View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by edoc

Proceedings of the Tenth International Symposium on Combinatorial Search (SoCS 2017)

Optimal Solutions to Large Logistics Planning Domain Problems

Gerald Paul
Boston University
Boston, Massachusetts, USA
gerryp@bu.edu

Gabriele Röger, Thomas Keller, Malte Helmert
University of Basel
Basel, Switzerland
{gabriele.roeger,tho.keller,malte.helmert}@unibas.ch

Abstract

We propose techniques for efficiently determining optimal solutions to large logistics planning domain problems. We map a problem instance to a directed graph and show that no more than one vehicle per weakly connected component of the graph is needed for an optimal solution. We propose techniques for efficiently finding the vehicles which must be employed for an optimal solution. Also we develop a strong admissible heuristic based on the analysis of a directed graph, the cycles of which represent situations in the problem state in which a vehicle must visit a location more than once. To the best of our knowledge, ours is the first method that determines optimal solutions for large logistics instances (including the largest instances in the IPC 1998 and IPC 2000 problem sets).

Introduction

The LOGISTICS domain (McDermott 2000) is a classical planning domain where packages must be delivered inside and between cities, using trucks and airplanes. It has been used in the International Planning Competition 1998 and 2000 (McDermott 2000; Bacchus 2001). While plans for LOGISTICS tasks can be found in polynomial time, it is NP-complete to decide whether there is a plan within a given cost-bound (Helmert 2003). Therefore, it is not possible to generate *optimal* plans in polynomial time unless P = NP. LOGISTICS also does not admit a polynomial-time approximation scheme unless P = NP (Helmert, Mattmüller, and Röger 2006; Helmert 2008), but there is a polynomial $4/3$-approximation algorithm (Helmert, Mattmüller, and Röger 2006; Helmert 2008). In this work, we solve large LOGISTICS problem instances *optimally*.

The core of our approach uses A* search with an admissible heuristic. However, the presence of many trucks and airplanes increases the difficulty of the problem significantly. To address this issue, we describe a process of *multi-vehicle simplification*; using a mapping of the problem to a directed graph, we show that no more than one airplane (truck) per weakly connected component of the graph is necessary for an optimal solution and we describe a method to determine the necessary vehicles. We also apply some domain-specific *search space pruning* techniques. For the *heuristic*, we extend the idea of a recently proposed heuristic for FreeCell solitaire games (Paul and Helmert 2016) to handle the additional complications inherent in logistics problems. The FreeCell heuristic is based on breaking cycles in a state-specific graph. Our generalization is based on the new insight that actually this graph encodes orderings between disjunctive action landmarks and we hope that in the future we can further generalize this idea into a domain-independent heuristic. We conclude the paper with an experimental evaluation.

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

LOGISTICS Tasks

A LOGISTICS task consists of locations situated in cities within which trucks can transport packages initially assigned to these locations. One or more trucks are assigned to each city and one location in each city is designated as an airport. One or more airplanes can travel between airports transporting packages between the cities, loading and unloading packages at airports. The goal is to move packages from their initial locations to designated goal locations.

Definition 1 (LOGISTICS Task)**.** *A* LOGISTICS *task is given as a tuple* $\langle L, C, P, T, A, city, airport, origin, dest\rangle$*, where*

- L *is a finite set of* locations*,*
- C *is a finite set of* cities*,*
- P *is a finite set of* packages*,*
- T *is a finite set of* trucks*,*
- A *is a finite set of* airplanes*,*
- $city : L \to C$ *assigns each location a city,*
- $airport : C \to L$ *assigns each city an airport location in this city, i. e.* $city(airport(c)) = c$ *for all* $c \in C$*,*
- $origin : P \cup T \cup A \to L$ *specifies the origin location of each package, truck and airplane, where the origin of an airplane is always an airport location, and*
- $dest : P \to L$ *defines a destination for each package.*

A *vehicle* is a truck or an airplane. A *state* s of a LOGISTICS task maps each vehicle v to a location $s(v)$ and each package p to a location, truck or airplane $s(p)$. The *initial state* is given by *origin*. There are four types of operators:

- Vehicles v can *load* packages p at the same location: $load(v, p, l)$ is applicable in state s if $s(v) = s(p) = l$ and leads to state s' that only differs from s in $s'(p) = v$.

- Vehicles v can *unload* loaded packages p: *unload*(v, p, l) is applicable in state s if $s(p) = v$ and $s(v) = l$, and leads to state s' that only differs from s in $s'(p) = l$.
- Trucks t can *drive* to locations l in the same city:[1] *drive*(t, l) is applicable in state s if *city*$(s(t)) = $ *city*(l) and leads to state s' that only differs from s in $s'(t) = l$.
- Airplanes a can *fly* to all airport locations l: *fly*(a, l) is applicable in state s if $l =$ *airport*(c) for some city c. The resulting state s' only differs from s in $s'(a) = l$.

A *plan* is a sequence of operators that are successively applicable to the initial state and lead to a state s_G with $s_G(p) =$ *dest*(p) for all packages $p \in P$. The *cost* of a plan is the length of the operator sequence. A plan is *optimal* if it has minimum cost among all plans.

We call packages that have the origin and the destination in the same city *intracity* packages and all other packages *intercity* packages. If a package p is in a vehicle at location l or it is directly at l, we refer to l as the *position* $pos_s(p)$ of p in state s; formally, $pos_s(p) = s(p)$ if $s(p) \in L$ and $pos_s(p) = s(s(p))$ if $s(p) \in T \cup A$. We use the term *region* to denote all locations of a city (a *truck region*) or all airports (a *plane region*).

