Automated Planning for Liner Shipping Fleet Repositioning

Kevin Tierney
IT University of Copenhagen
Copenhagen, Denmark
kevt@itu.dk

Amanda Coles and **Andrew Coles**
King's College London
London, UK
{amanda.coles,andrew.coles}@kcl.ac.uk

Christian Kroer and **Adam M. Britt** and **Rune Møller Jensen**
IT University of Copenhagen
Copenhagen, Denmark
{ckro,ambr,rmj}@itu.dk

Abstract

The Liner Shipping Fleet Repositioning Problem (LSFRP) poses a large financial burden on liner shipping firms. During repositioning, vessels are moved between services in a liner shipping network. The LSFRP is characterized by chains of interacting activities, many of which have costs that are a function of their duration; for example, sailing slowly between two ports is cheaper than sailing quickly. Despite its great industrial importance, the LSFRP has received little attention in the literature. We show how the LSFRP can be solved sub-optimally using the planner POPF and optimally with a mixed-integer program (MIP) and a novel method called Temporal Optimization Planning (TOP). We evaluate the performance of each of these techniques on a dataset of real-world instances from our industrial collaborator, and show that automated planning scales to the size of problems faced by industry.

Introduction

Situated at the heart of global trade, liner shipping networks transported over 1.3 billion tons of cargo on over 9,600 container vessels in 2011 (UNCTAD 2011). Vessels are regularly repositioned between services in liner shipping networks to adjust the networks to the world economy and stay competitive. Since repositioning a single vessel can cost hundreds of thousands of US dollars, optimizing the repositioning activities of vessels is an important problem to the liner shipping industry.

The Liner Shipping Fleet Repositioning Problem (LSFRP) consists of finding minimal cost sequences of activities that move vessels from one service to another within a liner shipping network. Fleet repositioning involves sailing and loading activities subject to complex handling and timing restrictions. As is the case for many industrial problems, the objective is cost minimization (including costs for CO_2 emissions and pollution), and it is important that all cost elements, including those that are only loosely coupled with activity choices, can be accurately modeled.

In this paper, we consider three methods for solving the LSFRP. First, we describe an automated planning model of the LSFRP that is available as a PDDL domain, and show that it can be solved using the planner POPF (Coles et al. 2011) by extending its TIL handling capabilities. However, since POPF is not an optimal planner, and no automated planner that is capable of solving the LSFRP to optimality exists, we propose a novel framework called Temporal Optimization Planning (TOP). TOP uses durative planning to build optimization models. In contrast to advanced temporal planning languages, e.g. (Fox and Long 2006), TOP does not enforce a strong semantic relation between planning actions and optimization components, since this enables the formulation of richer optimization models. TOP solves problems using an optimization version of partial-order planning (Penberthy and Weld 1992), based on a branch-and-bound algorithm. We define a general lower bound for partial plans in the naturally occurring case where the minimum costs of optimization components are non-negative. We show that this bound can be improved by an extension of the h_{max} heuristic (Haslum and Geffner 2000) that makes it possible to estimate the cost of required actions not currently in the plan.

We also model and solve the LSFRP with a mixed integer program (MIP) using a graph based model. And, finally, we present experimental results comparing solving time and solution quality for all three approaches on a number of problem instances based on a scenario from our industrial collaborator. Our results show that automated planning is capable of solving real-world scenarios of the LSFRP within the time required to create a usable decision support system.

Liner Shipping Fleet Repositioning

Container vessels are routinely *repositioned*, i.e. moved from one service to another, in order to better orient a liner shipping network to the economy. A liner shipping network consists of a set of circular routes, called services, that visit ports on a regular, usually weekly, schedule. Shipping lines regularly add and remove services from their networks in order to stay competitive, requiring vessel repositionings. The repositioning of vessels is expensive due to the cost of fuel (in the region of hundreds of thousands of dollars) and the revenue lost when a ship is not on a service carrying customers' cargo. Given that liner shippers around the world reposition hundreds of vessels per year, optimizing vessel movements can significantly reduce the economic and environmental burdens of containerized shipping.

Given a set of vessels, where each vessel is assigned an initial service and a goal service, the aim of the LSFRP is to reposition each vessel to its goal service within a given time period at minimal cost. Each vessel begins its repositioning when it *phases out* from its current service, meaning it ceases regular operations on the service. Vessels may phase out of any port on the service they are sailing on at the time the port is normally called by the service. After a vessel has phased out, it may undertake activities that are not part of its normal operations until it *phases in* at its goal service, which, like phasing out, must happen at a goal service port at the time the goal service is scheduled to call it. Throughout the time between the phase-out and the phase-in, except where noted, the repositioning vessel pays a fixed hourly cost, referred to as the *hotel cost* in shipping parlance.

Between the phase-out and the phase-in, a vessel may undertake the following activities. First, vessels may *sail* directly between two ports, incurring a cost that actually declines as the duration of the sailing increases, due to the fuel efficiencies of engines at low speeds. Second, a vessel may also sail with *equipment*, e.g. empty containers, from ports where they are in excess to ports where they are in demand, earning a profit per TEU[1] carried, but incurring a delay to load and unload the equipment. And third, a vessel may perform a *sail-on-service* (SOS), in which the vessel replaces a vessel on an already running service.

SOS opportunities are desirable because the repositioning vessel incurs no hotel or fuel costs on an SOS, but cargo may need to be *transshipped* from the replaced vessel to the repositioning vessel, depending on where the repositioning vessel starts the SOS. Cargo transshipments are subject to a fee per TEU transshipped and vessels are delayed depending on how much cargo must be transferred. An SOS may only start at certain ports due to *cabotage* restrictions, which are laws that prevent foreign vessels from offering domestic cargo services. Note that while we do not take a detailed view of cargo flows, the activities we allow a vessel to undertake are chosen such that they do not significantly disrupt the network's cargo flows.

One of the key difficulties in the LSFRP lies in the constraints that dictate how vessels may phase in to a new service. It is essential that the liner shipping nature of the service is enforced, meaning that once a vessel visits a port on the goal service, there must be a vessel visiting that port in every subsequent week within the planning horizon. This constraint is a business requirement, as once a service is started, customers expect to be able to ship their cargo without interuption. This constraint, however, leads to $v!$ different orderings at each port on the goal service, where v is the number of vessels being repositioned. Thus, each ordering at each port is potentially associated with a different cost.

We performed a case study with our industrial partner to better understand the nature of fleet repositioning problems. A new service in the network, the "Intra-WCSA", required three vessels[2] that were sailing on services in Asia. Repositioning coordinators were tasked with moving the vessels to the Intra-WCSA at as low a cost as possible.

[1]TEU stands for *twenty-foot equivalent unit* and represents a single twenty-foot intermodal container.

