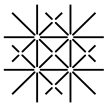

University
of Basel

Time Unrolling Heuristics

Master Thesis

Faculty of Science
Department of Mathematics and Computer Science
Artificial Intelligence
http://ai.cs.unibas.ch/

Examiner: Prof. Dr. Malte Helmert
Supervisor: Florian Pommerening

Philipp Oldenburg
philipp.oldenburg@stud.unibas.ch
13-061-064

03.03.2018

Contents

Acknowledgments

First I would like to thank Florian Pommerening for his constructive feedback. Without his feedback this thesis would not be at the current level of quality. Secondly I like to thank my friends Simon Wallny and Marc Schäfer for a multitude of insightful discussions and my family, especially my mom, for supporting me during the last six months. Further credit is due to Professor Malte Helmert for offering me to do my Master Thesis in the AI research group of the University of Basel. Last but not least special thanks goes to sciCORE – the center for scientific computing at the University of Basel – for letting me conduct experiments on their grid.

Abstract

Estimating cheapest plan costs with the help of network flows is an established technique. Plans and network flows are already very similar, however network flows can differ from plans in the presence of cycles. If a transition system contains cycles, flows might be composed of multiple disconnected parts. This discrepancy can make the cheapest plan estimation worse.
One idea to get rid of the cycles works by introducing time steps. For every time step the states of a transition system are copied. Transitions will be changed, so that they connect states only with states of the next time step, which ensures that there are no cycles.
It turned out, that by applying this idea to multiple transitions systems, network flows of the individual transition systems can be synchronized via the time steps to get a new kind of heuristic, that will also be discussed in this thesis.

1 Introduction

Classical planning problems can be solved via the A^* algorithm with a heuristic as guidance. A certain type of heuristic is the flow heuristic (e.g. van den Briel et al. (2007)), which tries to estimate cheapest goal distances via the cheapest flow. Flows can be thought of as ways to transport one unit from the initial state to goal states. Usually flow heuristics can not work directly on the transition system of the planning task, as they are usually too big. For this reason they are often combined with an abstraction which is able to reduce the transition system size, while still preserving all plans. The estimation from cheapest flow to cheapest plan works because a flow can be seen as a relaxed version of plan, which in turn also means that the flow heuristic is admissible. The reason why a flow can be seen as a relaxed plan is that for every plan there exists a flow, however the other direction is not given in general. If we would want to have the perfect estimation, we would enforce that for every flow there is also a plan, or if we just want to make the correspondence stricter, we could eliminate the cases where flows do not correspond to plan. This can either be achieved by altering the definition of flows to make them more like plans or we restrict the cases where they behave differently than plans. The approach that we will take is the later one.
Consider the abstract transition system $\mathcal{T}_1$ in Figure 1 (note that the characters are operators and the numbers are flow values along the transitions). It is obvious that there is no plan that corresponds to this flow, because the flow that goes along the self loop is isolated. Now this might not seem like a problem, because we are searching for the cheapest flows which do not have flow values along cycles.
This however becomes meaningful when we want to combine the information of multiple abstract transition systems, which can be done by finding flows for the individual transition systems which share the same summed flow for each operator. Consider a second abstract transition system $\mathcal{T}_2$ in Figure 1. We can see that there is a cheap synchronized flow that exploits the fact that there is a cycle in $\mathcal{T}_1$, which then in turn leads to a synchronized flow that does not correspond to any plan because of the isolated flow along the self loop in $\mathcal{T}_1$.

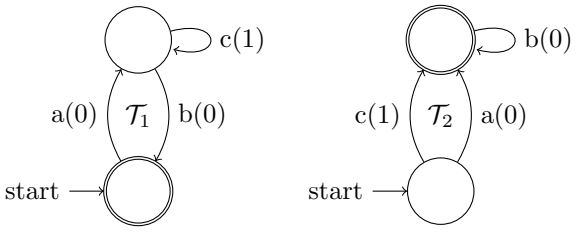

Figure 1: Depicts two abstract transition systems and a synchronized flow which does not correspond to a plan due to the isolated flow along the self loop in $\mathcal{T}_1$.

The idea that is presented in this thesis is based on introducing time steps to the transition systems to remove cycles, to avoid that they are exploited by synchronized flows. For a time unrolling of a transition systems the states are copied, so that every original state is present in every time point. The transitions now move along in time connecting the states they previously connected. Figure

2 shows the resulting transition systems when introducing 2 time steps. There is no cycle anymore that can be exploited, which leads to a cheapest synchronized flow with higher cost (also depicted in Figure 2), furthermore for the cheapest synchronized flow there might now be a corresponding plan of same cost (which means that it could lead to a perfect cheapest plan cost estimation).

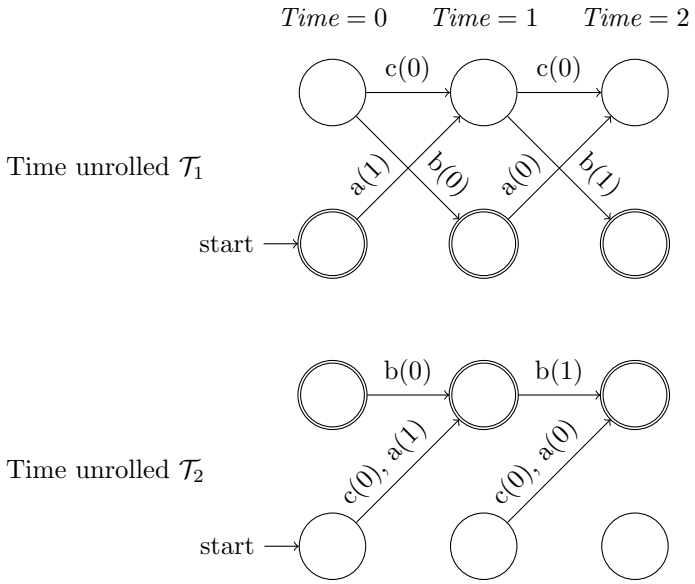

Figure 2: Depicts time unrolled transition systems. All cycles are gone which results in more expensive cheapest synchronized flow.

Normally a synchronized flow synchronizes the summed flow for each operator. But if we look at the figure above it appears to be natural to synchronize the summed flow for each operator *for every time step*. This is in fact possible without losing the connection that there is a synchronized flow for every plan of same cost and also might lead to better heuristic estimates as synchronizing for each time step is stricter than synchronizing over all time steps.
There is however a disadvantage of the time unrolling. In our example we have unrolled with 2 time steps. The resulting transition system only contain plans that are at most of length 2, because transitions move in time. This holds the danger that the cheapest plan of the original transition is no longer contained which in turn might make the heuristical guidance not admissible anymore. This problem can be tackled by introducing a new type of time unrolling, that copies transitions of the original transitions (cycles included) into the last time layer. This technique is depicted in Figure 3. Note how we are again introducing cycles, however the cheapest synchronized flow stays the same, but only if we synchronize the summed flows for every time step.
In this thesis we will introduce heuristics that use the time unrolling with and without repetition and combine it with searching for cheapest flows that are synchronized for each time step. We will conduct experiments of the postulated heuristics for the case where the abstractions are the atomic projections.