Delivery Graphs

If a package is not at its destination location, it must be transported there somehow. As trucks can only move inside cities and airplanes only between airports, a package whose current position is not in the same city as its destination must be transported from the airport of the current city to the airport of the destination city. Moreover, if the current position and/or the destination location is not the airport, it must be transported to/from the airport. Based on such insights, we can identify pairs of cities or locations between which a package must be transported in every plan by the same type of vehicle (trucks or airplanes). We represent this kind of information in so-called delivery graphs.

Definition 2 (Airplane Delivery Graph)**.** *For state s of logistics task $\langle L, C, P, T, A, city, airport, origin, dest\rangle$, the* airplane delivery graph *is the directed graph $D_s^A = (C, E)$, where $E = \{(c, c') \mid$ there is a $p \in P$ s.t. $c =$ *city*$(pos_s(p)) \neq$ *city*$(dest(p)) = c'\}$.*

Definition 3 (Truck Delivery Graph)**.** *For state s of logistics task $\langle L, C, P, T, A, city, airport, origin, dest\rangle$ and city $c \in C$, the* truck delivery graph for c *is the directed graph $D_s^c = (V, E)$, where*

- *$V = \{l \in L \mid$ city$(l) = c\}$ are the locations in city c, and*
- *E contains the following edges for each package p with $pos_s(p) \neq$ dest(p):*
  - *If city$(pos_s(p)) =$ city$(dest(p)) = c$ then there is an edge $pos_s(p) \to dest(p)$.*
  - *If city$(pos_s(p)) = c$, city$(dest(p)) \neq c$ and $pos_s(p) \neq$ airport(c) there is an edge $pos_s(p) \to$ airport(c).*
  - *If city$(pos_s(p)) \neq c$, city$(dest(p)) = c$ and $dest(p) \neq$ airport(c) there is an edge airport$(c) \to dest(p)$.*

Multi-vehicle Simplification

Our objective in this section is to prove the following theorem, which will help us optimally solve LOGISTICS tasks by reducing the number of possibilities we need to consider.

Theorem 1. *Every solvable* LOGISTICS *task has an optimal solution which only uses a single vehicle for each weakly connected component of each delivery graph for the initial state.*

For simplicity and brevity, we focus on the case where all delivery graphs are (essentially) weakly connected:

Theorem 2. *Consider a solvable* LOGISTICS *task where each delivery graph has exactly one non-trivial weakly connected component, i.e., one weakly connected component plus zero or more isolated locations.*[2] *Then there is an optimal plan using one truck from each city and one airplane.*

Once we have proved Theorem 2, it is not difficult to show Theorem 1 by observing that whenever there are multiple weakly connected components in a delivery graph, an optimal solution can be obtained by considering the subtasks for each component independently and combining their solutions. Hence, we focus on Theorem 2 from now on.

Theorem 2 does not immediately tell us *which* truck from each city and *which* airplane should be considered. However, just knowing that only one vehicle is needed in each delivery graph already significantly reduces the space of states that needs to be explored. To apply the theorem when searching for optimal solutions of LOGISTICS tasks, we exhaustively explore all possible choices of vehicles for each delivery graph, compute an optimal solution for the given set of choices, and then select the overall best one.

To limit the search space further, we impose additional restrictions on the vehicles we consider for a given delivery graph D. (It is easy to show that these restrictions preserve optimality.) We call a vehicle in D *useful* if its initial location has an outgoing edge in D, indicating that the first action that this vehicle performs in a plan might be a *load* action. If there are useful vehicles in D, then we only consider the useful vehicles. If there is a useful vehicle whose initial location has no ingoing edge in D we can remove all other vehicles from consideration. If no vehicle in D is useful, we can pick an arbitrary vehicle for D, as no matter which vehicle we choose, its first action must be a movement, giving no vehicle a benefit over another. Finally, if multiple vehicles in D have the same initial location, we only consider one of them, which suffices for symmetry reasons.

In the rest of this section, we describe a proof of Theorem 2. Due to space limitations, we only provide a sketch of the full argument. The full proof is available as a technical report (Paul et al. 2017).

[1] In contrast to a typical PDDL representation, we do not make the departure location in movement actions explicit because we are not restricted by the limitations of the STRIPS formalism.

[2] Isolated locations are ones where no package must be unloaded or loaded. They may serve as starting locations of vehicles, but are otherwise of no use.

No Unnecessary Load/Unload Actions

It is easy to see that in an optimal plan, a package is never transported by a truck within a city other than its city of origin or its destination city, that intracity packages are never transported by airplanes, and that intercity packages never re-enter their city of origin after they have first reached its airport or leave their destination city after they have first reached its airport. We now show that we can also assume that no reloading of packages between vehicles happen in any single "stage" of transportation (for example, transporting a package within its city of origin).

Lemma 1. *If the task is solvable, then there is an optimal plan that does not reload packages from truck to truck or airplane to airplane.*

Proof. We show the lemma for trucks; the same argument works for airplanes. Let π be a plan with a (usually non-contiguous) subsequence $\mathit{load}(t, p, l)$, $\mathit{unload}(t, p, l')$, $\mathit{load}(t', p, l')$, where t, t' are trucks, p is a package, l, l' are locations in the same city, and no load/unload actions for p happen in between these three actions. We modify π by removing the first pair of load/unload actions and replacing the action $\mathit{load}(t', p, l')$ by the "macro" $\mathit{drive}(t', l), \mathit{load}(t', p, l), \mathit{drive}(t', l')$. This removes two actions and adds two actions, so does not increase the plan cost. Repeat such replacements until no further replacements can be made. At this point, every package is only loaded/unloaded by a truck (at most) once per city. □

Together with the preceding comments, the lemma implies that we can restrict attention to plans where each package is loaded/unloaded the minimal required number of times. A plan can never be improved by adding load/unload actions to reduce the number of movement actions.