[2]For reasons of confidentiality, some details of the case study have been changed.

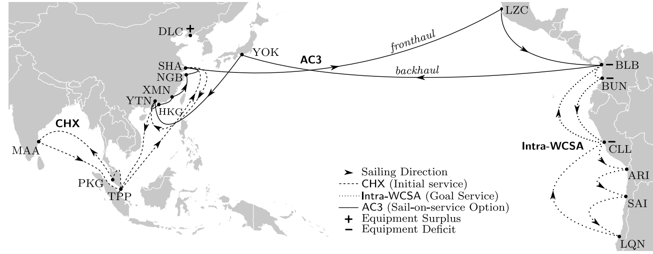

Figure 1: A subset of the case study we performed with our industrial collaborator is shown. A vessel on the CHX service must be repositioned to the new Intra-WCSA service.

Figure 1 shows a subset of the case study and the cost saving opportunities that repositioning coordinators had available to them. The Intra-WCSA required three vessels, one of which was on the CHX service. Two further vessels were on services that are not shown in the figure, but were also in southeast Asia. Vessels could carry equipment from northern China to South America, as well as utilize the AC3 service as a sail-on-service opportunity. The problem was solved by hand, as no automated tools existed to assist in solving the LSFRP. The solution sent all vessels on the AC3 SOS opportunity to BLB, where they phased in to the Intra-WCSA.

The LSFRP has received little attention in the literature, and was not mentioned in either of the most influential surveys of work in the liner shipping domain (Christiansen et al. 2007; Christiansen, Fagerholt, and Ronen 2004). Neither the Fleet Deployment Problem (Powell and Perakis 1997) nor the Liner Shipping Network Design Problem (Løfstedt et al. 2010) deals with the repositioning of ships or the important phase-in requirements. Tramp shipping problems, such as (Korsvik, Fagerholt, and Laporte 2011), also differ from the LSFRP due to a lack of cost-saving activities for vessels.

It has been observed in both the AI-planning and OR-scheduling fields (e.g. (Karger, Stein, and Wein 1997; Smith, Frank, and Jónsson 2000)) that the compound objectives of real-world problems, such as those found in the LSFRP, are often hard to express in terms of the simple objective criteria like makespan and tardiness minimization. Scheduling (Karger, Stein, and Wein 1997) has focused mainly on problems that only involve a small, fixed set of choices, while planning problems like the LSFRP often involve cascading sets of choices that interact in complex ways (Smith, Frank, and Jónsson 2000). Another limitation is that mainstream scheduling research has focused mainly on the optimization of selected, simple objective criteria such as minimizing makespan or minimizing tardiness (Smith 2005).

Existing Temporal-Numeric Planners

In this section we consider modelling and then solving the fleet repositioning problem using existing AI planners.

Modelling Fleet Repositioning in PDDL

The PDDL model of the LSFRP (Tierney et al. 2012) has interesting temporal features: required concurrency (Cushing et al. 2007), timed initial-literals (TILs) (Edelkamp and Hoffmann 2003) and duration-dependent effects. A basic model requires 6 actions, `phase-out`, `phase-in`, `sail`, `sail-on-service`, `sail-with-equipment` and calculate-hotel-cost. We discuss only the main points of interest for each action.

The actions phase-out and phase-in have preconditions that depend on absolute time. To model these, we use two collections of TILs: one representing the short windows of opportunity in which a given vessel will be at a given port and can phase out; and another representing the windows during which a port is open (to any ship) for phasing in. This yields problems with a much larger number of TILs than any existing planning benchmark problem: the smallest test problem has 146 TILs, the largest 266.

A challenging constraint from a PDDL modelling perspective is that all the vessels must phase-in in sequential weeks. Critically, it is not that vessels must phase in at exactly (or less than) one week spacings, but that once the first vessel has phased in, there must be one phase-in per calendar week. One can model weeks using propositional TILs adding is-week facts, and a week parameter for phase-in actions. This, however, forces commitment to the phase-in week, leading to backtracking to change what are effectively scheduling decisions. We therefore chose another model: a continually executing action encompassing the plan has an effect (increase (time-elapsed) #t), giving a counter of the current time. The phase-in action for the first ship has conditional effects that set the value of a variable to $t \in [0..w]$, where $t \in \mathbb{N}$ gives the week in which the action occurred, according to (time-elapsed). Subsequent phase-in actions require (time-elapsed) to be in the range $[168t, 168(t+n))$ (n is the number of vessels).

Duration-dependent effects are key in this problem. The cost of sail is higher if a vessel sails faster. It is computed as a fixed cost for the journey, minus a fixed constant multiplied by the duration. The duration can be chosen by the planner, within the bounds of the specified minimum and maximum possible journey time. To compute the hotel cost for a vessel we use an envelope action that must be started before a vessel can phase out, and may only finish once the vessel has phased in. This action, calculate-hotel-cost, also has a duration-dependent effect, increasing cost by a value proportional to its duration. We permit a vessel's calculate-hotel-cost action to end if it begins a sail-on-service action, so that hotel-cost need not be paid, but require it to be executing before sail-on-service ends. The necessary interleaving of actions in this way leads to required-concurrency[3]. We note that it would be more natural to model hotel cost calculation using PDDL+ processes (Fox and Long 2006), however the only available planner to handle these is UPMurphi (Penna et al. 2009), which we could not use directly as it requires the input of a discretization of the model in addition to PDDL. Discretizing the LSFRP is undesirable due to the vast differences in time scale between action durations.

Solving the Fleet Repositioning Problem

The major challenges the LSFRP poses to planners are its tightly coupled temporal and numeric interactions, including duration-dependent effects, and the requirement for optimization with respect to these. In recent years, a number of powerful planners capable of reasoning with durative actions and real-valued variables have been developed. Sapa (Do and Kambhampati 2003), which has a number of heuristics designed to work with multi-objective criteria, and LPG (Gerevini, Saetti, and Serina 2003), which remains one of the best optimizing planners, were amongst the first of these. Whilst a new version of LPG exists that handles required concurrency (Gerevini and Saetti 2010), the scheduling techniques employed by these planners are insufficient to reason about duration-dependent effects (with non-fixed durations). Recent Net-Benefit planners (e.g. HSP*, MIPS-XXL and Gamer) (Helmert, Do, and Refanidis 2008), whilst also strong at optimization, suffer from the same problem.

