and constraints that specify that two jobs must either both be completed, or both not completed. Unfortunately, at present, Indigo cannot calculate soft time window penalties correctly with these side constraints present, and other existing VRP solvers do not include these types of constraints at all to the authors' knowledge. For the moment, the approach of converting a scheduling problem to a VRP is limited only to simple scheduling problems without complex side constraints. More flexible VRP solvers may be developed in future.

Summary

The VRP modelling of the BPCTOP can be solved much faster than the CP model for large problems with 10 or more ships. Indigo is able to solve problems with up to 20 ships in a short timeframe—8 ships more than the largest problem solved with any CP solver. However, the results are suboptimal for all problems with 10 or more ships, and no certificate of optimality can be provided by Indigo for any instance.

Increasing the number of iterations does not significantly improve the quality of the schedules—in some cases, the best schedule found over five calculations is better for calculations with fewer search iterations, for example, the case with 11 ships. This indicates that the solver we chose for our experiments may be prone to getting stuck in local minima, since past a certain point, searching with restarts is more effective than simply increasing the number of iterations of large neighbourhood search.

Modelling a scheduling problem as a VRP may be an effective way to find close to optimal solutions for larger scheduling problems than can be solved using optimal methods. However, since VRP solvers only support the types of constraints and penalty functions found in VRP problems, many types of scheduling problems cannot be modelled accurately as a VRP, and a VRP solver will not be able to find optimal solutions to the scheduling problem. However, even for those scheduling problems where the

VRP solution is not good enough on its own, it may provide a good starting point for a slower method that can consider complex constraints and objective functions.

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 reviewed scheduling and routing problems with time-varying action costs, which increase the complexity of a problem, across a number of applications, including three different ship scheduling problems, the vehicle routing problem with soft time windows, project scheduling with net present value, and satellite imaging scheduling. We then applied several approaches that had been successfully used to solve other problems with time-varying action costs to the liner shipping fleet repositioning problem (LSFRP) and the bulk port cargo throughput optimisation problem (BPCTOP).

We presented a novel CP model for the LSFRP and compared it against existing MIP and automated 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 BPCTOP in an earlier paper (Kelareva et al. 2012a). 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. On the LSFRP, LCG solves 3 more instances than not using LCG and provides a faster average solution time across the entire dataset.

Since time-varying action costs make a problem more difficult to solve, one approach previously used for these problems has been solve-and-improve, which uses simplified models without time-varying action costs to find initial solutions (Lin et al. 2005). We compared the LTOP and CP models 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.

Overall, we are able to recommend CP techniques as a worthwhile modelling approach for time-dependent task costs. We find the results of the CP models on the BPCTOP, LSFRP and in the literature promising, as CP tends to perform well on such task costs even without having custom propagators that support them. This eases problem modelling and makes CP a good testbed for a first approach for solving problems with time-dependent task costs. While MIP approaches do not perform poorly on such problems, we see that time-dependent task costs are rather detrimental to the LP-relaxation. This results in large search trees in comparison to CP, which is able to achieve stronger bounding. It is possible that with the addition of a strong cutting mechanism for such costs, MIP approaches could achieve better performance.

We also investigated the approach of converting a scheduling problem with time-varying action costs into a VRPSTW to take advantage of the highly efficient specialised solvers available for the extensively researched VRPSTW. We modelled the BPCTOP as a VRPSTW and solved it using the existing Indigo VRP solver. Modelling scheduling problems as a VRP can allow solutions to be found quickly for large problems; however, if the scheduling problem contains complex cost functions or side constraints, VRP solvers may not be able to model them accurately, leading to a reduction in solution quality. VRP solvers may be a fast way to solve simple scheduling problems or to quickly find an approximate initial solution to use as input to a more specialised search algorithm.

For future work, we intend to investigate the integration of time-dependent task costs into other optimization problems to increase their realism and thereby their relevance to the industry. For example, including time-dependent draft restrictions in problems such as the liner shipping network design problem (see, e.g., Brouer et al. 2014) or adding detailed traffic congestion to problems like inter-terminal transportation delay reduction at container terminals (Tierney et al. 2014). A number of vehicle routing applications could benefit from a view of time-dependent task costs as in Malandraki and Daskin (1992) or the pollution routing problem (Bektaş and Laporte 2011; Franceschetti et al. 2012). Although adding time-dependent task costs to these problems would likely increase the solving difficulty, the benefits of more closely matching real-world processes is worth the trade-off.