Partitioning to Subtasks

Next, we show that we can restrict attention to subtasks involving only airplanes or only trucks of one city. Consider an optimal plan π. Partition the plan into subsequences: one subplan π_c for each city c, containing all actions involving trucks in this city, and one subplan π_A for the airplanes, containing all actions involving airplanes. To prove Theorem 2, we must show that there is an optimal plan π where each subplan π_x (where x is a city or the set of airplanes A) only uses one vehicle.

We prove this by a *local replanning* argument. Assume we are given π such that some subplan π_x uses multiple vehicles. Then we modify *only* this subplan to form a new subplan π'_x that uses only one vehicle. We construct π'_x in such a way that it has the same cost as π_x and is *compatible* with the other subplans of π: the subplans of π, but replacing π_x with π'_x, can be interleaved to form a new valid global plan. Considering local subplans of this kind that can be pieced together to form a global plan is the essential idea of *factored planning* (Amir and Engelhardt 2003; Brafman and Domshlak 2006). Clearly, if we can show that this local replanning operation is always possible, Theorem 2 follows by a sequence of replanning steps, one for each subplan π_x using multiple vehicles.

A critical aspect in local replanning is understanding which constraints are imposed on π'_x by the other subplans. Firstly, π'_x must solve the local delivery task for the packages and vehicles in its delivery graph: for example, if there exists an intercity package to be transported from city c to city c', then π'_A must transport it from $\mathit{airport}(c)$ to $\mathit{airport}(c')$. This aspect of local replanning can be viewed as a regular LOGISTICS task, limited to the airports and airplanes (or alternatively to the locations and trucks of one city).

Secondly, going beyond a regular LOGISTICS task, π'_x must respect certain ordering constraints on the delivery of packages imposed by the other subplans. For example, if the subplan π_c (for city c) loads an incoming intercity package p from $\mathit{airport}(c)$ before it unloads an outgoing intercity package p' at $\mathit{airport}(c)$, then π'_A must unload p at $\mathit{airport}(c)$ before it loads p' at $\mathit{airport}(c)$. In such a situation, we say that π_c imposes the *precedence constraint* $p \prec p'$ on π'_A.

A key observation is that *all* relevant interactions between subplans can be captured by precedence constraints of the form $p \prec p'$, where p and p' are packages such that π_A must unload p at the same location where it must load p'. Intuitively, as long as each subplan delivers all packages it owes to the other subplans "in time", i.e., before loading packages that it receives from these subplans at a later stage in the original global plan π, combining the subplans into a global plan is guaranteed to succeed.

In summary, it remains to show that each *local task* can be optimally solved by using a single vehicle, where a local task is an airplane-only or truck-only LOGISTICS task with added precedence constraints of the form $p \prec p'$, expressing that package p must be dropped at its goal location before p' may be picked up at its initial location. Moreover, for all such precedence constraints, $\mathit{dest}(p) = \mathit{origin}(p')$ in the local task. (Note that it does not matter whether the vehicles in a local task are airplanes or trucks, so we do not need to consider two different "kinds" of local tasks.)

Replanning Local Tasks to Use a Single Vehicle

Assume we are given a local task Π_x, w.l.o.g. represented as a LOGISTICS tasks where all locations are airports and all vehicles are airplanes, along with a set of precedence constraints on package delivery. We are also given an optimal plan π_x for Π_x. Our aim is to construct another optimal plan π'_x for Π_x that only uses one airplane.

We first consider the case where all airplanes are initially located at isolated locations, i.e., locations that are not the origin or destination of any package in Π_x. We will consider the general case afterwards. In the restricted case, π'_x can be constructed from π_x as follows:

1. Select an arbitrary airplane v to use in π'_x.
2. Let *moves* be the subsequence of movement actions in π_x, i.e., π_x with load and unload actions removed.
3. Obtain a new sequence of movement actions *moves*$'$ from *moves* by moving to exactly the same sequence of locations as in *moves*, but using airplane v for all movements. In other words, v follows the movements of *all* airplanes in π_x, in the same order that the movements occur in π_x.

4. Compute π'_x from *moves'* by inserting load and unload actions for all packages at the appropriate times. It is sufficient to do this opportunistically, i.e., performing each action that loads a package from its origin or unloads it at its destination as soon as the action becomes applicable (taking into account the precedence constraints).

It is obvious that π'_x only uses one airplane and has the same cost as π_x (both plans have the same number of movement actions, and both have one load and unload action per package). It is less obvious that step 4. in the algorithm is always possible while satisfying all precedence constraints, and we refer the reader to the previously mentioned technical report for a complete formal treatment.

We remark that the restriction to the case where the airplanes begin at isolated locations is important for the construction to work. Without it, v might not have the opportunity to load packages in time (or at all) that are located at the location of origin of some airplane used in π_x.

We now consider the general case without this restriction. Let $\tilde{\Pi}_x$ be a modified task obtained from Π_x by changing the initial location of each airplane to a new isolated location $\tilde{l}$. Let V be the set of airplanes used in π_x. Then $\tilde{\Pi}_x$ can be solved by prefixing π_x by $|V|$ movements, flying each airplane in V from $\tilde{l}$ to its origin location in Π_x. We denote this plan for $\tilde{\Pi}_x$ by $\tilde{\pi}_x$. Its cost is $c^* + |V|$, where c^* is the cost of π_x. Because $\tilde{\Pi}_x$ satisfies the conditions of the restricted case (all airplanes originate at an isolated location), we can convert $\tilde{\pi}_x$ to a plan $\tilde{\pi}'_x$ of the same cost $c^* + |V|$ that only uses one airplane. Moreover, it is easy to see that $\tilde{\pi}'_x$ solves not just the modified task $\tilde{\Pi}_x$ but also the original task Π_x.