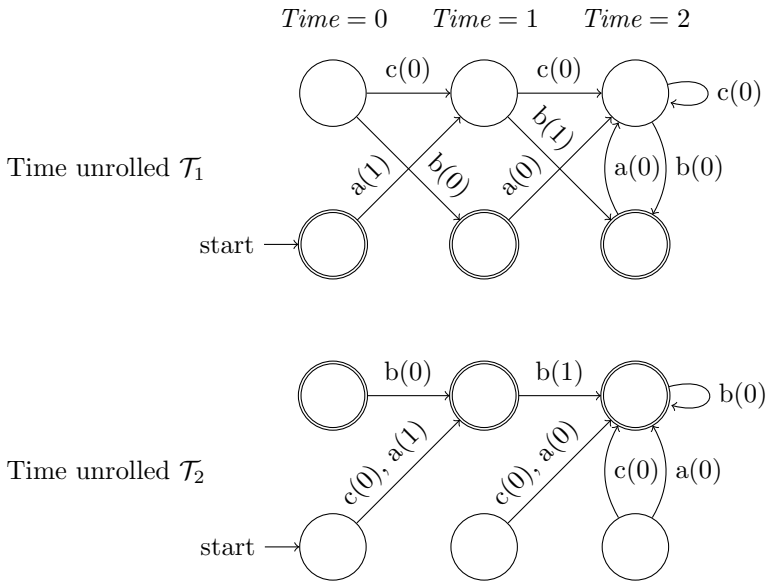

Figure 3: Depicts time unrolled transition systems with repetitions in the last time layer. Cycles got introduced however without getting exploited if the summed flows are synchronized for each time step.

2 Background

We work with SAS^+ tasks, that are in transition normal form (TNF) (Pommerening and Helmert, 2015). It is possible to efficiently transform every SAS^+ task into one which is equivalent and in TNF. A *planning task* Π is a tuple $\langle \mathcal{V}, \mathcal{O}, s_I, s_G, cost \rangle$, where $\mathcal{V}$ is a set of *variables* and each variable $v \in \mathcal{V}$ has its *domain* D_v. A *fact* is a pair, where the first element is a variable and the second element is one of the variable's domain values. *States* are set of facts with the condition that every variable occurs exactly once. *Partial states* are subsets of states. $vars$ is a function that maps a partial state s to a set of all first elements of the tuples in s. Whereas $s(var)$ yields the second element of the tuple in s, that has var as its first element. Let s' be another partial state, then $s \subseteq s'$ if $s(var) = s'(var)$ for every $var \in vars(s)$. $\mathcal{O}$ is a set of *operators* and each operator $o \in \mathcal{O}$ has a set of *preconditions* $pre(o)$ and a set of *effects* $eff(o)$, both being partial states. An operator o is *applicable* in a state s if $pre(o) \subseteq s$. Applying operator o in state s leads to a state s', written $o(s) = s'$, where $eff(o) \subseteq s'$ and s' agrees with s on all variables that are not in $vars(eff(o))$. As already mentioned we will consider only SAS$^+$ tasks, that are in TNF, i.e. $vars(eff(o)) = vars(pre(o))$ for every operator o and the goal state s_G is, as well as the initial state s_I, an actual state (not a partial state). The function cost maps every $o \in \mathcal{O}$ to a non-negative real number. Π_s denotes $\langle \mathcal{V}, \mathcal{O}, s, s_G, cost \rangle$ for every state s of Π.
The *transition system* $\mathcal{T}$ induced by planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_I, s_G, cost \rangle$ is a tuple $\langle \mathcal{S}, \mathcal{O}, cost, T, s_I, S_G \rangle$. $\mathcal{S}$ is the set of possible states, i.e. one state for every valuation of the variables in $\mathcal{V}$ (according to the domain of each variable). The set T contains a transition $\langle s, o, s' \rangle$ for all $s, s' \in S$ if o is applicable in s and $o(s) = s'$. The set S_G contains the goal states, i.e. only s_G. A *path* is a tuple of operators $\langle o_0, \ldots, o_l \rangle$ that can be applied in order to the initial state so that in every step the respective operator is applicable. $\mathcal{T}_s$ denotes the transition system induced by Π_s. A path induces the following *state path* $\langle s_I, o_0(s_I), \ldots, o_l(\ldots o_0((s_I)\ldots) \rangle$. *Plans* are paths that have to end in the goal state after applying the last operator. The cost of a path is the sum of the individual operator costs. For all states $s \in \mathcal{S}$ we define the length of the shortest paths to s as $d^*(s)$, if there is no path to s then $d^*(s) = \infty$. A state s is called dead if it cannot be reached from the initial state or the goal state cannot be reached from s.
To see the concepts used on an specific problem instance we use a running example, namely a problem instance of a TNF version of the miconic domain (the original miconic domain can be found in optimal tracks of the IPC 2000), it is

defined as $\Pi_{RE} = \langle \mathcal{V}, \mathcal{O}, s_I, s_G, cost \rangle$ where

$$\mathcal{V} = \{\textit{lift-at}, \textit{boarded}, \textit{served}\},$$
$$\mathcal{O} = \{\textit{up}, \textit{down}, \textit{board}, \textit{depart}\},$$
$$\textit{pre}(\textit{up}) = \{(\textit{lift-at}, 0)\}, \textit{eff}(\textit{up}) = \{(\textit{lift-at}, 1)\},$$
$$\textit{pre}(\textit{down}) = \{(\textit{lift-at}, 1)\}, \textit{eff}(\textit{down}) = \{(\textit{lift-at}, 0)\},$$
$$\textit{pre}(\textit{board}) = \{(\textit{lift-at}, 1), (\textit{boarded}, 0)\}, \textit{eff}(\textit{board}) = \{(\textit{lift-at}, 1), (\textit{boarded}, 1)\},$$
$$\textit{pre}(\textit{depart}) = \{(\textit{lift-at}, 0), (\textit{boarded}, 1), (\textit{served}, 0)\},$$
$$\textit{eff}(\textit{depart}) = \{(\textit{lift-at}, 0), (\textit{boarded}, 0), (\textit{served}, 1)\},$$
$$s_I = \{(\textit{lift-at}, 0), (\textit{boarded}, 0), (\textit{served}, 0)\},$$
$$s_G = \{(\textit{lift-at}, 0), (\textit{boarded}, 0), (\textit{served}, 1)\},$$
$$cost(o) = 1 \; \forall o \in \mathcal{O}.$$

With $\mathcal{T}_{RE}$ we depict the corresponding induced transition system (see Figure 4).
The idea is that there is one passenger in the first floor, that wants to get to ground level. An elevator is ready at ground level. The operators are moving the elevator up and down, boarding and departing. There are a few technicalities, apart from obvious conditions. The passenger can only depart once the elevator picked him up and is at ground level again. Boarding is only possible from the first (initial) floor and departing is only possible on ground level.

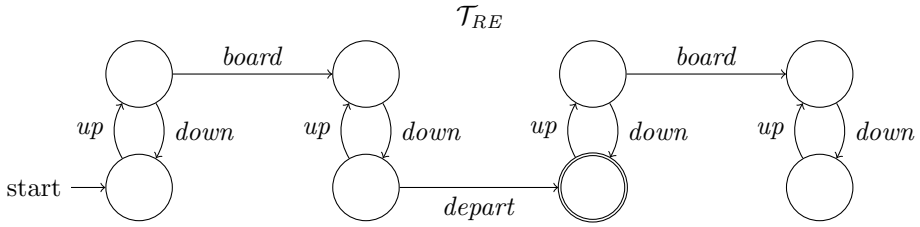

Figure 4: Transition system $\mathcal{T}_{RE}$ of the running example task. Note how the passenger can board again after departure, this technicality however won't be of any interest for us. The cheapest plan is $\langle \textit{up}, \textit{board}, \textit{down}, \textit{depart} \rangle$ with cost of 4.