Acknowledgments 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. They 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

Achterberg T (2009) SCIP: solving constraint integer programs. Math Progr Comput 1(1):1–41

Agnetis A, de Pascale G, Detti P, Vicino A (2013) Load scheduling for household energy consumption optimization. IEEE Trans Smart Grid 4(4):2364–2373. doi:10.1109/TSG.2013.2254506http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6509468

Bausch D, Brown G, Ronen D (1998) Scheduling short-term marine transport of bulk products. Marit Policy Manag 25(4):335–348

Bektaş T, Laporte G (2011) The pollution-routing problem. Transp Res Part B Methodol 45(8):1232–1250
Brouer B, Alvarez J, Plum C, Pisinger D, Sigurd M (2014) A base integer programming model and benchmark suite for liner shipping network design. Transp Sci 48(2):281–312
Brown G, Graves G, Ronen D (1987) Scheduling ocean transportation of crude oil. Manag Sci 33(3):335–346
Christiansen M, Fagerholt K (2002) Robust ship scheduling with multiple time windows. Naval Res Logist 49(6):611–625
Christiansen M, Fagerholt K, Nygreen B, Ronen D (2007) Chapter 4: Maritime transportation. In: Barnhart C, Laporte G (eds) Transportation, handbooks in operations research and management science, Elsevier, vol 14, pp 189–284
Christiansen M, Fagerholt K, Nygreen B, Ronen D (2013) Ship routing and scheduling in the new millennium. Eur J Oper Res 228(3):467–483
Christiansen M, Fagerholt K, Ronen D (2004) Ship routing and scheduling: status and perspectives. Transp Sci 38(1):1–18
Chu G, de la Banda M, Mears C, Stuckey P (2010) Symmetries and lazy clause generation. In: Proceedings of the 16th international conference on principles and practice of constraint programming (CP'10) Doctoral programme, pp 43–48 (2010)
Coles AJ, Coles AI, Fox M, Long D (2010) Forward-chaining partial-order planning. In: Proceedings of the twentieth international conference on automated planning and scheduling (ICAPS-10)
Fagerholt K (2000) Evaluating the trade-off between the level of customer service and transportation costs in a ship scheduling problem. Marit Policy Manag 27(2):145–153
Fagerholt K (2001) Ship scheduling with soft time windows: an optimisation based approach. Eur J Oper Res 131(3):559–571
Fagerholt K (2004) A computer-based decision support system for vessel fleet scheduling: experience and future research. Decis Support Syst 37(1):35–47
Fagerholt K, Laporte G, Norstad I (2010) Reducing fuel emissions by optimizing speed on shipping routes. J Oper Res Soc 61:523–529
Fagerholt K, Laporte G, Norstad I (2010) Reducing fuel emissions by optimizing speed on shipping routes. J Oper Res Soc 61:523–529
Feydy T, Stuckey P (2009) Lazy clause generation reengineered. In: Proceedings of the 15th international conference on principles and practice of constraint programming (CP'09), LNCS, vol 5732, pp 352–366
Figliozzi M (2010) An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. Transp Res Part C Emerg Technol 18(5):668–679
Figliozzi M (2012) The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. Transp Res Part E Logist Transp Rev 48(3):616–636
Fikes R, Nilsson N (1971) STRIPS: A new approach to the application of theorem proving to problem solving. Artif Intell 2(3–4):189–208
Fox M, Long D (2003) PDDL2.1: an extension to PDDL for expressing temporal planning domains. J Artif Intell Res 20(1):61–124
Franceschetti A, Van Woensel T, Honhon D, Bektaş T, Laporte G (2012) The timedependent pollution routing problem. Industrial engineering and innovation Sciences, Technology University of Eindhoven, Eindhoven, technical report
Grinold R (1972) The payment scheduling problem. Naval Logist Res Q 19(1):123–136
Kadioglu S, Malitsky Y, Sellmann M, Tierney K (2010) ISAC: Instance-specific algorithm configuration. In: Coelho H, Studer R, Wooldridge M (eds) Proceedings of the 19th European conference on artificial intelligence (ECAI-10), Frontiers in Intelligence and Applications, vol 215, pp 751–756
Kautz H, Walser J (1999) State-space planning by integer optimization. In: Proceedings of the national conference on artificial intelligence, pp 526–533
Kelareva E, Brand S, Kilby P, Thiébaux S, Wallace M (2012) 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), pp 110–118
Kelareva E, Kilby P, Thiébaux S, Wallace M (2012) Ship scheduling with time-varying draft. In: 5th international workshop on freight transportation and logistics (ODYSSEUS'12)
Kelareva E, Kilby P, Thiébaux S, Wallace M (2012) Ship scheduling with time-varying draft: constraint programming and benders decomposition. Transportation Science (submitted)