One problem remains: we need a single-airplane plan of cost c^*, but the cost of $\tilde{\pi}'_x$ is $c^*+|V|$. To remedy this, we permute the plan $\tilde{\pi}_x$ before converting it to single-airplane form in such a way that it contains $|V|-1$ occurrences where two subsequent movement actions (of different airplanes) move to the same location, i.e., $\mathit{fly}(v', l)$ and $\mathit{fly}(v'', l)$ occur next to each other in the plan. Intuitively, this is possible because with a weakly connected delivery graph, the path traced by each used airplane v' must eventually intersect with the path of another airplane. Until this point, the actions of v' can be commuted freely with the actions of the other airplanes, and this allows us to interleave the subplans for the different airplanes in such a way that v' reaches the location l where it joins the path of another airplane at the same time as another airplane v'' does. (Again, the full formal treatment is slightly more complex, and we refer to the technical report.)

With $|V|$ airplanes, there must be $|V|-1$ "join points" until the travel paths of all airplanes are connected. At each join point, the permuted plan $\tilde{\pi}_x$ contains consecutive actions of the form $\mathit{fly}(v', l)$, $\mathit{fly}(v'', l)$. In the single-airplane plan $\tilde{\pi}'_x$, these become consecutive movements $\mathit{fly}(v, l)$, $\mathit{fly}(v, l)$, where the second movement is clearly redundant and can be omitted. Altogether, this argument permits us to save $|V|-1$ of the $|V|$ extraneous actions.

To save the remaining action, we observe that by construction, the first action in $\tilde{\pi}'_x$ flies (from the artificially introduced isolated location $\tilde{l}$) to the origin location in Π_x of some airplane $v \in V$. By choosing this airplane as the single airplane we use in the plan $\tilde{\pi}'_x$, we can save this action and reduce the cost of $\tilde{\pi}'_x$ to c^*, concluding the proof.

Search Space Pruning

To reduce the size of the search space, we apply two pruning techniques: *operator elimination* removes a large number of operators from consideration that are never part of an optimal plan. *Instant operator application* can be seen as a form of partial-order reduction or meta actions.

From Lemma 1, we know that it is never necessary to reload packages between vehicles of the same type. The following operators can hence be ignored:

- For intracity packages p
  - all operators $\mathit{load}(v, p, l)$ where v is an airplane or $l \neq \mathit{origin}(p)$, and
  - all operators $\mathit{unload}(v, p, l)$ where v is an airplane or $l \neq \mathit{dest}(p)$.
- For intercity packages p with $\mathit{city}(\mathit{origin}(p)) = c$ and $\mathit{city}(\mathit{dest}(p)) = d$
  - all operators $\mathit{load}(v, p, l)$ where v is an airplane and $l \neq \mathit{airport}(c)$,
  - all operators $\mathit{unload}(v, p, l)$ where v is an airplane and $l \neq \mathit{airport}(d)$,
  - all operators $\mathit{load}(v, p, l)$ where v is a truck except those where $l = \mathit{airport}(d) \neq \mathit{dest}(p)$ or $l = \mathit{origin}(p)$,
  - all operators $\mathit{unload}(v, p, l)$ where v is a truck except those where $l = \mathit{airport}(c) \neq \mathit{origin}(p)$ or $l = \mathit{dest}(p) \neq \mathit{airport}(d)$.

After the elimination of unnecessary operators, packages can always be unloaded immediately, if such an operator becomes applicable. Moreover, if there is only one truck in a city or there is only one airplane then packages can be loaded whenever such an operator becomes applicable. This optimization does not threaten the optimality guarantee of A* because whenever a state has been reached by a prefix of an optimal plan, then it is possible to extend this prefix to an optimal plan, continuing with these operators.

Heuristics

The key to efficient A^* search is a strong admissible heuristic that, from any state in the search, accurately estimates the cost of reaching a goal state.

Counting Heuristic

The simplest heuristic we consider is the *single visit and load/unload counting* heuristic, h_0. It includes estimates for the number of vehicle movements and estimates for the number of applications of load and unload operators.

It is easy to see that if the position of a package is in the same city as its destination location a package must be

- unloaded from a plane iff it is in a plane,
- loaded into a truck iff its position is not the destination location and it is not in a truck, and

- unloaded from a truck iff its position is not the destination location or it is in a truck.

Similarly, if the position of a package is *not* in the same city as its destination a package must be

- loaded into a truck at its current position iff its position is not an airport and it is not in a truck,
- unloaded from a truck at the airport of the current city iff its position is not an airport or it is in a truck,
- loaded into a plane iff it is not in a plane,
- unloaded from a plane,
- loaded into a truck at the airport of the destination city iff its destination location is not an airport, and
- unloaded from a truck in the destination city iff the destination location is not an airport.

The load/unload contributions to the heuristic estimate are exact; in an optimal solution, packages should not be reloaded between planes or between trucks in the same city.

Each location that a *truck* must visit to load or unload a package contributes a value of one to the counting heuristic. This is the case when a package must be brought to this location or when it must be collected from the location but there is currently no vehicle there. These locations can easily be determined from the truck delivery graphs:

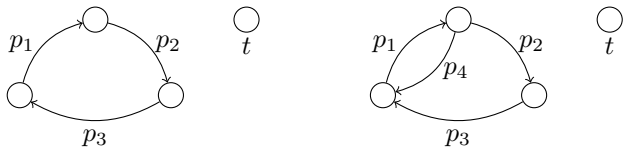
Definition 4 (Truck Landmark)**.** *For* LOGISTICS *task* $\langle L, C, P, T, A, city, airport, origin, dest\rangle$*, state* s *and city* $c \in C$*, the set* L_c^{truck} *of truck landmarks consists of the locations* l *that have an ingoing edge in the truck delivery graph* D_s^c *or that have an outgoing edge and there is no* $t \in T$ *with* $s(t) = l$*.*

Analogously, we can define a set of airplane landmarks for the cities that must be visited by an airplane:

Definition 5 (Airplane Landmark)**.** *For* LOGISTICS *task* $\langle L, C, P, T, A, city, airport, origin, dest\rangle$ *and state* s *the set* L^{airplane} *of airplane landmarks consists of the cities* c *that have an ingoing edge in the airplane delivery graph* D_s^a *or that have an outgoing edge and there is no* $a \in A$ *with* $s(a) = airport(c)$*.*

The counting heuristic accounts $\sum_{c \in C} |L_c^{\text{truck}}| + |L^{\text{airplane}}|$ for the movements of vehicles.