Definition 1. *Given two transition systems $\mathcal{T}$, $\mathcal{T}'$ and a set of paths (plans) P of $\mathcal{T}$, we say that $\mathcal{T}'$ preserves paths (plans) P of $\mathcal{T}$ if p is a path (plan) in $\mathcal{T}'$ and has same cost, for all $p \in P$. $\mathcal{T}'$ preserves paths (plans) of $\mathcal{T}$ if all paths (plans) of $\mathcal{T}$ are preserved.*

A heuristic maps every state of a given transition system to a non-negative real number or ∞. The perfect heuristic h^* maps a state s to the cost of a cheapest path from s to the goal state, if there is no path the heuristic is ∞. We say that a heuristic is *admissible* if the value for every state s is equal to or smaller than h^*.
We define an *abstraction mapping* α as a mapping of a set of states to a subset of states. Given a transition system $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, cost, T, s_I, S_G \rangle$, the induced

abstract transition system is defined as $\mathcal{T}^\alpha = \langle S^\alpha, \mathcal{O}, cost, T^\alpha, s_I^\alpha, S_G^\alpha \rangle$, where $S^\alpha = \{\alpha(s) | s \in \mathcal{S}\}$, $T^\alpha = \{\langle \alpha(s), o, \alpha(s') \rangle | \text{ for all } \langle s, o, s' \rangle \in T\}$, $s_I^\alpha = \alpha(s_I)$ and $s_G^\alpha = \{\alpha(s) | s \in S_G\}$.

Theorem 1 (e.g. Helmert et al. (2008))**.** *Given a planning task* Π*, its transition system* $\mathcal{T}$ *and an abstraction mapping* α*. Then any plan of* $\mathcal{T}$ *is also a plan in* $\mathcal{T}^\alpha$*.*

An *atomic projection* α_v is an abstraction mapping that maps multiple states to the same state if they agree on the value of variable v. See Figure 5 for atomic projections of the transition system of our running example.

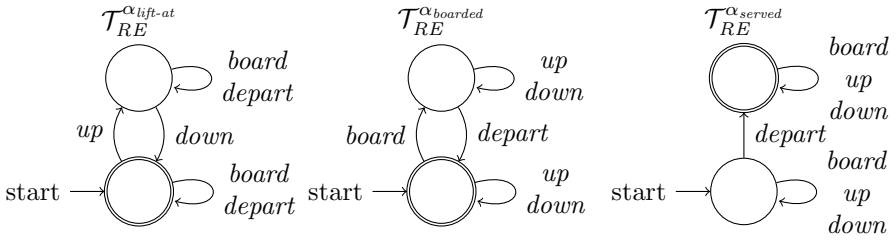

Figure 5: Depicts all three atomic projections.

The next concept is called network flow. A network flow describes how to get one unit from the initial state to a set of goal states and we will use it to formulate an LP. The idea is that by finding the most cost efficient way of getting the unit across we also get an estimate on how costly a cheapest plan is. We will however call network flows just flows for convenience.
Given a transition system $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, cost, T, s_I, S_G \rangle$ with $S_G = \{s_G^1, \ldots, s_G^k\}$ and *goal variables* $\boldsymbol{g} = \{g_1, \ldots, g_k\}$. A *flow* F is a mapping from $T \cup \boldsymbol{g}$ to the non-negative real numbers that has to fulfill the flow conditions, which is described in the following. Let $in(s) = \{\langle s', o, s \rangle \in T | o \in \mathcal{O} \text{ and } s' \in \mathcal{S}\}$ for all $s \in \mathcal{S}$ be the incoming transitions of s and $out(s) = \{\langle s, o, s' \rangle \in T | o \in \mathcal{O} \text{ and } s' \in \mathcal{S}\}$ for all $s \in \mathcal{S}$ the outgoing transitions of s. The flow conditions are defined as:

$$\forall s \in \mathcal{S} : \sum_{t \in out(s)} F(t) - \sum_{t \in in(s)} F(t) = [s = s_I] - [s \in S_G] \cdot g_s \text{ and}$$
$$\sum_x g_x = 1$$

where [ ] denotes the Iverson bracket and g_s is 0 if $s \notin S_G$ and g_x if $s = s_G^x$. The cost of F is defined as $cost(F) = \sum_{t=\langle s,o,s' \rangle \in T} F(t) \cdot cost(o)$.

The following theorem will make a link between plans and flows, which is necessary to relate the problem of finding cheap plans to finding cheap flows.

Theorem 2. *Given a transition system* $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, cost, T, s_I, S_G \rangle$ *with* $S_G = \{s_G^1, \ldots, s_G^k\}$ *and goal variables* $\boldsymbol{g} = \{g_1, \ldots, g_k\}$*. Then there exists a flow for every plan of* $\mathcal{T}$ *with the same cost.*

Proof. Given a plan $\pi : \langle o_1, \ldots, o_l \rangle$ of $\mathcal{T}$ with corresponding state path $\langle s_0, \ldots, s_l \rangle$ and the corresponding transitions $\boldsymbol{T} = \langle \langle s_0, o_1, s_1 \rangle, \ldots, \langle s_{l-1}, o_l, s_l \rangle \rangle$. Consider the mapping $M : T \cup \boldsymbol{g} \to \mathbb{R}^+$ that is defined as $M(t) = \sum_{t' \in \boldsymbol{T}} [t = t']$ for all $t \in T$ and $M(g_x) = 1$ if $s_l = s_G^x$ and 0 otherwise for all $x \in \{1, \ldots, k\}$. We will now check the flow conditions of M:

$$\forall s \in \mathcal{S} : \sum_{t \in out(s)} \sum_{t' \in \boldsymbol{T}} [t = t'] - \sum_{t \in in(s)} \sum_{t' \in \boldsymbol{T}} [t = t'] = [s = s_I] - [s \in S_G] \cdot g_s \text{ and}$$

$$\sum_x g_x = 1$$

First of all the goal variable condition is true because there is exactly one goal variable that is 1 and all the others are 0. Note that $\sum_{t \in out(s)} \sum_{t' \in \boldsymbol{T}} [t = t']$ is equal to the number of occurrences of outgoing transitions of s in $\boldsymbol{T}$. $\sum_{t \in in(s)} \sum_{t' \in \boldsymbol{T}} [t = t']$ analogously is equal to the number of occurrences of incoming transitions of s in $\boldsymbol{T}$.
Given any state $s \in \mathcal{S}$ we know that there is a transition of $out(s)$ at the start of $\boldsymbol{T}$ iff $s = s_0$, there is a pair of a transition in $in(s)$ and a transition in $out(s)$ in $\boldsymbol{T}$ if $s = s_i$ for all $i \in \{1, \ldots, l-1\}$ and a transition of $in(s)$ at the end of $\boldsymbol{T}$ iff $s \in S_G$. Furthermore there are no other transitions of s in $\boldsymbol{T}$ apart from those. Therefore the condition holds for s.
For every $o \in \pi$ there is a transition in $\boldsymbol{T}$ which means that $\sum_{t = \langle s, o, s' \rangle | s, s' \in \mathcal{S}} M(t)$ just adds up to the occurrences of o in π. Therefore we also know that $cost(M) = \sum_{t = \langle s, o, s' \rangle \in T | s, s' \in \mathcal{S}} M(t) \cdot cost(o) = \sum_{o \in \mathcal{O}} (\sum_{t = \langle s, o, s' \rangle | s, s' \in \mathcal{S}} M(t)) \cdot cost(o) = cost(\pi)$. □

As already mentioned we want to use flows for estimating the cost of plans. To do this we define the following flow heuristic.

Definition 2. *Given a planning task $\Pi : \langle \mathcal{V}, \mathcal{O}, s_I, s_G, cost \rangle$, its transition system $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, cost, T, s_I, S_G \rangle$ and a transition system $\mathcal{T}'$. The* flow heuristic *$h^{\mathcal{T}'}_{flow}(s)$ is then defined as the cost of a minimum flow of $\mathcal{T}'$. If there is no minimum flow of $\mathcal{T}'$, due to the fact that no flow exists, the heuristic is ∞, which also means that it is not possible to reach the goal.*

Theorem 3. *Given a planning task $\Pi : \langle \mathcal{V}, \mathcal{O}, s_I, s_G, cost \rangle$, its transition system $\mathcal{T}$ and a transition system $\mathcal{T}'$ which preserves at least one cheapest plan of $\mathcal{T}$. $h^{\mathcal{T}'}_{flow}(s)$ is admissible.*

Proof. With Theorem 2 we know that there exists a flow in $\mathcal{T}'$ for one of the cheapest paths from s to s_G with the same cost. As the heuristic value is the cost of the *cheapest* flow it either is equal to the cheapest path cost flow or to the cost of a flow which is even cheaper, which makes it an admissible heuristic. □

3 Time Unrolling

Next we are going to introduce the time unrolling which will be the core of this thesis, as it allows us to get rid of cycles in transition systems.