Kelareva E, Tierney K, Kilby P (2013) CP Methods for scheduling and routing with time-dependent task costs. In: Gomes C, Sellmann M (eds) Integration of AI and OR techniques in constraint programming for combinatorial optimization problems, Lecture notes in computer science, vol 7874. Springer, Berlin, pp 111–127
Kilby P, Verden A (2011) Flexible routing combing constraint programming, large neighbourhood search, and feature-based insertion. In: Schill K, Scholz-Reiter B, Frommberger L (eds) Proceedings of 2nd workshop on artificial intelligence and logistics (AILOG'11), pp 43–49
Lemaître M, Verfaillie G, Jouhaud F, Lachiver JM, Bataille N (2002) Selecting and scheduling observations of agile satellites. Aerosp Sci Technol 6:367–381
Lin W, Liao D, Liu C, Lee Y (2005) Daily imaging scheduling of an earth observation satellite. Syst Man Cybern Part A Syst Hum IEEE Trans 35(2):213–223
Lougee-Heimer R (2003) The common optimization interface for operations research: promoting open-source software in the operations research community. IBM J Res Dev 47(1):57–66
Malandraki C, Daskin MS (1992) Time dependent vehicle routing problems: formulations, properties and heuristic algorithms. Transp Sci 26(3):185
Nau D, Ghallab M, Traverso P (2004) Automated planning: theory and practice. Morgan Kaufmann Publishers Inc. San Francisco, CA
Nethercote N, Marriott K, Rafeh R, Wallace M, de la Banda M (2010) Specification of zinc and minizinc. http://www.g12.cs.mu.oz.au/minizinc/downloads/doc-1.2/zinc-spec.pdf
Nethercote N, Stuckey P, Becket R, Brand S, Duck G, Tack G (2007) MiniZinc: Towards a standard CP modelling language. In: Bessière C (ed) Principles and practice of constraint programming (CP'07), LNCS, vol 4741. Springer, pp 529–543
Norstad I, Fagerholt K, Laporte G (2011) Tramp ship routing and scheduling with speed optimization. Transp Res 19:853–865
O'Brien T (2002) Experience using dynamic underkeel clearance systems. In: Proceedings of the PIANC 30th international navigational congress, pp 1793–1804
Ohrimenko O, Stuckey P, Codish M (2009) Propagation via lazy clause generation. Constraints 14(3):357–391
OMC International: DUKC helps Port Hedland set ship loading record (2009). http://www.omc-international.com/images/stories/press/omc-20090810-news-in-wa.pdf
Opturion Pty Ltd: Opturion CPX User's Guide, Version 1.0 (2013)
Penberthy J, Weld D (1992) UCPOP: a sound, complete, partial order planner for ADL. In: Proceedings of the 3rd international conference on knowledge representation and reasoning
Port Hedland Port Authority: 2009/10 cargo statistics and port information (2011). http://www.phpa.com.au/docs/CargoStatisticsReport.pdf
Qureshi A, Taniguchi E, Yamada T (2009) An exact solution approach for vehicle routing and scheduling problems with soft time windows. Transp Res Part E Logist Transp Rev 45(6):960–977
Rakke J, Christiansen M, Fagerholt K, Laporte G (2011) The traveling salesman problem with draft limits. Comput Oper Res 39:2161–2167
Rao L, Liu X, Xie L, Liu W (2010) Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment. In: Proceedings of INFOCOM, IEEE, pp 1145–1153
Rossi F, Van Beek P, Walsh T (2006) Handbook of constraint programming. Elsevier Science, Amsterdam
Russell A (1970) Cash flows in networks. Manag Sci 16(5):357–373
Schutt A, Chu G, Stuckey P, Wallace M (2012) Maximising the net present value for resource-constrained project scheduling. In: Proceedings of CPAIOR 2006, LNCS, vol 7298. Springer, pp 362–378
Sexton T, Choi Y (1985) Pickup and delivery of partial loads with soft time windows. Am J Math Manag Sci 6:369–398
Shaw P (1997) A new local search algorithm providing high quality solutions to vehicle routing problems. Working paper, University of Strathclyde, Glasgow, Scotland
Smith S (2005) Is scheduling a solved problem? In: Proceedings of Multidisciplinary scheduling: theory and applications, pp 3–17
Song JH, Furman K (2010) A maritime inventory routing problem: practical approach. Comput Oper Res 40(3):657–665
Tierney K (2013) Optimizing liner shipping fleet repositioning plans. PhD thesis, IT University of Copenhagen
Tierney K, Áskelsdóttir B, Jensen R, Pisinger D (2014) Solving the liner shipping fleet repositioning problem with cargo flows. To appear in transportation science