Despite its simplicity, the quality of the counting heuristic compares favorably with heuristics typically used for domain-independent planning. In particular, it is not difficult to see that for LOGISTICS tasks with one vehicle per weakly connected component of each region, such as the tasks generated by the multi-vehicle simplification, it is equal to the h^+ heuristic, which is known to be very accurate for LOGISTICS tasks compared to other domain-independent planning heuristics (Helmert and Mattmüller 2008). For LOGISTICS tasks with multiple vehicles per weakly connected component, the counting heuristic may slightly underestimate h^+; however, h^+ is known to be NP-hard to compute for arbitrary LOGISTICS states (Betz and Helmert 2009).



(a) One cycle (b) Two cycles

Figure 1: Example tasks: in both tasks it is necessary and sufficient to visit one of the locations twice.

Cycle Heuristic

The estimate of the single visit and load/unload counting heuristic is extremely optimistic. It does not account for the fact that some locations must be visited more than once. We illustrate this in Fig. 1a that shows a single city with four locations. There are three packages, where the edges indicate the origin and the destination location. The truck is located at an additional location. The edges in this graph form a cycle and for all packages in this cycle to be delivered to their goal locations, the truck in the city must visit one of the locations twice.

To derive a lower bound on the number of such locations that must be visited more than once, we analyze dependencies between the landmarks. For this purpose, we associate each truck landmark l with the operators set $\{drive(t, l) \mid t \in T\}$ and each airplane landmark c with the operator set $\{fly(a, l) \mid a \in A, airport(c) = l\}$. These are *disjunctive action landmarks* that encode that at some point in each plan an operator from this set must be applied. We take in addition *orderings* between these action landmarks into account:

Definition 6 (Landmark Ordering)**.** *For two disjunctive action landmarks* L *and* L'*, there is an ordering* $L \to L'$ *if in each plan the first application of an operator from* L *must happen before the last application of an operator from* L'*.*

This is a new notion of landmark orderings which is different from earlier such notions such as necessary or natural orderings (Hoffmann, Porteous, and Sebastia 2004; Richter and Westphal 2010). It is easy to see that orderings between two truck landmarks or two airplane landmarks can be derived from the transportation graphs: there is an ordering if there is a package that must be moved between the locations and there is no suitable vehicle at the package position. The following definition of landmark graphs captures this information for individual cities and the air space:

Definition 7 (Landmark Graph of City and Air Space)**.** *For state* s *of task* $\langle L, C, P, T, A, city, airport, origin, dest\rangle$ *and city* $c \in C$*, the* landmark graph *for* c *is the directed graph* $G_{c,s}^{\text{LM}} = (L_c^{\text{truck}}, E)$*, where* E *contains an edge* $l \to l'$ *if the delivery graph* D_s^c *contains such an edge and there is no* $t \in T$ *with* $s(t) = l$*.*

The landmark graph $G_{\text{air},s}^{\text{LM}}$ *for the air space is the directed graph* (L^{airplane}, E)*, where* E *contains an edge* $c \to c'$ *if the airplane delivery graph* D_s^A *contains such an edge and there is no* $a \in A$ *with* $city(s(a)) = c$*.*

It would not be admissible to just count the number of cycles in the landmark graphs: consider the example in Fig.

1b which shows a variation of the earlier example with one additional package. In this example, the landmark graph for the city corresponds to the non-trivial connected component in the given graph, which contains two cycles. Still, it is sufficient to only visit the origin location of package p_4 twice.

However, we can derive an admissible estimate from the size, *MFVS*, of a *minimum feedback vertex set*, which is a set of vertices of minimum size whose removal renders the graph acyclic. It is NP-hard to determine *MFVS* for a graph (Karp 1972), but we will show in the experiments that the resulting heuristic still is practically feasible.

Theorem 3. *Let π be a plan for state s and let $G = (V, E)$ be a landmark graph encoding orderings between landmarks. Then there are at least MFVS(G) different landmarks in V for which π contains at least two applications of operators from the associated disjunctive action landmark.*

Proof. Let $V' \subseteq V$ be the landmarks for which π contains more than one operator application and assume $|V'| <$ *MFVS*(G). Then the subgraph of G induced by $V \setminus V'$ contains a simple cycle $Y = (L_0, L_1, \ldots, L_n)$ with $n > 1$ and $L_0 = L_n$. For $i \in \{0, \ldots, n-1\}$, each plan must apply an operator from the set associated with L_{i+1} after it has already applied an operator from the operator set for L_i. Therefore there are at least $n + 1$ operator applications for these n sets and π contains two operator applications for one of the sets. As none of the L_i can be in V' this is a contradiction to the definition of V'. □

Note that the disjunctive action landmarks for two truck or airplane landmarks are disjoint. This allows us to admissibly improve the estimate of the baseline heuristic h_0.

Definition 8. *For a task with set of cities C, the* cycle heuristic h_{cycle} *for state s is defined as*

$$h_{cycle}(s) = h_0(s) + \mathit{MFVS}(G^{\text{LM}}_{\text{air},s}) + \sum_{c \in C} \mathit{MFVS}(G^{\text{LM}}_{c,s}).$$

Theorem 4. *The cycle heuristic is admissible.*

Proof. Heuristic h_0 only accounts for load and unload operations and at most one drive and one fly operation to each location. From Theorem 3 and the fact that the action landmarks associated with the truck or airplane landmarks are disjoint, we know that every plan must contain at least *MFVS*$(G^{\text{LM}}_{\text{air},s})$ additional fly and $\sum_{c \in C} \mathit{MFVS}(G^{\text{LM}}_{c,s})$ additional drive operators. □

Integrated Cycle Heuristic

Up to this point, we have considered the regions separately. However, this way we miss some orderings that can only be derived when considering the transportation of a package with vehicles of different types.