Definition 3. *A* time unrolling *(TU) τ_n is a mapping that removes cycles in transition systems, by introducing n time steps. Given a transition system $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, cost, T, s_I, S_G \rangle$, the induced* time unrolled transition system *is defined as $\mathcal{T}^{\tau_n} = \langle \mathcal{S}^{\tau_n}, \mathcal{O}, cost, T^{\tau_n}, s_I^{\tau_n}, S_G^{\tau_n} \rangle$, where*

$$\begin{aligned}
\mathcal{S}^{\tau_n} &= \{s_t | s \in \mathcal{S} \text{ and } t \in \{0 \dots n\}\} \text{ with } time(s_t) = t, \\
T^{\tau_n} &= \{\langle s_t, o, s'_{t+1} \rangle | \text{ for all } \langle s, o, s' \rangle \in T \text{ and } t \in \{0 \dots n-1\}\}, \\
s_I^{\tau_n} &= s_{I0} \text{ and} \\
S_G^{\tau_n} &= \{s_t | \text{ for all } t \in \{0 \dots n\} \text{ and } s \in S_G\}.
\end{aligned}$$

Figure 6 depicts the transition system of an atomic projection induced by time unrolling with 2 time steps. We can see that all cycles are gone, however we can see that only paths of length 2 are preserved, because transitions move in time and there are only 2 time steps. We group states and transitions into time layers as labeled.

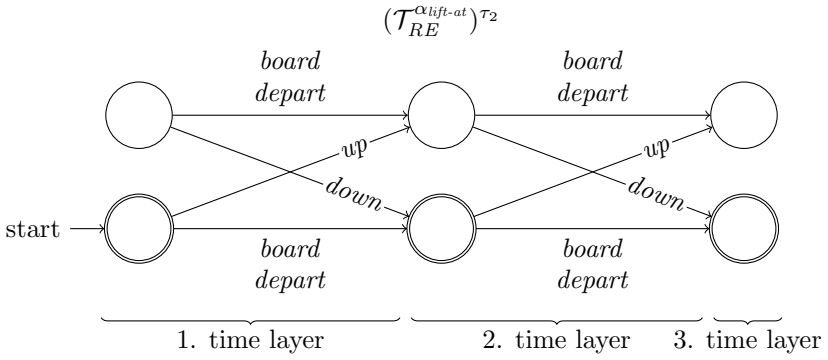

Figure 6: Shows time unrolling of atomic projection of variable *lift-at*.

In order to prove the admissibility of following heuristics the path preserving property of the time unrolling will be proven next.

Theorem 4. *Given a planning task Π, its transition system $\mathcal{T}$ and a time unrolling mapping τ_n. Then any plan of $\mathcal{T}$, which has a length smaller than or equal to n is also a plan in $\mathcal{T}^{\tau_n} = \langle \mathcal{S}^{\tau_n}, \mathcal{O}, cost, T^{\tau_n}, s_I^{\tau_n}, S_G^{\tau_n} \rangle$.*

Proof. Let $\pi = \langle o_1, \dots, o_l \rangle$ be an arbitrary plan in $\mathcal{T}$ of length $l \leq n$, with corresponding state path $\langle s_0, \dots, s_l \rangle$. We will now show that π is a path in $\mathcal{T}^{\tau_n}$ with state path $\langle (s_0)_0, \dots, (s_l)_l \rangle$ via induction over all subpaths. Let $\pi_x = \langle o_1, \dots, o_x \rangle$ be a subpath of π of length x.

Basis: *π_0 is a path in $\mathcal{T}^{\tau_n}$ with state path $\langle (s_0)_0 \rangle$.*
As π_0 is a plan of $\mathcal{T}$ we know that $s_I \in S_G$. This also means that $s_I^{\tau_n} \in S_G^{\tau_n}$.

Therefore the plan with length 0 is a plan in $\mathcal{T}^{\tau_n}$. Additionally $\langle (s_0)_0 \rangle$ is the state path of π_0 in $\mathcal{T}^{\tau_n}$ as $(s_0)_0$ is the initial state.
Inductive step: *If π_x is a path in $\mathcal{T}^{\tau_n}$ with state path $\langle (s_0)_0, \ldots, (s_x)_x \rangle$, then π_{x+1} is a path in $\mathcal{T}^{\tau_n}$ with state path $\langle (s_0)_0, \ldots, (s_{x+1})_{x+1} \rangle$.*
After applying all operators until o_x of π_{x+1} we already reached s_x in $\mathcal{T}$ and $(s_x)_x$ in $\mathcal{T}^{\tau_n}$. As we know that o_{x+1} is applicable in s_x (as π_{x+1} is a path) we know that there is a transition $\langle s_x, o_{x+1}, s_{x+1} \rangle$ in $\mathcal{T}$. Therefore we also have the transition $t = \langle (s_x)_x, o_{x+1}, (s_{x+1})_{x+1} \rangle$ in $\mathcal{T}^{\tau_n}$ (note that this is only true, because $x + 1 \leq l \leq n$). As π_x is a path in $\mathcal{T}^{\tau_n}$ we also know that π_{x+1} is path of $\mathcal{T}^{\tau_n}$, because of the existence of t whose start state is $(s_x)_x$. Additionally the state path of π_{x+1} in $\mathcal{T}^{\tau_n}$ is $\langle (s_0)_0, \ldots, (s_{x+1})_{x+1} \rangle$.

By mathematical induction we now know that after applying all operators of π in $\mathcal{T}^{\tau_n}$ the state $(s_l)_l$ is reached and this is also by definition a goal state as s_l is a goal state of $\mathcal{T}$. This means that π is not only a path but also a plan in $\mathcal{T}^{\tau_n}$. □

The first newly presented heuristic is based on time unrolling.

Definition 4. *Given a planning task $\Pi : \langle \mathcal{V}, \mathcal{O}, s_I, s_G, cost \rangle$ and its transition system $\mathcal{T}$, an abstraction mapping α and a time unrolling mapping τ_n we define the time unrolling heuristic $h^{TU}_{\alpha,\tau}$ to be $h^{\mathcal{T}_s^{\alpha\tau_n}}_{flow}(s)$.*

The time unrolling heuristic is admissible if the n of the time unrolling is bigger than the shortest length of the cheapest plans, as then both the abstraction and the time unrolling mapping preserve at least one cheapest plan (see Theorem 1 and 4).
The length of a cheapest plan is smaller than or equal to the product over the domain sizes of all variables. This means that it is always possible to set n high enough to make the heuristic admissible, but this makes the computation exponential in general.

The idea of the *time unrolling with repetition* (TUR) is to introduce all transitions of the original transition system after the last time layer (this might lead to cycles in the last time layer), so that it becomes possible to find flows that correspond to plans of arbitrary length.

Definition 5. *Given a transition system $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, cost, T, s_I, S_G \rangle$ and the time unrolled transition system $T^{\tau_n} = \langle \mathcal{S}^{\tau_n}, \mathcal{O}, cost, T^{\tau_n}, s_I^{\tau_n}, S_G^{\tau_n} \rangle$. A TUR τ_n^r is a mapping that removes cycles in transition systems in the first $n+1$ time layers, by introducing n time steps and furthermore copies the transitions of the original transition system into the last time layer. The induced* time unrolled and repeated transition system *is defined as $\mathcal{T}^{\tau_n^r} = \langle \mathcal{S}^{\tau_n}, \mathcal{O}, cost, T^{\tau_n} \cup T^r, s_I^{\tau_n}, S_G^{\tau_n} \rangle$, where*

$$T^r = \{ \langle s_n, o, {s'}_n \rangle | \textit{for all } \langle s, o, s' \rangle \in T \}$$

We will first show the path preserving property, to guarantee admissibility of the heuristic that is based on TUR.