More sophisticated scheduling techniques exist in the more recent cohort of planners capable of reasoning with continuous numeric change: Colin (Coles et al. 2009), POPF (Coles et al. 2010), Kongming (Li and Williams 2008) and TMLPSAT (Shin and Davis 2005). Each of these makes use of a linear program (LP) or a MIP to perform scheduling with respect to continuous dynamics, and this is sufficient to capture duration-dependent change. Colin and its successor POPF are the only two available systems that can reason with all the necessary language features for this problem. Kongming does not allow multiple parallel updates to the same variable, but here total cost is updated by multiple vessels sailing in parallel, and TMLPSAT does not exist in a runnable form. Colin and POPF are not optimal planners, though POPF has a mechanism for continuing search once a solution is found to improve quality (Coles et al. 2011). Solving the problem proved challenging for POPF, and highlighted that there is much scope for general planning research into solving problems with many TILs and cost-based optimization.

POPF uses a forward-chaining search approach to temporal planning, splitting durative actions into two snap-actions representing the start and end of the action. To ensure a plan is temporally consistent, and to assign timestamps to actions, an LP is used. Each step i of the plan has a real-valued time-stamp variable $step_i$ in the LP, directly constrained in two ways. First, if the search orders i before j, then $step_j \geq step_i + \epsilon$, where ϵ is some small constant. Second, for an action starting at i and finishing at k, the value of $(step_k - step_i)$ must obey the action's duration constraints. The numeric effects and preconditions of actions are captured by adding further variables, constrained to capture actions' preconditions, and to reflect their effects. For example, the effect of hotel-cost-calc (starting at b and ending at e) on total-cost would be appear as a term in the LP objective function as $(step_e - step_b) * hotelcost$.

In POPF, TILs have traditionally been treated as dummy actions that must be applied in sequence order. At each point in search, the planner can choose to apply an action or the next TIL. However, when there are many TILs, as in the LSFRP, the search space is unnecessarily blown up by what are often scheduling choices. To this end, we have developed a generic technique for abstracting TILs representing time windows of opportunity. This introduces binary variables, and hence the LP is now a MIP; though we solve the LP re-

[3] Many problems also exhibit required concurrency insofar as vessels must sail in parallel for a solution to be found.

laxation of the MIP at non-goal states. We consider a fact f suitable for abstraction in this way if it is only ever added and deleted by TILs. A TIL-controlled fact f may be added and deleted multiple times, presenting n windows of opportunity in which to use it. Once identified, we remove the f-TILs from search, and presume f to be true in the initial state. If step i of the plan requires f, we add the following disjunctive temporal problem (DTP) constraints to the MIP:

$$\sum_{k=0}^{n} f_k^+ w_k^i \leq step_i \leq \sum_{k=0}^{n} f_k^- w_k^i$$
$$\sum_{k=0}^{n} w_k^i = 1,$$

where w_k^i is a binary variable indicating whether $step_i$ occurs in window k, and f_k^+ and f_k^- are the timestamps at which f is added and deleted for the kth time, respectively.

A more specialised case of this is a fact f added and deleted by TILs, and all snap-actions with a precondition on f delete it. Again, this is a general idea, e.g. a machine becoming available once a day, and available for one task only. In fleet repositioning, it occurs in the *phase-in* action: once a given vessel phases in at a given opportunity no other vessel can take that opportunity. Given the set of steps F that all refer to f, we add the following constraint:

$$\forall_{k=1}^{n} \sum_{i \in F} w_k^i \leq 1.$$

Having removed these TILs from being under the explicit control of search, we must still consider them in the heuristic. For this, we make a slight modification to Temporal Relaxed Planning Graph (TRPG) expansion: once the non-TIL-controlled preconditions for a snap-action have become true in fact layer t, it appears in the next action layer $t' \geq t$, such that at time t', all the TIL-controlled preconditions of the action would be true. This is stronger than the TRPG previously: effectively, we no longer ignore TIL-delete effects on facts that are only ever added by TILs.

To improve the cost estimation in the heuristic in the presence of duration-dependent costs, we make a further modification. Suppose we have previously found a solution with cost c', and are now evaluating a state S, with cost c. We know that it is not worth taking any action with cost greater than $(c' - c)$ in order to complete the solution plan from S (assuming we have proved cost is monotonically worsening). Therefore, no action can have a duration that would give a duration-dependent cost in excess of this, and we can tighten the bounds on RPG actions' durations accordingly.

Optimality Considerations

To solve this problem to guaranteed optimality we need a temporally-aware cost-sensitive admissible planning heuristic. We can obtain an admissible estimate of the cost of a partial plan $cost_p^s$ to reach state S by solving the corresponding MIP minimizing cost. Given a solution of cost c' we can discard any state with $cost_p^s > c'$ (assuming monotonically increasing cost). This equates to assuming the cost of reaching the goal is zero: an admissible, but poor, heuristic.

POPF records the minimum cost to achieve each fact at each timestamped TRPG layer during graph building (Coles et al. 2011). These estimates are, however, only admissible within the timeframe covered by the TRPG: graph expansion terminates when all goals appear, but expanding the TRPG further could potentially reduce the cost. We can gain admissible estimates of the cost to achieve each goal by expanding the TRPG until new actions have stopped appearing, and costs have stopped changing, as is possible in Sapa (Do and Kambhampati 2003). In general, since one action could contribute towards the achievement of multiple goals, we cannot add the costs of achieving the goals; instead, to maintain admissibility, we can only take the cost of achieving the most expensive goal. Additive h_{max} (Haslum, Bonet, and Geffner 2005) allows some addition of costs with no loss of admissibility, but we leave this for future work.

Temporal Optimization Planning

In the absence of an optimal method for solving problems with duration linked objectives, we introduce Temporal Optimization Planning (TOP). TOP fundamentally diverges from classical AI-planning approaches by introducing two sets of 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 built together to complete optimization models using partial-order planning. This decoupling makes it possible to formulate any objective that can be expressed by the applied optimization model. Moreover, computationally expensive action models, including real-valued state variables and general objective functions, are avoided.

TOP is built on a state variable representation of propositional STRIPS planning (Fikes and Nilsson 1971). TOP 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 modeled. Second, instead of focusing on simply achieving feasibility, TOP 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.

In contrast to the current trend in advanced temporal planning, TOP bypasses computationally expensive dense time models of shared resources like electric power consumption during activities. These models are important for the robotic or aerospace applications often targeted in AI-planning (e.g., (Frank, Gross, and Kurklu 2004; Muscettola 1993)), but TOP focuses on more physically separated activities where resources are exclusively controlled. While this decoupling offers some new possibilities, it makes TOP less capable of solving traditional planning problems, specifically where resources can appear in preconditions and are not solely for tracking an optimization function.

TOP differs from existing temporal planners in two further ways. First, TILs are not needed to model problems in which some actions are only available at specific times, such as the `phase-out` and `phase-in` actions in the LSFRP. Rather, constraints on the start or end time of an action can be built directly into actions' optimization models. Second, through shared variables in their optimization models, actions can refer directly to the start and end times of other actions. This means the encoding of, for example, the hotel cost calculation can be embedded within the effects of other actions that imply it. In PDDL actions cannot refer to the start and end times of other actions, hence the envelope encoding above, which blows up the search space, is necessary.