Each package induces up to six disjunctive action landmarks (truck movement to package position and start city airport, plane movement to start and destination city airport, truck movement to destination city airport and package destination). From the 30 possible pairs of these landmarks only nine can define valid orderings. Three of them are already covered by the landmark graphs of the cities and the airspace. We introduce the remaining six as part of the following definition:

Definition 9 (Integrated Landmark Graph)**.** *For state s of task $\langle L, C, P, T, A, \mathit{city}, \mathit{airport}, \mathit{origin}, \mathit{dest}\rangle$, the* integrated landmark graph $G^{\text{LM}}_s = (V, E)$ *is the directed graph, where*

- $V = L^{\text{airplane}} \cup \bigcup_{c \in C} L^{\text{truck}}_c$ *consists of all truck and airplane landmarks, and*
- E *contains all edges from the landmark graphs for all cities and the air space plus the following edges for each package p with $\mathit{city}(\mathit{pos}_s(p)) = c$ and $\mathit{city}(\mathit{dest}(p)) = d \neq c$:*
  - *if there is no $t \in T$ with $s(t) = \mathit{pos}_s(p)$ and neither $\mathit{pos}_s(p)$ nor $\mathit{dest}(p)$ is an airport, there is an edge $\mathit{pos}_s(p) \to \mathit{dest}(p)$;*
  - *if there is no $t \in T$ with $s(t) = \mathit{pos}_s(p)$ and $\mathit{pos}_s(p)$ is not an airport, there is an edge $\mathit{pos}_s(p) \to d$;*
  - *if neither $\mathit{pos}_s(p)$ nor $\mathit{dest}(p)$ is an airport, there is an edge $\mathit{airport}(c) \to \mathit{dest}(p)$;*
  - *if $\mathit{pos}_s(p)$ is not an airport, there is an edge $\mathit{airport}(c) \to d$;*
  - *if there is no $a \in A$ with $s(a) = \mathit{airport}(c)$ and $\mathit{dest}(p)$ is not an airport, there is an edge $c \to \mathit{dest}(p)$;*
  - *if $\mathit{dest}(p)$ is not an airport, there is an edge $d \to \mathit{dest}(p)$.*

It is easy to verify that the edges correspond to landmark orderings, keeping in mind that the city nodes are associated with fly operators and the locations with drive operators.

Therefore, we can again exploit Theorem 3 to admissibly improve the estimate of the cycle heuristic h_{cycle}.

Definition 10. *The* integrated cycle heuristic h_{ic} *for state s is defined as*

$$h_{ic}(s) = h_0(s) + \mathit{MFVS}(G^{\text{LM}}_s).$$

Theorem 5. *h_{ic} dominates h_{cycle}, i. e., $h_{ic} \geq h_{cycle}$.*

Proof. As the size of an MFVS of a graph is equal to the sum of the sizes of the MFVSs of its connected components, the only difference between h_{ic} and h_{cycle} are the additional arcs. Since additional arcs can only cause additional cycles (and not remove any), the minimum feedback vertex set can only be larger, and hence $h_{ic}(s) \geq h_{cycle}(s)$ for all states s. □

Theorem 6. *The integrated cycle heuristic is admissible.*

Proof. As before, heuristic h_0 only accounts for load and unload operations and at most one drive and one fly operation to each location. The integrated cycle heuristic adds the minimal number of move operations that are required to satisfy cyclic ordering constraints of the disjunctive action landmarks, which is an admissible estimate. □

Experimental Evaluation

We have performed an evaluation of our algorithm on an Intel core i3 4160 processor running at 3.60 GHz with a limit of one million evaluated states.

Table 1 shows the results for the 35 LOGISTICS instances of the International Planning Competition (IPC) 1998 and

		all vehicles				single vehicle			
		h_0		h_{ic}		h_0		h_{ic}	
inst.	h^*	Δ	states	Δ	states	Δ	states	Δ	states
01	26	1	131	0	48	1	16	0	16
02	32	0	177	0	177	0	33	0	33
03	54	0	354	0	354	0	43	0	43
04	58	0	532	0	532	0	57	0	57
05	22	0		0	45	0	18	0	18
06	69	0	288	0	472	1	1145	0	94
07	33	0	267	0	288	0	30	0	30
08	40	0	267	0	267	0	36	0	36
09	80	1		0		1	967	0	69
10	101	1		1		1	141	0	141
11	29	1	1196	1	1196	0	15	0	17
12	41	0	440	0	440	0	45	0	45
13	67	0	1192	0	1192	0	72	0	72
14	86	1		0		0	160	0	160
15	87	2		1		2	821	0	46
16	53	1	70393	1	70393	0	26	0	26
17	42	3	129289	3	129289	0	17	0	17
18	161	3		2		1	235085	0	271
19	135	3		2		3	3196	0	109
20	135	3		2		3	47579	0	124
21	99	2		1		1	6902	0	98
22	264	6		4		5		0	499
23	106	0		0		0	1276	0	138
24	40	1	194208	1	194208	0	25	0	25
25	180	1		1		0	245	0	245
26	183	0		0		0	558	0	558
27	136	1		1		0	113	0	113
28	251	2		0		2		0	304
29	295	9		4		6		0	632
30	128	1		1		0	157	0	157
31	13	0	17	0	17	0	9	0	9
32	20	0	35	0	35	0	11	0	11
33	27	1	353	0	66	1	29	0	16
34	45	1	1673	0	433	1	102	0	31
35	30	0	85	0	85	0	26	0	26