Theorem 5. *Given a TUR τ_n^r with n time steps and any transition system $\mathcal{T}$. Every plan of $\mathcal{T}$ is also a plan in $\mathcal{T}^{\tau_n^r}$.*

Proof. Let $\pi_l = \langle o_1, \ldots, o_l \rangle$ be an arbitrary plan in $\mathcal{T}$ of length l, with corresponding state path $\langle s_0, \ldots, s_l \rangle$. We will now show that π_l is a path in $\mathcal{T}^{\tau_n^r}$ with state path $\langle (s_0)_0, \ldots, (s_n)_n, (s_{n+1})_n, \ldots, (s_l)_n \rangle$, via induction over the length l.

Basis: *π_k is a path in $\mathcal{T}^{\tau_n^r}$ with state path $\langle (s_0)_0, \ldots, (s_k)_k \rangle$ for all $k \in \{1, \ldots, n\}$.*
Consider a time unrolling τ_n (with the same amount of time steps as τ_n^r). We know that $\mathcal{T}^{\tau_n}$ and $\mathcal{T}^{\tau_n^r}$ are the same transition system, apart from the additional transitions in $\mathcal{T}^{\tau_n^r}$, which means that every path in $\mathcal{T}^{\tau_n}$ is also a path in $\mathcal{T}^{\tau_n^r}$ (with identical state paths). If we additionally take Theorem 4 into consideration we know that the basis holds.
Inductive step: *If π_x is a path in $\mathcal{T}^{\tau_n^r}$ where $\langle (s_0)_0, \ldots, (s_n)_n, (s_{n+1})_n, \ldots, (s_x)_n \rangle$ is the corresponding state path, then π_{x+1} is a path in $\mathcal{T}^{\tau_n^r}$ with state path $\langle (s_0)_0, \ldots, (s_n)_n, (s_{n+1})_n, \ldots, (s_{x+1})_n \rangle$ for all $x > n$.*
After applying all operators until o_x of π we already reached s_x in $\mathcal{T}$ and $(s_x)_n$ in $\mathcal{T}^{\tau_n^r}$. As we know that o_x is applicable in s_x (as π_x is a path) we know that there is a transition $\langle s_x, o_x, s_{x+1} \rangle$ in $\mathcal{T}$. Therefore we also got the transition $t = \langle (s_x)_n, o_x, (s_{x+1})_n \rangle$ in $\mathcal{T}^{\tau_n^r}$. As π_x is a path in $\mathcal{T}^{\tau_n}$ we also know that π_{x+1} is path of $\mathcal{T}^{\tau_n}$, because of the existence of t whose start state is $(s_x)_n$. Additionally the state path of π_{x+1} in $\mathcal{T}^{\tau_n}$ is $\langle (s_0)_0, \ldots, (s_n)_n, (s_{n+1})_n, \ldots, (s_{x+1})_n \rangle$ (see end state of t).

By mathematical induction we now know that after applying all operators of π_l in $\mathcal{T}^{\tau_n^r}$ we are in $(s_l)_n$ and this is also by definition a goal state as s_l is a goal state of $\mathcal{T}$. This means that π_l is not only a path but also a plan in $\mathcal{T}^{\tau_n^r}$. □

We define the TUR heuristic as a flow heuristic.

Definition 6. *Given a planning task $\Pi : \langle \mathcal{V}, \mathcal{O}, s_I, s_G, cost \rangle$ and its transition system $\mathcal{T}$, an abstraction mapping α and a TUR τ_n^r the TUR heuristic $h_{\alpha,n}^{TUR}$ is defined as $h_{flow}^{(\mathcal{T}_s^\alpha)^{\tau_n^r}}(s)$.*

Because TURs preserve all paths no matter how many time steps are used $h_{\alpha,n}^{TUR}$ is admissible in all cases.

We want to be able to combine the information of multiple time unrolled abstract transition systems. This leads us to the more general idea of t-synchronizable transition systems.

Definition 7. *Given a transition system $\mathcal{T}$ with state set $\mathcal{S}$. $\mathcal{T}$ is n* times time synchronizable*, if for every state $s \in \mathcal{S}$ with $d^*(s) < n$ all paths to s have length $d^*(s)$.*

Note that if a transition system is n times time synchronizable, we know that for every state $s \in \mathcal{S}$ with $d^*(s) < n$ the outgoing transitions end in a state s' with $d^*(s') = d^*(s) + 1$, because if that wouldn't be the case, then there would be shorter or longer paths to s' than the shortest paths, which contradicts with $\mathcal{T}$ being n times time synchronizable.
Given an n times time synchronizable transition system, we say that a state s is in the k-th time layer if $d^*(s) = k$. Analogously a transition is in the k-th time layer if its start state is in the k-th time layer.

We will now show that time unrolled transition systems are time synchronizable.

Theorem 6. *If a transition system $\mathcal{T}$ is induced by a TU or TUR with n time steps, it is n times time synchronizable.*

Proof. If a transition system $\mathcal{T}$ is induced by a TU or TUR with n time steps, we know that for any state s with $d^*(s) < n$ it holds that all paths to s have length $time(s)$, as for states with $time(s) < n$ it holds that they can only be reached by states of the previous time layer and the initial state is in the first time layer. □

The idea behind time synchronizability is that if a transition system is n times time synchronizable we know that every transition that can be reached within n steps from the initial state can only be the n-th transition of a path or plan. We can use this information to synchronize the flow of several transition systems.

Given a set $\boldsymbol{\mathcal{T}}$ of several abstract transition systems, that are n times time synchronizable and a plan π of the original transition system with corresponding flows (see Theorem 2 for details). From the equations of the state equation heuristic (Pommerening et al. (2014)) we already know that the sum of flows of transitions with same operator have to be equal (note that all flows originate from the same plan and thus the operator counts are equal). As the transition systems of $\boldsymbol{\mathcal{T}}$ are n times time synchronizable we can additionally guarantee the operator counts within each of the first n time layers are equal (instead of summing them up over all time layers). This follows from the fact that these sums are either one or zero depending on whether an operator was used in the according time step. Therefore the time synchronization conditions not only guarantee that the same amount of operators are used, but also that they are used in the same time steps up until n.
With the notion of a *time synchronized flow* we will be able to combine information of multiple synchronizable transition systems and use the described synchronization technique.