Formally, let $\mathcal{V} = \{v_1, \cdots, v_n\}$ denote a set of *state variables* with finite domains $D(v_1), \cdots, D(v_n)$. A *state variable assignment* ω is a mapping of state variables to values $\{v_{i(1)} \mapsto d_{i(1)}, \cdots, v_{i(k)} \mapsto d_{i(k)}\}$ where $d_{i(1)} \in D(v_{i(1)}), \cdots, d_{i(k)} \in D(v_{i(k)})$. We also define $vars(\omega)$ as the set of state variables used in ω.

A TOP problem is represented by a tuple

$$P = \langle \mathcal{V}, \mathcal{D}, \mathcal{A}, \mathcal{I}, \mathcal{G}, pre, eff, \boldsymbol{x}, obj, con \rangle,$$

where $\mathcal{D}$ is the Cartesian product of the domains $D(v_1) \times \cdots \times D(v_n)$, $\mathcal{A}$ is the set of actions, $\mathcal{I}$ is a total state variable assignment (i.e. $vars(\mathcal{I}) = \mathcal{V}$) representing the initial state, $\mathcal{G}$ is a partial assignment (i.e. $vars(\mathcal{G}) \subseteq \mathcal{V}$) representing the goal states, pre_a is a partial assignment representing the precondition of action a, eff_a is a partial assignment representing the effect of action a[4], $\boldsymbol{x} \in \mathbb{R}^m$ is a vector of optimization variables[5] that includes the begin and end time of each action, x_b^a and x_e^a respectively, for all actions $a \in A$, $obj_a : \mathbb{R}^m \to \mathbb{R}$ is a cost term introduced by action a, and $con_a : \mathbb{R}^m \to \mathbb{B}$ is a constraint expression introduced by action a with $con_a \models x_b^a \leq x_e^a \wedge x_b^a \geq 0 \wedge x_e^a \geq 0$.

Let $S = \{\omega | vars(\omega) = \mathcal{V}\}$ denote the set of all the possible states. An action a is applicable in $s \in S$ if $pre_a \subseteq s$ and is assumed to cause an instantaneous transition to a successor state defined by the total assignment

$$succ_{a,s}(v) = \begin{cases} eff_a(v) & \text{if } v \in vars(eff_a), \\ s(v) & \text{otherwise.} \end{cases}$$

We further define $M_a = \min\{obj_a | con_a\}$, which is the minimal cost of action a's optimization model component.

A temporal optimization plan is represented by a tuple $\langle A, C, O, M \rangle$, where A is the set of actions in the plan, C is a set of causal links $a \xrightarrow{\mu} b$ with $a, b \in A$ and $\mu \in eff_a \cup pre_b$, O is a set of ordering constraints of the form $a \prec b$ with $a, b \in A$, and M is an optimization model associated with the plan defined by

$$\begin{aligned} \min \quad & \sum_{a \in A} obj_a(\boldsymbol{x}) \\ \text{s.t.} \quad & x_e^{a_i} \leq x_b^{a_j} \qquad \forall a_i \prec a_j \in O \qquad (1) \\ & con_a(\boldsymbol{x}) \qquad \forall a \in A. \qquad (2) \end{aligned}$$

The objective of M is to minimize the sum of the costs introduced by actions, subject to action orderings (1) and the constraints associated with each action in π (2). Let $cost(\pi)$ be the cost of an optimal solution to M to a partial plan π.

An *open condition* $\xrightarrow{\mu} b$ is an unfulfilled precondition μ of action $b \in A$, that is, $\mu \in pre_b$ and $\forall a \in A, a \xrightarrow{\mu} b \notin C$. An *unsafe link* is a causal link $a \xrightarrow{\mu} b$ that is threatened by an action c such that *i)* $vars(\mu) \in eff_c$, *ii)* $\mu \notin eff_c$, and *iii)* $\{a \prec c \prec b\} \cup O$ is consistent.

To deal with durative actions in TOP we need to keep track of another type of flaw called *interference*. We adopt an interference model based on the exclusive right to state variables (Sandewall and Rönnquist 1986). Thus, two actions a and b interfere if $vars(eff_a) \cap vars(eff_b) \neq \emptyset$ and O implies neither $a \prec b$ nor $b \prec a$.

An open condition flaw $\xrightarrow{\mu} b$ can be repaired by linking μ to an action a such that $\mu \in eff_a$ and by posting an ordering constraint over a and b. Thus, $C \leftarrow C \cup \{a \xrightarrow{\mu} b\}$ and $O \leftarrow O \cup \{a \prec b\}$. In the case that $a \notin A$, $A \leftarrow A \cup \{a\}$ and $O \leftarrow O \cup \{a_0 \prec a, a \prec a_\infty\}$.

An unsafe link $a \xrightarrow{\mu} b$ that is threatened by action c can be repaired by either adding the ordering constraint $c \prec a$ (*demotion*) or $b \prec c$ (*promotion*) to O. Similar to unsafe links, an interference between actions a and b can be fixed by posting either $a \prec b$ or $b \prec a$ to O.

Together, open conditions, unsafe links and interferences constitute *flaws* in a plan. Let $flaws(\pi) = open(\pi) \cup unsafe(\pi) \cup interfere(\pi)$ be the set of flaws in the plan π, where $open(\pi)$ is the set of open conditions, $unsafe(\pi)$ is the set of unsafe links, and $interfere(\pi)$ is the set of interferences. We say that π is a *complete plan* if $|flaws(\pi)| = 0$, otherwise π is a *partial plan*. A plan π^* is optimal if it is feasible and for all feasible solutions π, $cost(\pi^*) \leq cost(\pi)$.

Linear Temporal Optimization Planning

To solve the LSFRP, we introduce linear temporal optimization planning (LTOP). In LTOP, all of the optimization models associated with planning actions have a linear cost function and a conjunction of linear constraints. Thus, obj_a is of the form $\mathbf{c^a}\boldsymbol{x'}$, where $\mathbf{c^a} \in \mathbb{R}^m$ and con_a is of the form $\bigwedge_{1 \leq i \leq n_a} (\boldsymbol{\alpha}_i^a \boldsymbol{x'} \leq \beta_i)$, where $\boldsymbol{\alpha}_i^a \in \mathbb{R}^m, \beta_i \in \mathbb{R}$ (n_a is the number of constraints associated with action a). Thus, M_a and M are LPs. Note that M is very similar to the LPs in POPF, and serves a similar purpose: enforcing temporal constraints and optimizing cost.