		all vehicles				single vehicles			
		h_0		h_{ic}		h_0		h_{ic}	
inst.	h^*	Δ	states	Δ	states	Δ	states	Δ	states
20-0	107	1		0	729	1	293	0	62
25-0	143	1		0	486182	1	153	0	83
30-0	175	3		1		3	1012	0	107
35-0	177	2		0	214062	2	5058	0	144
36-0	192	3		1		3	15719	0	171
37-0	223	5		2		4	28561	0	171
38-0	209	2		1		2	2195	0	150
39-0	224	4		2		4	139347	0	177
40-0	228	3		0		3	170652	0	208
50-0	286	3		2		3	191249	0	266
60-0	369	6		3		6		0	379
70-0	405	4		0		4		0	969
80-0	476	6		2		6		0	1262
90-0	513	5		3		5		0	679
90-1	529	6		1		1		0	712
91-0	534	6		4		6		0	1487
91-1	555	11		3		7		0	745
92-0	539	8		3		8		0	751
92-1	532	7		4		7		0	770
93-0	556	7		4		8		0	822
93-1	532	5		2		5		0	1420
94-0	550	6		2		6		0	787
94-1	554	7		2		7		0	757
95-0	579	7		1		7		0	827
95-1	564	9		3		8		0	785
96-0	577	7		3		7		0	1646
96-1	568	6		2		6		0	1594
97-0	563	8		4		7		0	819
97-1	556	8		4		8		0	853
98-0	581	7		3		7		0	830
98-1	539	7		3		7		0	824
99-0	588	8		3		8		0	1806
99-1	581	8		3		8		0	878
100-0	590	7		2		7		0	1787
100-1	589	6		4		7		0	887

Table 1: Results for LOGISTICS problems of IPC 1998 (left) and a subset of the IPC 2000 instances (right). The first column gives the problem number, followed by the optimal cost. The next four sections, consisting of two columns each, show the results without and with multi-vehicle simplification for h_0 and h_{ic}. Δ is the difference to h^* and *states* is the number of evaluated states. Blank entries represent instances that were not solved within the bound of 1 million evaluations.

a representative subset of the largest instances of the IPC 2000 Track 2 Additional benchmarks suite (which consists of 170 instances). The IPC 1998 instances contain a wide range of package quantities (4–57), number of cities (3–47), sizes of cities (1–16), number of trucks (5–106) and number of planes (1–15). The IPC 2000 instances contain up to 100 packages that are distributed among up to 34 cities and can be transported by up to nine planes, but all have in common that there are only one truck and two locations (a non-airport and an airport location) per city. The table shows results for h_0 and h_{ic}, both with and without multi-vehicle simplification. For each configuration, we report the difference between the optimal plan length (h^*) and the initial heuristic estimate as $\Delta = h^* - h(s_0)$, where h is the corresponding heuristic. We furthermore include the number of evaluated states, which may be smaller than the optimal plan length due to instant operator applications. With multi-vehicle simplification, we use an exhaustive search over the possible choices of vehicles, skipping a sub-search if the f-values already prove that it will not improve the currently best solution. The number of states reports the sum over all these searches. It can clearly be seen that considering only a single vehicle and using a more sophisticated heuristic significantly improve the baseline configuration.

Although the table only shows as subset of the IPC 2000 instances, with our best configuration (h_{ic} with multi-vehicle simplification) we are able to solve *all* LOGISTICS instances of both IPCs optimally, and are, to the best of our knowledge, the first to report plan lengths for all considered IPC instances. As a point of comparison, we performed exper-

iments with the winner of the IPC 2014 sequential optimization track, the symbolic search planner SymBA* (Torralba, Linares López, and Borrajo 2016) and Fast Downward (Helmert 2006) equipped with three heuristics from the literature: LM-cut (Helmert and Domshlak 2009), the state-equation heuristic (Bonet 2013), and merge-and-shrink using bisimulation and the DFP merge strategy (Helmert et al. 2014; Sievers, Wehrle, and Helmert 2014). Most of the considered instances (as well as several of the smaller IPC 2000 Standard instances) cannot be solved by any of these state-of-the-art domain-independent planning systems.

Betz and Helmert (2009) evaluated the performance of the h^+ heuristic on LOGISTICS with a domain-specific heuristic implementation. Betz (2009) reports 7 solved instances for this approach on the IPC 1998 instances. A set of *sub-optimal* domain-specific approaches was compared in the hand-tailored track of IPC 2000, with TALPlanner (Doherty and Kvarnström 2001) being the clear winner on the domain. The system solved all of the IPC 2000 Track 2 additional instances very fast, but none of the instances reported in Table 1 optimally (with plan lengths varying between 3.7% and 9.5% longer than the optimal solution).

Table 2 shows results for a set of 26 randomly generated instances with 6–34 cities, each containing five non-airport and one airport location. There is only one plane and each city is assigned one truck. The larger city size allows for the presence of more complex graph cycles, and the assignment of trucks and planes removes the difficulty associated with multiple trucks per city and multiple planes. As there is only one vehicle per region, we can concentrate on the comparison of h_0, h_{cycle} and h_{ic}. Both cycle-based heuristics show a clear advantage over the baseline heuristic. Comparing h_{cycle} and h_{ic}, the integration of the individual landmarks graphs results in an impressive improvement of heuristic accuracy, with a reduction of evaluated states of up to three orders of magnitude and five additional solved instances.