Given a set of transition systems $\boldsymbol{\mathcal{T}} = \{\mathcal{T}^1, \ldots, \mathcal{T}^k\}$, that are at least n times time synchronizable, and the corresponding goal variable sets $\boldsymbol{g}^i$ with $i \in \{1, \ldots, k\}$ and $\mathcal{T}^i = \langle \mathcal{S}^i, \mathcal{O}, cost, T^i, s^i_I, S^i_G \rangle$. Let $\boldsymbol{T}_u$ be the union of the transition sets and $\boldsymbol{g}_u$ the union of the goal variable sets. A *n times time synchronized flow* $F^{\boldsymbol{\mathcal{T}}}_n$ is a function $\boldsymbol{T}_u \cup \boldsymbol{g}_u \to \mathbb{R}^+$ that maps transitions and goal variables to the non-negative real numbers and has three types of conditions.
First of all $F^{\boldsymbol{\mathcal{T}}}_n$ has to fulfill the flow conditions of $\boldsymbol{\mathcal{T}} = \{\mathcal{T}^1, \ldots, \mathcal{T}^k\}$. Next we want to synchronize the flow of an operator in a single time step, for which we define the *operator time flow* of operator o and time t in transition system $\mathcal{T}^i$ for all $i \in \{1, \ldots, k\}$ as

$$otflow^{=t}_o(T^i) = \sum_{\substack{t' = \langle s, o, s' \rangle \in T^i \\ s, s' \in \mathcal{S}^i \\ d^*(s') = t}} F^{\boldsymbol{\mathcal{T}}}_n(t').$$

The operator time flow after time step t for operator o $otflow^{>t}_o(T^i)$ is defined analogously.
The flow per operator and time steps can be synchronized for the first n time

steps with the *time synchronization conditions*, which are defined as follows

$$\begin{aligned} &\mathit{otflow}_o^{=t}(T^1) = \cdots = \mathit{otflow}_o^{=t}(T^k) \\ &\text{for all } o \in \mathcal{O} \text{ and all } t \in \{1, \ldots, n\}. \end{aligned}$$

The flow of the remaining time steps will only be synchronized with sums over the remaining time steps. The *summed operator flow* conditions are

$$\begin{aligned} &\mathit{otflow}_o^{>n}(T^1) = \cdots = \mathit{otflow}_o^{>n}(T^k) \\ &\text{for all } o \in \mathcal{O}. \end{aligned}$$

The *summed flow sflow* of $F_n^{\boldsymbol{\mathcal{T}}}$ is defined as

$$\sum_{o \in \mathcal{O}} \sum_{t \in \{1,\ldots,n\}} \mathit{otflow}_o^t,$$

it will be required for later proofs.
We furthermore define $otflow_o^{=t} = otflow_o^{=t}(T^i)$ for all $i \in \{1, \ldots, k\}$ and $otflow_o^{>n}$ analogously. Finally the cost of $F_n^{\boldsymbol{\mathcal{T}}}$ is defined as

$$\mathit{cost}(F^{\boldsymbol{\mathcal{T}}}) = \sum_{o \in \mathcal{O}, t \in \{1,\ldots,n\}} \mathit{otflow}_o^t \cdot \mathit{cost}(o) + \sum_{o \in \mathcal{O}} \mathit{otflow}_o^{>n} \cdot \mathit{cost}(o).$$

Additionally we define an n times time synchronized integer flow as an n times time synchronized flow that only maps to non-negative integers. If we only talk about a *time synchronized flow* we mean a n times time synchronized flow where n is equal to the minimum number for that holds that every transition system in $\boldsymbol{\mathcal{T}}$ is n times time synchronizable.

The notion of time synchronized flows will be used in the following heuristic.

Definition 8. *Given a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_I, s_G, \mathit{cost} \rangle$ with transition system $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, \mathit{cost}, T, s_I, S_G \rangle$, a set of transition systems $\boldsymbol{\mathcal{T}} = \{\mathcal{T}^1, \ldots, \mathcal{T}^k\}$, that are at least n times time synchronizable. The n times time synchronized flow heuristic $h^{\boldsymbol{\mathcal{T}}}_{n\text{-}ts\text{-}flow}(s)$ is equal to the cost of the cheapest $F_n^{\boldsymbol{\mathcal{T}}}$.*

To show when time synchronized flow heuristics are admissible in certain cases, we show that time synchronized flows have an existence property similar to the existence property of normal flows.

Theorem 7. *Given a transition system $\mathcal{T}$ and a set of transition systems $\boldsymbol{\mathcal{T}} = \{\mathcal{T}^1, \ldots, \mathcal{T}^k\}$, that are at least n times time synchronizable and preserve plans P of $\mathcal{T}$, the corresponding goal variable sets $\boldsymbol{g}^i$ with $i \in \{1, \ldots, k\}$ and $\mathcal{T}^i = \langle \mathcal{S}^i, \mathcal{O}, \mathit{cost}, T^i, s_I^i, S_G^i \rangle$. Let $\boldsymbol{T}_u$ be the union of the transition sets and $\boldsymbol{g}_u$ the union of the goal variable sets.*
For every plan $\pi = \langle o_1, \ldots, o_l \rangle$ in P there is a j times time synchronized flow of $\boldsymbol{\mathcal{T}}$ with the same cost for all $j \in \{0, \ldots, n\}$.

Proof. We know that π is also a plan in the transition systems in $\boldsymbol{\mathcal{T}}$. There are flows $\boldsymbol{F} = \{F^1, \ldots, F^k\}$ in π (as described in Theorem 2) for corresponding transition systems in $\boldsymbol{\mathcal{T}}$. With these we define a mapping $M : \boldsymbol{T}_u \cup \boldsymbol{g}_u \to \mathbb{R}^+$ with $M(t) = F^i(t)$ for all $i \in \{1, \ldots, k\}$ and all $t \in T^i$, additionally $M(\alpha) =$

$F^i(\alpha)$ for all $i \in \{1, \dots, k\}$ and all $\alpha \in \boldsymbol{\alpha}^i$. We now check the flow conditions of M:

$$\forall i \in \{1, \dots, k\} \forall s \in \mathcal{S}^i : \sum_{t \in out(s)} M(t) - \sum_{t \in in(s)} M(t) = [s = s_I^i] - [s \in S_G^i] \cdot g_s^i \text{ and}$$
$$\sum_x g_x^i = 1$$

With the definition of M we know that these are equivalent to:

$$\forall i \in \{1, \dots, k\} \forall s \in \mathcal{S}^i : \sum_{t \in out(s)} F^i(t) - \sum_{t \in in(s)} F^i(t) = [s = s_I^i] - [s \in S_G^i] \cdot g_s^i \text{ and}$$
$$\sum_x g_x^i = 1$$

We can see that these are just the union of the flow conditions of the flows in $\boldsymbol{F}$. As we know that the individual flow conditions are fulfilled, we also know that the union holds.
Next we check the time synchronization conditions:

$$otflow_o^{=t}(T^1) = \dots = otflow_o^{=t}(T^k)$$
$$\text{for all } o \in \mathcal{O} \text{ and all } t \in \{1, \dots, j\}$$

It holds that

$$otflow_o^{=t}(T^i) = \sum_{\substack{t' = \langle s, o, s' \rangle \in T^i \\ s, s' \in \mathcal{S}^i \\ d^*(s') = t}} \sum_{t^* \in \boldsymbol{T^i}} [t' = t^*] \overset{(*)}{=} \begin{cases} 1 & \text{if } t \leq l \text{ and } o = o_t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$
$$\text{for all } o \in \mathcal{O}, \text{ all } t \in \{1, \dots, n\} \text{ and all } i \in \{1, \dots, k\},$$
$$\text{where } \boldsymbol{T^i} \text{ is transition path of } \pi \text{ in } T^i.$$

(*) If we look at the transitions in T^i that end in a state with a fixed minimal distance $t \in \{1, \dots, n\}$, we know that there is exactly one of them (its operator is equal to the t-th operator in π: o_t) that is once in $\boldsymbol{T^i}$, provided that π is least t long (if π is shorter, there is none at all), and all others are not at all or more than once in $\boldsymbol{T^i}$ as otherwise there is a contradiction with T^i being n times time synchronizable.
The summed operator flow conditions are:

$$otflow_o^{>n}(T^1) = \dots = otflow_o^{>n}(T^k)$$
$$\text{for all } o \in \mathcal{O}.$$

We know that

$$otflow_o^{>n}(T^i) = \sum_{\substack{t' = \langle s, o, s' \rangle \in T^i \\ s, s' \in \mathcal{S}^i \\ d^*(s') > n}} \sum_{t^* \in \boldsymbol{T^i}} [t' = t^*] \overset{(*)}{=} \sum_{o' \in \{o_n, \dots, o_l\}} [o = o'] \tag{2}$$
$$\text{for all } o \in \mathcal{O}, \text{ all } \text{ and all } i \in \{1, \dots, k\},$$
$$\text{where } \boldsymbol{T^i} \text{ is transition path of } \pi \text{ in } T^i.$$