Figure 2 shows an example LTOP plan for the repositioning in Figure 1. The optimization variables $h_{b,v}$ and $h_{e,v}$, representing the begin and end of the hotel period, respectively, are of particular note, as they replace the action `calculate-hotel-cost` required by our PDDL model. Each action updates the upper bound of $h_{b,v}$, this shared variable allows the hotel cost of the vessel to be accounted for, even in a partial plan. An example of implicit TIL handling within LTOP can be seen in the *out* action. The starting time of the action is bound to $t_{CHX,TPP}^{out}$, which is a constant representing the time the vessel may phase out at port TPP.

Algorithm 1 shows a branch-and-bound algorithm that finds an optimal plan to an LTOP problem, based on the POP algorithm in (Williamson and Hanks 1996). First, an initial plan π_{init} is created by the INITIALTOP function (line 2). We define $\pi_{init} = \langle \{a_0, a_\infty\}, \emptyset, \{a_0 \prec a_\infty\}, M_{init} \} \rangle$, where a_0 is an action representing $\mathcal{I}$ with $pre_{a_0} = \emptyset$ and

[4]In practice, it is often more convenient to represent actions in a more expressive form, e.g. by letting the precondition be a general expression on states $pre_a : S \to \mathbb{B}$ and represent conditional effects like resource consumption by letting the effect be a general transition function, depending on the current state of S, $eff_{a,s} : S \to S$. Such expressive implicit action representations may also be a computational advantage. We have chosen a ground explicit representation of actions because it simplifies the presentation and more expressive forms can be translated into it.

[5]We sometimes let $\boldsymbol{x}$ denote a set rather than a vector.

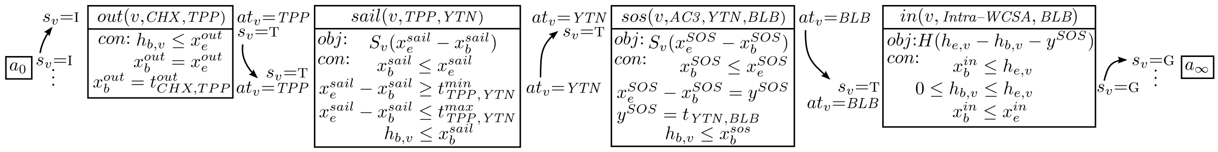

Figure 2: A complete TOP plan showing a solution to the repositioning in Figure 1. Boxes represent actions and contain their associated optimization models. Causal links are shown with arrows. The optimization variables x_b^a and x_e^a represent the begin and end time of action a, and $h_{b,v}$, $h_{e,v}$ are the begin and end hotel time of the vessel, respectively. The state variable $s_v \in \{\text{I}, \text{T}, \text{G}\}$ represents the vessel being on its initial service, in transit or at its goal service, respectively.

Algorithm 1 Temporal optimization planning algorithm.

1: **function** TOP($\mathcal{I}$, $\mathcal{G}$)
2: $\quad \Pi \leftarrow \{$INITIALTOP($\mathcal{I}$,$\mathcal{G}$)$\}$
3: $\quad \pi_{best} \leftarrow null$
4: $\quad u \leftarrow \infty$ ▷ Cost of the incumbent (upper bound)
5: $\quad$ **while** $\Pi \neq \emptyset$ **do**
6: $\quad\quad \pi \leftarrow$ SELECTPLAN(Π)
7: $\quad\quad \Pi \leftarrow \Pi \setminus \{\pi\}$
8: $\quad\quad$ **if** NUMFLAWS(π) $= 0 \wedge$ COST(π) $< u$ **then**
9: $\quad\quad\quad u \leftarrow$ COST(π)
10: $\quad\quad\quad \pi_{best} \leftarrow \pi$
11: $\quad\quad$ **else if** ESTIMATECOST(π) $< u$ **then**
12: $\quad\quad\quad f \leftarrow$ SELECTFLAW(π)
13: $\quad\quad\quad \Pi \leftarrow \Pi \cup$ REPAIRFLAW(π, f)
14: $\quad$ **return** π_{best}

$\mathit{eff}_{a_0} = \mathcal{I}$; a_∞ is an action representing $\mathcal{G}$ with $pre_{a_\infty} = \mathcal{G}$ and $\mathit{eff}_{a_\infty} = \emptyset$; and M_{init} is an optimization model with no objective and two constraints, con_{a_0} and con_{a_∞}, which are special constraints on the dummy actions a_0 and a_∞ such that $con_{a_0} = (x_b^{a_0} = x_e^{a_0} \wedge x_b^{a_0} \ge 0)$ and $con_{a_\infty} = (x_b^{a_\infty} = x_e^{a_\infty} \wedge x_b^{a_\infty} \ge 0)$. The optimization variables $x_b^{a_0}, x_e^{a_0}, x_b^{a_\infty}$ and $x_e^{a_\infty}$ represent the begin and end times of actions a_0 and a_∞ respectively. The algorithm then selects a plan from Π (line 6) and checks if it is a complete plan. If so, its cost is compared with the current upper bound (u), and if the cost is lower, the *incumbent* π_{best} is replaced with the current plan π and the upper bound is updated (lines 9 and 10). When π is a partial plan, an estimated lower bound of the plan is computed: if it is higher than the cost of the incumbent solution, the plan is discarded (line 11). Otherwise, a flaw is selected and repaired (lines 12 and 13). This process is repeated until Π=$\emptyset$, then the current incumbent is returned, if one exists.

The algorithm of POPF (if we force A*) can be understood in a similar manner. POPF uses different heuristics in SELECTPLAN(Π) and to compute COST(π) and ESTIMATECOST(π). Instead of selecting a flaw to repair (lines 12,13), POPF creates a new plan by selecting an action to append (Π initially contains the empty plan). Effectively, POPF searches forwards, while LTOP searches backwards.

Algorithm 1 is guaranteed to find the optimal solution (if there is one) as long as ESTIMATECOST does not overestimate the cost of completing a partial plan. To prune as much of the branch-and-bound tree as possible, we need tight lower bounds. If we require that the cost of each action subject to its constraints is non-negative, we can prove that $cost(\pi)$ is such a lower bound.

Proposition 1. *Given any valid partial plan* $\pi = \langle A, C, O, M\rangle$ *where* $M_a \ge 0$, $\forall a \in A$, $cost(\pi) \le cost(\bar{\pi})$ *for any completion* $\bar{\pi}$ *of* π.

Proof. Let π' be π with a single flaw repaired. The flaw is either *i*) an unsafe link, *ii*) an interference, *iii*) an open condition being satisfied by an action in the plan, or *iv*) an open condition being satisfied by an action not in the plan.