Data on computation time is omitted in both tables for space reasons, but can be summarized briefly: most instances are solved in less than a second or not solved within the state budget of one million evaluated states at all, with no measurable difference between the different configurations. The few exceptions correspond to those instances where the number of evaluated states is significantly larger.

		h_0		h_{cycle}		h_{ic}	
inst.	h^*	Δ	states	Δ	states	Δ	states
16-6	107	4	3453	0	89	0	89
20-7	127	6		1	4275	0	130
25-9	173	3	2400	0	258	0	258
30-10	204	5	122678	0	177	0	177
31-10	212	7		0	11953	0	249
32-10	215	5	224758	0	381	0	194
33-11	226	9		0	134024	0	300
34-12	232	9		0	14383	0	323
35-12	237	8		0	3939	0	296
36-12	249	10		1	357764	0	322
37-13	239	7		0	1865	0	239
38-13	250	8		0	489	0	272
39-13	255	9		0	591	0	591
40-14	278	10		1		1	23776
45-15							
50-17	369	12		0		0	566
55-17	392	14		0		0	699
60-20	417	11		0		0	649
65-22	454	12		0	61282	0	1112
70-24							
75-25							
80-27							
85-29	598	19		0	1422	0	1422
90-30	641	22		1		0	1541
95-32							
100-34							

Table 2: Results for randomly generated problems. Instance x-y contains x packages and y cities. The second column gives the optimal cost. There are three sections for h_0, h_{cycle}, and h_{ic}, analogously to Table 1.

Discussion and Future Work

We have combined three techniques to efficiently solve large LOGISTICS problems optimally: multi-vehicle simplification, search space pruning and strong admissible heuristics. These techniques can be applied independently, but their impact is not independent because search space pruning and the heuristics benefit from multi-vehicle simplification.

The instant application of load operators requires that there is only one vehicle in the relevant region, which is almost always the case with multi-vehicle simplification. A possible additional optimization would eliminate all operators where a vehicle operates on a connected component of the delivery graph for which it is not "responsible". We excluded this pruning from consideration because it is only applicable with multi-vehicle simplification.

A large number of vehicles also has a negative impact on the presented heuristics because vehicles can reduce the number of landmarks: an outgoing edge in the delivery graph only justifies a landmark if there is no vehicle at this location. For the cycle-based heuristics, this also means that the minimum feedback vertex set is potentially smaller.

Efficiently solving LOGISTICS tasks optimally is already a contribution in itself, and we compute for the first time optimal plans for all LOGISTICS tasks used in the International Planning Competitions. However, a core question for future work will be what aspects of the paper can be generalized beyond LOGISTICS. In our opinion, the cycle heuristic is the contribution that looks most promising for such a generalization. Similar ideas have been presented for Blocksworld (Slaney 2014), using hitting sets, and solitaire games (Paul and Helmert 2016). Indeed the heuristic by Paul and Helmert can be seen as a special case of the cycle heuristic but it has not been formulated in terms of disjunctive action landmarks and landmark orderings. This new perspective makes the underlying idea much more accessible and it is now much clearer how it could be applied to domain-independent heuristic search.

Acknowledgments

This work was supported by the Swiss National Science Foundation (SNSF) as part of the project "Reasoning

about Plans and Heuristics for Planning and Combinatorial Search" (RAPAHPACS).

References

Amir, E., and Engelhardt, B. 2003. Factored planning. In Gottlob, G., and Walsh, T., eds., *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 929–935. Morgan Kaufmann.

Bacchus, F. 2001. The AIPS'00 planning competition. *AI Magazine* 22(3):47–56.

Betz, C., and Helmert, M. 2009. Planning with h^+ in theory and practice. In Mertsching, B.; Hund, M.; and Aziz, Z., eds., *Proceedings of the 32nd Annual German Conference on Artificial Intelligence (KI 2009)*, volume 5803 of *Lecture Notes in Artificial Intelligence*, 9–16. Springer-Verlag.

Betz, C. 2009. Komplexität und Berechnung der h^+-Heuristik. Diplomarbeit, Albert-Ludwigs-Universität Freiburg.

Bonet, B. 2013. An admissible heuristic for SAS$^+$ planning obtained from the state equation. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2268–2274. AAAI Press.

Brafman, R. I., and Domshlak, C. 2006. Factored planning: How, when and when not. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 2006)*, 809–814. AAAI Press.

Doherty, P., and Kvarnström, J. 2001. TALplanner: A temporal logic based planner. *AI Magazine* 22(3):95–102.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Helmert, M., and Mattmüller, R. 2008. Accuracy of admissible heuristic functions in selected planning domains. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 938–943. AAAI Press.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM* 61(3):16:1–63.

Helmert, M.; Mattmüller, R.; and Röger, G. 2006. Approximation properties of planning benchmarks. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, 585–589.

Helmert, M. 2003. Complexity results for standard benchmark domains in planning. *Artificial Intelligence* 143(2):219–262.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Helmert, M. 2008. *Understanding Planning Tasks – Domain Complexity and Heuristic Decomposition*, volume 4929 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22:215–278.

Karp, R. M. 1972. Reducibility among combinatorial problems. In Miller, R. E., and Thatcher, J. W., eds., *Complexity of Computer Computations*. Plenum Press. 85–103.

McDermott, D. 2000. The 1998 AI Planning Systems competition. *AI Magazine* 21(2):35–55.

Paul, G., and Helmert, M. 2016. Optimal solitaire game solutions using A* search and deadlock analysis. In Baier, J. A., and Botea, A., eds., *Proceedings of the Ninth Annual Symposium on Combinatorial Search (SoCS 2016)*, 135–136. AAAI Press.

Paul, G.; Röger, G.; Keller, T.; and Helmert, M. 2017. Optimal solutions to large logistics planning domain problems – detailed proofs. Technical Report CS-2017-001, University of Basel, Department of Mathematics and Computer Science.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized label reduction for merge-and-shrink heuristics. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, 2358–2366. AAAI Press.

Slaney, J. 2014. Set-theoretic duality: A fundamental feature of combinatorial optimisation. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, 843–848. IOS Press.

Torralba, Á.; Linares López, C.; and Borrajo, D. 2016. Abstraction heuristics for symbolic bidirectional search. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3272–3278. AAAI Press.