(*) $\boldsymbol{T^i}$ contains a transition for every $o \in \pi$. The transitions for the operators $\{o_1, \ldots, o_n\}$ correspond to transitions with end states that have a minimal distance of at most o_n and are not part of the outer sum of $\mathit{otflow}_o^{>n}(T^i)$ for all $o \in \mathcal{O}$. The transitions for the leading operators: $\{o_{n+1}, \ldots, o_l\}$ however are part of the sum (otherwise there is a contradiction with T^i being n times time synchronizable) and thus $\mathit{otflow}_o^{>n}(T^i)$ is just equal to the number of times o appears in $\{o_{n+1}, \ldots, o_l\}$ for all $o \in \mathcal{O}$.
With (1) and (2) we know that $\sum_{o \in \mathcal{O}, t \in \{1,\ldots,n\}} \mathit{otflow}_o^t$ adds up to the number of occurrences of o in $\{o_1, \ldots, o_n\}$ and $\sum_{o \in \mathcal{O}} \mathit{otflow}_o^{>n}$ adds up the occurrences of o in $\{o_{n+1}, \ldots, o_l\}$, therefore M and π have the same cost. □

Using this relationship of time synchronized flows and plans to proof admissibility of the time synchronized flow heuristic in certain cases is the next step.

Theorem 8. *Given a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_I, s_G, \mathit{cost} \rangle$ with transition system $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, \mathit{cost}, T, s_I, S_G \rangle$, a set of transition systems $\boldsymbol{\mathcal{T}} = \{\mathcal{T}^1, \ldots, \mathcal{T}^k\}$, that are at least n times time synchronizable.*
$h^{\boldsymbol{\mathcal{T}}}_{n\text{-}ts\text{-}flow}(s)$ is admissible if the transition systems in $\boldsymbol{\mathcal{T}}$ preserve at least one of the cheapest plans in $\mathcal{T}_s$ for all $s \in \mathcal{S}$.

Proof. With Theorem 7 we know that there is a $F_n^{\boldsymbol{\mathcal{T}}}$ with the same cost as one of the cheapest plans of Π_s. As the heuristic value is equal to the *cheapest* $F_n^{\boldsymbol{\mathcal{T}}}$, the heuristic is admissible. □

For the rest of the thesis we will now fix the $\boldsymbol{\mathcal{T}}$ in $h^{\boldsymbol{\mathcal{T}}}_{n\text{-}ts\text{-}flow}(s)$ to TUs or TURs of atomic projections to further investigate properties of this fixed type of heuristic.

Definition 9. *Given a planning task $\Pi = \langle \mathcal{V} = \{v_1, \ldots, v_k\}, \mathcal{O}, s_I, s_G, \mathit{cost} \rangle$ with transition system $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, \mathit{cost}, T, s_I, S_G \rangle$, the set of all atomic projections $\{\alpha_{v_1}, \ldots, \alpha_{v_k}\}$, a TU(R) $\tau_n^{(r)}$ and the set $\boldsymbol{\mathcal{T}}_s = \{(\mathcal{T}_s^{\alpha_{v_1}})^{\tau_n^{(r)}}, \ldots, \mathcal{T}_s^{\alpha_{v_k}})^{\tau_n^{(r)}}\}$ for all $s \in \mathcal{S}$. The n times atomic time unrolling (with repetition) heuristic $h_n^{ATU(R)}$ is defined as $h^{\boldsymbol{\mathcal{T}}_s}_{n\text{-}ts\text{-}flow}(s)$. The IP version of $h_n^{ATU(R)}$ is denoted by $h_{n,\mathbb{N}}^{ATU(R)}$.*

We will now analyze how the heuristics h_0^{ATUR} and h_1^{ATUR} evaluate the goal distance from the initial state of our running example. h_0^{ATUR} is equal to the cheapest 0 times synchronized flow of the atomic projections, which is depicted in Figure 7. Note how the summed flows for each operator are equal in all three flows. The heuristic value is equal to 2. If we now look at corresponding cheapest flow of h_1^{ATUR} at the bottom of Figure 7, we can see that the summed flows for each operator are equal in each of the two time layers. Introducing one time step leads to an increase in heuristic value from 2 to 4, where 4 is already the perfect value. The synchronized flow in the original atomic projections is missing the point, that for boarding and departing the lift has to be at the right place. The time synchronization forces the flow in the first time layer to be along the same operator in all three transition systems, which in turn leads to a situation as with 0 time steps, however with different starting positions. For the cheapest time synchronized flow the flow in the first time layer corresponds to operator up. This leads to a situation where the lift is at the top and has to get to the bottom again and the passenger has to board and depart still.
With Theorem 8 describing admissibility criteria of previously introduced heuristics is straight forward.

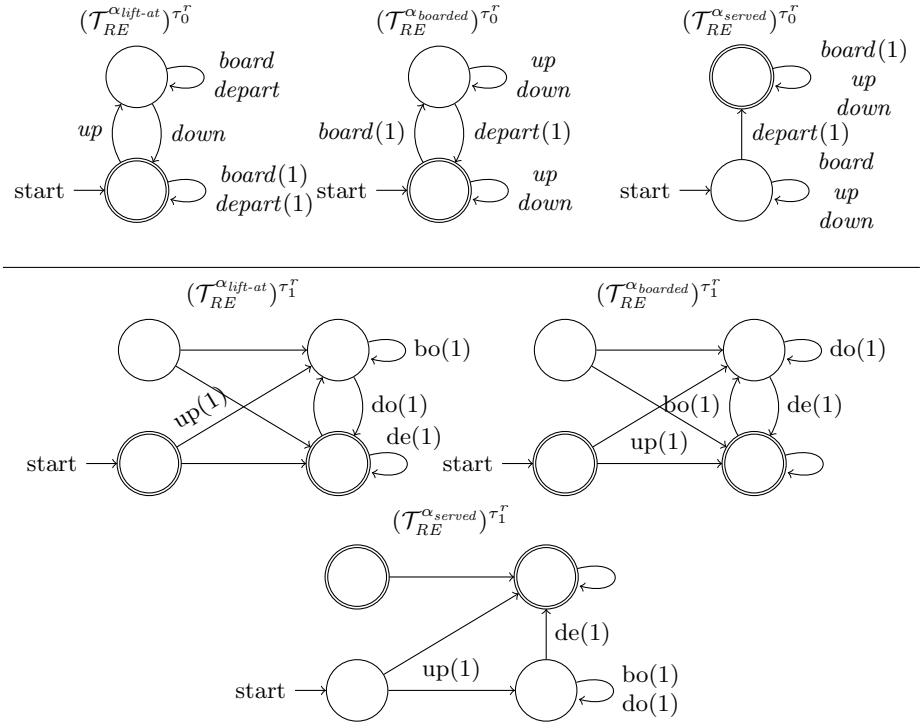

Figure 7: At the top we can see the time synchronized flow corresponding to $h_0^{ATUR}(s_I)$ and at the bottom the corresponding time synchronized flow of $h_1^{ATUR}(s_I)$. In the bottom part the operator names got truncated to two characters and are only visible if they correspond to a transition with a flow value. Initial heuristic value increases from 2 to 4.

Theorem 9. *h_n^{ATU} is admissible if n is equal to or greater than the length of the cheapest plan of Π and h_n^{ATUR} is admissible in all cases.*

Proof. h_n^{ATU} is admissible if n is equal to or greater than the length of the cheapest plan of Π, because this means that the time unrolled atomic transition systems contain the cheapest plan (Theorem 4), which means that there exists a synchronized flow of same cost (Theorem 7), which makes h_n^{ATU} admissible, because it is equal to the cheapest flow.
The heuristic h_n^{ATUR} is admissible in all cases, because the time unrolled atomic transition systems contain the cheapest plan regardless of the value of n (Theorem 4) and the same chain of reasoning can be applied. □

The same holds for $h_{n,\mathbb{N}}^{ATU(R)}$, as the flow created in Theorem 7 is an integer flow. h_0^{ATUR} is closely related to the state equation h^{SEQ}.