In cases *i* and *ii* the flaw is repaired by adding an ordering constraint to π, which further constrains π, thus $cost(\pi) \le cost(\pi')$. Case *iii* results in a new causal link and an ordering constraint, and is therefore the same as cases *i* and *ii*. In case *iv*, the action's optimization model is added to π, but since the cost function of the action must be non-negative under its constraints, $cost(\pi')$ cannot be less than $cost(\pi)$. By applying this argument inductively on the complete branch-and-bound subtree grown from π, we get $cost(\pi) \le cost(\bar{\pi})$ for any completion $\bar{\pi}$ of π. □

Heuristic Cost Estimation

Although $cost(\pi)$ provides a reasonable lower bound for π, the bound is only computed over actions in the plan. As we noted when discussing POPF, it can be strengthened by also reasoning over actions that are needed to complete the plan. We present an extension of h_{max} (Haslum and Geffner 2000), h^{cost}_{max}, which estimates the cost of achieving the open conditions of a plan π in a similar manner to VHPOP (Younes and Simmons 2003). The extension is that instead of using action cost, we use the (precomputed) value of the minimized objective model of an action (M_a).

$$h^{cost}_{max}(\omega, \pi) = \begin{cases} 0 & \text{if } \omega \subseteq \mathit{effs}_\pi\text{, else} \\ f(\omega, \pi) & \text{if } \omega = \{\mu\}\text{, else} \\ g(\omega, \pi) & \text{if } |\omega| > 1, \end{cases}$$

$$f(\omega, \pi) = \min_{\{a \in \mathcal{A} \setminus A | \mu \in \mathit{eff}_a\}} \{M_a + h^{cost}_{max}(pre_a, \pi)\},$$
$$g(\omega, \pi) = \max_{\mu \in \omega} \{h^{cost}_{max}(\{\mu\}, \pi)\},$$

where ω is a partial state variable assignment, μ is a single state variable assignment $v \mapsto d$, and $\mathit{effs}_\pi = \bigcup_{a \in A} \mathit{eff}_a$. The heuristic takes the max over the estimated cost of achieving the elements in the given assignment ω. The cost is zero if the elements are already in π, otherwise the minimum cost of achieving each element is computed by finding the cheapest way of bringing that element into the plan. A comparison of h_{max} to costed-RPG style heuristics (such as that of POPF) can be found in (Do and Kambhampati 2003).

It is possible to extend more recent work in admissible heuristics for cost-optimal planning (Haslum et al. 2007; Helmert, Haslum, and Hoffmann 2007; Helmert and Domshlak 2009; Katz and Domshlak 2010; Haslum, Bonet, and Geffner 2005) in the same way to produce even more accurate admissible estimates, we leave this to future work.

Proposition 2. *Given any valid partial plan* $\pi = \langle A, C, O, M\rangle$ *where* $M_a \geq 0$, $\forall a \in A$, $cost(\pi) + h^{cost}_{max}(open(\pi), \pi) \leq cost(\bar{\pi})$ *for any completion* $\bar{\pi}$ *of* π.

Proof. We have $h^{cost}_{max}(\omega, \pi) = \sum_{a\in R} M_a$, where R is a set of actions not currently in π ($R \cap A = \emptyset$) that are required to resolve ω and among such sets has the minimum cost. Thus, any completion $\bar{\pi}$ of π as described in Proposition 1 must at least increase $cost(\pi)$ by $h^{cost}_{max}(\omega, \pi) = \sum_{a\in R} M_a$. □

Domain Specific Heuristics We also explored domain specific heuristics in order to better solve the LSFRP. First, we modified the branching scheme of LTOP in order to avoid multiple `sail` actions in a row, observing that it will never be cheaper to sail through an intermediate port than to directly sail between two ports. This is straightforward to implement, requiring only for LTOP to check if adding a `sail` action to a partial plan would result in two sailings in a row.

Our second heuristic is able to complete a vessel's repositioning once the vessel is assigned a `sail-on-service` action. Since the starting time of the `sail-on-service` is fixed, and so are the times of the phase-outs, the optimal completion to the plan can be computed by simply looping over the vessel's allowed phase-out ports and choosing the one with the lowest cost. We add a `phase-out` action to the plan along with a `sail` action, if necessary.

Mixed Integer Programming (MIP) Model

We have modeled the same subset of fleet repositioning as in our PDDL and LTOP models using a MIP. Our model considers the activities that a vessel may undertake and connects activities based on which ones can feasibly follow one another temporally. Thus, the structure of the LSFRP is embedded directly into the graph of our MIP, meaning that the MIP is unable to model general automated planning problems as in (Van Den Briel, Vossen, and Kambhampati 2005) and (Kautz and Walser 1999). Note that, unlike LTOP, the MIP is capable of handling negative costs.

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 or not 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 $x^s_a, x^e_a \in \mathbb{R}^+$ are action a's start and end time, respectively. Finally, the variables h^s_v and h^e_v 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^{min}_a and d^{max}_a. 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 $\eta : A \rightarrow 2^{|A|}$. There are also n sets of mutually exclusive actions, given by $\mu : \{1, \ldots, n\} \rightarrow 2^{|A|}$. We differentiate between *phase-out* and *phase-in* actions for each vessel using the sets $A^{po}_v, A^{pi}_v \subseteq A^t$, respectively, and let $A' = A \setminus \cup_{v\in V}(A^{po}_v \cup A^{pi}_v)$. Finally, let $c^H_v \in \mathbb{R}^+$ represent each vessel's hourly hotel cost.

There are several "big-Ms" in the model, which are constants used in MIP models to enforce logical constraints. The upper bound on the difference between the end and start of two actions is given by $M^y_{a,b}$. 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^s_v and m^e_v respectively. The MIP model is as follows:

$$\min \sum_{a\in A}(c_a w_a + \alpha_a(x^e_a - x^s_a)) + \sum_{v\in V} c^H_v (h^e_v - h^s_v)$$

$$\text{s.t.} \sum_{(a,b)\in T} y_{a,b} = 1 \qquad \forall a \in A^{po}_v, b \in A \setminus A^{po}_v, v \in V \tag{3}$$

$$\sum_{(a,b)\in T} y_{a,b} = \sum_{(b,c)\in T} y_{b,c} \qquad \forall b \in A' \tag{4}$$

$$\sum_{(a,b)\in T} y_{a,b} \leq 1 \qquad \forall b \in A' \tag{5}$$

$$x^e_a - x^s_b \leq M^y_{a,b}(1 - y_{a,b}) \qquad \forall (a, b) \in T \tag{6}$$

$$x^s_a \leq x^e_a \qquad \forall a \in A \tag{7}$$

$$d^{min}_a w_a \leq x^e_a - x^s_a \leq d^{max}_a w_a \qquad \forall a \in A \tag{8}$$

$$x^s_a = t_a w_a \qquad \forall a \in A^t \tag{9}$$

$$h^s_v + M^s_v w_a \leq M^s_v + t_a \qquad \forall a \in A^{po}_v, v \in V \tag{10}$$

$$h^e_v + m^e_v w_a \geq m^e_v + t_a \qquad \forall a \in A^{pi}_v, v \in V \tag{11}$$

$$\sum_{a\in\mu(i)} w_a \leq 1 \qquad \text{for } i = 1, 2, \ldots n \tag{12}$$

$$|\eta(a)|w_i + \sum_{b\in\eta(b)} w_b \leq |\eta(a)| \qquad \forall a \in A \tag{13}$$