Tierney K, Coles A, Coles A, Jensen R (2012) A PDDL domain of the liner shipping fleet repositioning problem. Technical report TR-2012-152, IT University of Copenhagen

Tierney K, Coles A, Coles A, Kroer C, Britt A, Jensen R (2012) Automated planning for liner shipping fleet repositioning. In: Proceedings of the 22nd international conference on automated planning and scheduling (ICAPS'12), pp 279–287

Tierney K, Jensen R (2011) Liner shipping fleet repositioning. In: Proceedings of international conference on computational logistics (ICCL'11), Abstract

Tierney K, Jensen R (2012) The liner shipping fleet repositioning problem with cargo flows. In: Hu H, Shi X, Stahlbock R, Voß S (eds) Computational logistics, LNCS, vol 7555. Springer, pp 1–16

Tierney K, Jensen RMJ (2013) A node flow model for the inflexible visitation liner shipping fleet repositioning problem with cargo flows. In: Pacino D, Voß S, Jensen RM (eds) Computational logistics, Lecture notes in computer science, vol 8197. Springer, pp 18–34

Tierney K, Voß S, Stahlbock R (2014) A mathematical model of inter-terminal transportation. Eur J Oper Res 235(2):448–460. doi:10.1016/j.ejor.2013.07.007 http://www.sciencedirect.com/science/article/pii/S0377221713005778 (Maritime logistics)

University of Melbourne: MiniZinc challenge (2011). http://www.g12.cs.mu.oz.au/minizinc/challenge2011/challenge.html

Van Den Briel M, Vossen T, Kambhampati S (2005) Reviving integer programming approaches for AI planning: a branch-and-cut framework. In: Proceedings of the 15th international conference on automated planning and scheduling (ICAPS-05), pp 310–319

Vanhoucke M, Demeulemeester E, Herroelen W (1999) On maximising the net present value of a project under resource constraints. In: Proceedings of Research report 9915, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium

Vanhoucke M, Demeulemeester E, Herroelen W (2001) On maximizing the net present value of a project under renewable resource constraints. Manag Sci 47:1113–1121

Wallace M (2009) G12: towards the separation of problem modelling and problem solving. In: Proceedings of CPAIOR 2009, LNCS, vol. 5547. Springer, pp 8–10

Wang J, Jing N, Li J, Chen H (2007) A multi-objective imaging scheduling approach for earth observing satellites. In: Proceedings of the 9th annual conference on genetic and evolutionary computation (GECCO'07), pp 2211–2218

Wolfe W, Sorensen S (2000) Three scheduling algorithms applied to the earth observing systems domain. Manag Sci 46(1):148–168

Yao F, Li J, Bai B, He R (2010) Earth observation satellites scheduling based on decomposition optimization algorithm. Int J Image Gr Signal Process 1:10–18