The objective sums the fixed and variable costs of each action that is used along with the hotel cost for each vessel. The single unit flow structure of the graph is enforced in (3) – (5), and (6) enforces 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 (7), and the duration of each action is limited by (8). Actions with fixed start times are bound to this time in (9), and (10) – (11) connect the hotel start and end times to the time of the first and last action, respectively. The mutual exclusivity of certain sets of actions is enforced in (12), and (13) prevents actions from being included in the plan if they are excluded by an action that was chosen.

Experimental Evaluation

Using a dataset of real-world instances based on a scenario from our industrial collaborator, we evaluated the performance of POPF, LTOP, and our MIP model. We created ten instances based on the case study shown in Figure 1 containing up to three vessels and various combinations of sail-on-service, equipment opportunities (e) and cabotage restrictions (c). The instances have between 99 and 590 grounded actions in LTOP, and between 470 and 2378 decision variables in the reduced MIP computed by CPLEX.

Table 1 shows the results of solving the LSFRP to optimality with the MIP model and with LTOP, and suboptimally with POPF[6]. We explored the performance of several combinations heuristics in LTOP, using domain specific

[6]All experiments were conducted on AMD Opteron 2425 HE processors with a maximum of 4GB of RAM per process. The MIP and LTOP used CPLEX 12.3, POPF used CPLEX 12.1.

Inst.	MIP	LTOP				POPF (Optimal)		POPF (Satisficing)									
		DLH	DL	LH	L	Forwards	Reversed	Standard		Makespan		No MIP relax		No-TIL-Abs		Reversed	
AC3_1_0	**0.4**	1.1	1.1	1.1	1.1	0.7	1.4	0.4	(0.0)	0.1	(1.7)	0.7	(0.0)	105.8	(0.0)	0.4	(0.0)
AC3_2_0	**9.3**	51.0	51.5	50.4	53.5	-	809.6	32.5	(0.0)	3.2	(1.6)	113.2	(0.0)	13.0	(0.1)	78.1	(0.0)
AC3_3_0	**23.0**	188.3	196.8	193.3	202.0	-	-	1105.1	(0.0)	117.5	(2.3)	3041.6	(0.0)	88.2	(0.1)	39.2	(0.8)
AC3_1_1e	**3.8**	3.9	3.9	5.2	5.3	3.3	4.0	1.7	(0.0)	0.1	(0.7)	2.3	(0.0)	1079.3	(0.3)	1.2	(1.2)
AC3_2_2ce	27.7	**15.2**	25.2	55.2	126.8	-	-	1550.6	(0.3)	1.1	(19)	2284.2	(0.0)	31.3	(3.7)	892.5	(1.6)
AC3_3_2c	250.5	**203.2**	362.2	2979.7	3715.8	-	-	399.2	(0.2)	9.2	(7.3)	26.3	(1.4)	303.4	(1.3)	602.8	(1.1)
AC3_3_2e	228.8	**217.1**	263.0	1453.1	2092.8	-	-	291.5	(1.3)	9.6	(11)	28.4	(2.4)	310.8	(2.3)	688.6	(1.9)
AC3_3_2ce1	312.2	**218.2**	260.8	1451.6	2068.4	-	-	303.9	(1.3)	9.7	(11)	28.4	(2.4)	314.5	(2.3)	697.2	(1.9)
AC3_3_2ce2	252.6	**192.4**	216.0	2624.1	3094.3	-	-	1464.2	(1.6)	10.0	(12)	204.9	(2.8)	303.4	(2.7)	690.1	(2.3)
AC3_3_2ce3	706.5	**516.9**	685.5	2959.1	-	-	-	348.0	(1.1)	10.3	(8.7)	29.4	(1.9)	308.4	(1.7)	603.3	(1.5)
AC3_3_3	148.3	**80.0**	102.4	735.0	1140.8	-	-	1975.5	(2.3)	10.1	(15)	226.0	(3.6)	352.6	(3.4)	699.6	(2.9)

Table 1: Results comparing our MIP model to POPF and LTOP using several different planning heuristics with a timeout of one hour. All times are the CPU time in seconds. Figures in brackets are the best optimality gap found by POPF alongside the CPU time required to find it. The optimality gap is computed by $(c - c^*)/c^*$, where c is the plan cost and c^* is the optimal solution.

heuristics (D), the h^{cost}_{max} heuristic (H), and using the LP of a partial plan (L). The MIP outperforms LTOP on AC3_1_0 through AC3_1_1e, the smallest instances in our dataset. Once the instances begin growing in size with AC3_2_2ce, LTOP requires only 75% of the time of the MIP with the DLH heuristics. The MIP easily outperforms LTOP with only domain independent heuristics (LH and L), but this is not surprising considering that the MIP is able to take our domain specific heuristics into account through its graph construction. The instance that most realistically represents the scenario our industrial collaborator faced is AC3_3_3, which LTOP is able to solve in slightly over half the time of the MIP. Overall, the MIP requires an average time of 178 seconds versus only 153 seconds for LTOP on our dataset.

POPF is a highly expressive general system and as a result of the overhead of the additional reasoning it has to do is not as efficient as the MIP or LTOP. Each temporal action is split into a start and end action (necessary for completeness in general temporal planning), which dramatically increases plan length. Recall also that the PDDL model contains actions to model hotel cost calculation, which also introduces extra steps in the search, meaning the POPF plan corresponding to the LTOP plan of length four in Figure 2 has ten steps.

Not only does POPF have a deeper search tree, it has a higher branching factor, due to the above factors and plan permutations. The state memoization in POPF is not sophisticated enough to recognize that different orders of `hotel-cost-calc` actions for different vessels are equivalent (similarly for unrelated `hotel-cost-calc` and `sail` actions). Thus, if using POPF to prove optimality (A*, admissible costs from expanding the TRPG fully) it spends almost all of its time considering permutations of `hotel-cost-calc` actions. To synthesize as close a comparison to LTOP as possible, we made a reversed domain: vessels begin by phasing in and end by phasing out. We did not model the problem this way initially due to the 'physics, not advice' mantra of PDDL: it is less natural, though more efficient here, and closer to LTOP. Using this, POPF can prove optimality in only 3 problems: AC3_1_0 and AC3_1_1e which have 1 vessel; and AC3_2_0 which has 2 vessels. In AC3_1_1e, it expands twice as many nodes (and evaluating each takes far longer.) This is an interesting new problem for temporal-numeric planning research, motivating research into supporting processes to avoid explicit search over cost-counting actions, and more sophisticated state memoization.

To highlight some of the successes of POPF, when performing satisficing search, the column 'Standard' in Table 1 demonstrates that POPF is sensitive to the metric specified, and is successfully optimizing with respect to a cost function that is not makespan. The 'Makespan' results confirm that optimizing makespan would not be a surrogate for low-cost in this domain, and indeed reflect that whilst POPF does not find optimal solutions in all problems, the solutions it is finding are relatively rather good.

As an evaluation of our modifications to POPF, the 'No MIP relax' column indicates performance when not relaxing the MIP to an LP at non-goal states. This configuration suffers from high per-state costs, limiting the search space covered in one hour. An alternative means of avoiding the MIP is to disable TIL abstraction, which again is demonstrably worse than the 'Standard' configuration. Finally, the 'Reversed' model, though better for optimal search, gives worse performance: it forces premature commitment to the phase-in port without having considered how to sail there. This is harmless in the optimal case, where all phase-in options are considered anyway, but detrimental here.

Conclusion

We have shown three methods of solving the LSFRP, an important real-world problem for the liner shipping industry, and in doing so extended the TIL handling capabilities of the planner POPF, as well as introduced the novel method TOP. Our results indicate that automated planning techniques are capable of solving real-world fleet repositioning problems within the time required to create a usable LSFRP decision support system. For future work, we will explore tighter lower bounds in LTOP, as well as improved state memoization and the reduction of dummy actions in POPF.

Acknowledgements

We would like to thank our industrial collaborators Mikkel Muhldorff Sigurd and Shaun Long at Maersk Line for their support and detailed description of the fleet repositioning problem. This research is sponsored in part by the Danish Council for Strategic Research as part of the ENERPLAN research project. Amanda Coles is funded by EPSRC Fellowship EP/H029001/1.

References

Christiansen, M.; Fagerholt, K.; Nygreen, B.; and Ronen, D. 2007. Maritime transportation. *Transportation* 14:189–284.

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

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2009. Temporal planning in domains with linear processes. In *IJCAI-09*.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *ICAPS-10*.

Coles, A. J.; Coles, A. I.; Clark, A.; and Gilmore, S. T. 2011. Cost-sensitive concurrent planning under duration uncertainty for service level agreements. In *ICAPS-11*.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. 2007. When is temporal planning *really* temporal planning? In *IJCAI-07*, 1852–1859.

Do, M., and Kambhampati, S. 2003. Sapa: A multi-objective metric temporal planner. *JAIR* 20(1):155–194.

Edelkamp, S., and Hoffmann, J. 2003. PDDL2.2: The language for the classical part of the 4th international planning competition. Technical Report No. 195, Institut für Informatik.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3-4):189–208.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *JAIR* 27(1):235–297.

Frank, J.; Gross, M.; and Kurklu, E. 2004. SOFIA's choice: an AI approach to scheduling airborne astronomy observations. In *Proc. National Conference on Artificial Intelligence*, 828–835.

Gerevini, A., and Saetti, A. 2010. Temporal planning with problems requiring concurrency through action graphs and local search. In *ICAPS-10*.

Gerevini, A.; Saetti, A.; and Serina, I. 2003,. Planning through stochastic local search and temporal action graphs. In *JAIR*, volume 28, 239–290.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proceedings of AIPS*, 140–149.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *AAAI-07*.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New Admissible Heuristics for Domain-Independent Planning. In *AAAI-05*.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: Whats the difference anyway? In *ICAPS-09*.

Helmert, M.; Do, M.; and Refanidis, I. 2008. The sixth international planning competition, deterministic track.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS-07*.

Karger, D.; Stein, C.; and Wein, J. 1997. Scheduling algorithms. *CRC Handbook of Computer Science*.

Katz, M., and Domshlak, C. 2010,. Implicit abstraction heuristics. In *JAIR*, volume 39, 51–126.

Kautz, H., and Walser, J. 1999. State-space planning by integer optimization. In *Proc. National Conference on Artificial Intelligence*, 526–533.

Korsvik, J.; Fagerholt, K.; and Laporte, G. 2011. A large neighbourhood search heuristic for ship routing and scheduling with split loads. *Computers & Operations Research* 38(2):474 – 483.

Li, H., and Williams, B. 2008. Generative planning for hybrid systems based on flow tubes. In *ICAPS-08*.

Løfstedt, B.; Alvarez, J.; Plum, C.; Pisinger, D.; and Sigurd, M. 2010. An integer programming model and benchmark suite for liner shipping network design. Technical Report 19, DTU Management.

Muscettola, N. 1993. HSTS: Integrating planning and scheduling. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling*. Morgan Kaufmann. 169–212.

Penberthy, J., and Weld, D. 1992. UCPOP: A sound, complete, partial order planner for ADL. In *Proc. 3rd Int. Conference on Knowledge Representation and Reasoning*.

Penna, G. D.; Intrigila, B.; Magazzeni, D.; and Mercorio, F. 2009. UPMurphi: a Tool for Universal Planning on PDDL+ Problems. In *ICAPS-09*.

Powell, B., and Perakis, A. 1997. Fleet deployment optimization for liner shipping: An integer programming model. *Maritime Policy and Management* 24(2):183–192.

Sandewall, E., and Rönnquist, R. 1986. A Representation of Action Structures. In *Proceedings of 5th (US) National Conference on Artificial Intelligence*, 89–97.

Shin, J., and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artificial Intelligence* 166(1-2):194–253.

Smith, D.; Frank, J.; and Jónsson, A. 2000. Bridging the gap between planning and scheduling. *The Knowledge Engineering Review* 15(1):47–83.

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

Tierney, K.; Coles, A.; Coles, A.; and Jensen, R. M. 2012. A PDDL Domain of the Liner Shipping Fleet Repositioning Problem. Technical Report TR-2012-152, IT University of Copenhagen.

United Nations Conference on Trade and Development. 2011. Review of maritime transport.

Van Den Briel, M.; Vossen, T.; and Kambhampati, S. 2005. Reviving integer programming approaches for AI planning: A branch-and-cut framework. In *ICAPS 2005*, 310–319.

Williamson, M., and Hanks, S. 1996. Flaw selection strategies for value-directed planning. In *AIPS-96*, volume 23, 244.

Younes, H., and Simmons, R. 2003,. VHPOP: Versatile heuristic partial order planner. In *JAIR*, volume 20, 405–430.