Theorem 10. $h_0^{ATUR} = h^{SEQ}$.

Proof. We know that h^{SEQ} can be described as flows over domain transition graphs (DTGs) (Bonet et al., 2014). These graphs are, at least for TNF tasks, equally defined as atomic projections. τ_0^r does per definition not alter transition systems. With 0 time steps there are only the summed operator flow conditions active which is also the way how the flows over DTGs are combined by h^{SEQ}. □

We will now define another time heuristic that we want to examine experimentally and theoretically.

Definition 10. *The minimal atomic time unrolling heuristic h^{MATU} is defined as h_n^{ATU} where $n = \text{argmin}_n\, h_n^{ATU} \neq \infty$. If there is no n with $h_n^{ATU} \neq \infty$, then $h^{MATU} = \infty$. The integer version is denoted by $h_{\mathbb{N}}^{MATU}$.*

For the admissibility proof we need the following theorem.

Theorem 11. *Given a set of transition systems $\boldsymbol{\mathcal{T}} = \{\mathcal{T}^1, \ldots, \mathcal{T}^k\}$, that are at least n times time synchronizable, and the corresponding goal variable sets $\boldsymbol{g}^i$ with $i \in \{1, \ldots, k\}$ and $\mathcal{T}^i = \langle \mathcal{S}^i, \mathcal{O}, cost, T^i, s_I^i, S_G^i \rangle$. Let $\boldsymbol{T}_u$ be the union of the transition sets and $\boldsymbol{g}_u$ the union of the goal variable sets.*
If $F_m^{\boldsymbol{\mathcal{T}}}$ with $m \leq n$ is a m times time synchronized flow, then it holds that

$$tflow^{=t} \leq 1 \text{ for all } t \in \{1, \ldots, m-1\}$$

where for a time step t the time flow *$tflow^{=t}$ is defined as $\sum_{o \in \mathcal{O}} otflow_o^{=t}$.*

Proof. First of all if $m = 0$ the theorem holds as there are no conditions in this case. We will prove the other cases with the following intermediate results:

$$tflow^{=1} \leq 1 \text{ and} \tag{1}$$

$$tflow^{=t+1} \leq tflow^{=t} \text{ for all } t \in \{1, \ldots, m-1\}. \tag{2}$$

(1): We will now look at a subpart of the flow conditions of $F_m^{\boldsymbol{\mathcal{T}}}$, which effect $tflow^{=1}$, namely the conditions for the states s in $\mathcal{S}^i$ with $d^*(s) = 0$ for all $i \in \{1, \ldots, k\}$:

$$\sum_{t \in out(s)} F(t) - \sum_{t \in in(s)} F(t) = [s = s_I] - [s \in (S_G)^i] \cdot g_s^i$$

Remember that we are only considering transition systems without dead states and therefore there is only the initial state that belongs to this case. There are no incoming transitions to it, because that would contradict with $\mathcal{T}^i$ being at least 1 times time synchronizable.
It furthermore holds that $\sum_{t \in out(s)} F(t) = tflow^{=1}$ and the right side of the equation is between 0 and 1 (remember goal variables sum up to 1) therefore (1) holds.
(2): We know that

$$tflow^{=t+1} = \sum_{\substack{s \in \mathcal{S}^i \\ d^*(s)=t}} \sum_{t \in out(s)} F(t) = \sum_{\substack{s \in \mathcal{S}^i \\ d^*(s)=t}} \sum_{t \in in(s)} F(t) + \underbrace{[s = s_I]}_{0} \underbrace{-[s \in (S_G)^i] \cdot g^i_s}_{\leq 0}$$
$$\leq \sum_{\substack{s \in \mathcal{S}^i \\ d^*(s)=t}} \sum_{t \in in(s)} F(t) = tflow^{=t} \text{ for all } t \in \{1, \ldots, m-1\}.$$

As (1) and (2) hold and $\leq$ is transitive, the theorem holds. □

Now that we have bound the summed flow of a time synchronized flow in each time layer, we can prove admissibility of h^{MATU}, but only in the case of planning tasks with unit costs.

Theorem 12. *Given a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_I, s_G, cost \rangle$, h^{MATU} is admissible if $cost(o) = 1$ for all $o \in \mathcal{O}$.*

Proof. With $n = \text{argmin}_n h^{ATU}_n(s) \neq \infty$ we know that the shortest and also cheapest path is at least of length and cost n (see Theorem 7). Together with Theorem 11 we also know that the the overall flow $sflow$ can be at most n if there are n time layers. This also means that the cost can be at most n, as the cost is just the individual flows times the corresponding operator cost and the costs are all 1. Therefore h^{MATU} is admissible. □

We now want to prove that $h^{ATU}_{n,\mathbb{N}}(s) = h^*(s)$ if $d^*(s) \leq n$. For this we are going to convert integer time synchronized flows into plans with the same cost. The next theorem is used to make this proof easier.

Theorem 13. *Given a set of abstract transition system $\boldsymbol{\mathcal{T}}$ that are at least k times time synchronizable and preserve at least one of the cheapest plans of the original transition system. Let $\boldsymbol{\mathcal{T}}_\setminus$ be the set which holds the transition systems of $\boldsymbol{\mathcal{T}}$ without the dead states.*
It holds that $h^{\boldsymbol{\mathcal{T}}}_{n\text{-}ts\text{-}flow}(s) \overset{(1)}{\leq} h^{\boldsymbol{\mathcal{T}}_\setminus}_{n\text{-}ts\text{-}flow}(s) \overset{(2)}{\leq} h^*(s)$ for all $n \leq k$.

Proof. As we know that all states and transitions of the transition systems in $\boldsymbol{\mathcal{T}}_\setminus$ are also contained in the corresponding transition systems in $\boldsymbol{\mathcal{T}}$, we know that for every synchronized flow in $\boldsymbol{\mathcal{T}}_\setminus$ there is one in $\boldsymbol{\mathcal{T}}$ with the same cost (note that for the corresponding flows the additional constraints of the dead states are fulfilled, because dead states are neither initial nor goal states and the synchronized flow of incoming and outgoing transitions are 0, because they are not contained in $\boldsymbol{\mathcal{T}}_\setminus$), thus (1) holds.
With all dead states removed the transition systems in $\boldsymbol{\mathcal{T}}_\setminus$ still preserve at least one of the cheapest plans, therefore we know that $h^{\boldsymbol{\mathcal{T}}_\setminus}_{n\text{-}ts\text{-}flow}(s)$ is a n times time synchronized flow heuristic and thus (2) is true.

□

If a transition system contains dead states it is possible to identify them with a forward and backward search. To make proofs easier we are going to assume that every transition system mentioned in the rest of this section does not contain dead states.
We will now prove that certain integer time synchronized flows can be transformed into plans of same cost.

Theorem 14. *Let* Π *be a planning task with goal state* s_G, $\mathcal{T}$ *its transition system with variables* $v_1, \ldots, v_k$ *and* $\boldsymbol{\mathcal{T}}$ *a set of transition systems defined as* $\{(\mathcal{T}^{\alpha_{v_1}})^{\tau_n}, \ldots, \mathcal{T}^{\alpha_{v_k}})^{\tau_n}\}$.
Given any integer n times time synchronized flow $F_n^{\boldsymbol{\mathcal{T}}}$. *There exists a plan of* Π *with same cost.*

Proof. With Theorem 11 and the fact that $F_n^{\boldsymbol{\mathcal{T}}}$ only has integer values, we know that there is one transition per time layer for that $F_n^{\boldsymbol{\mathcal{T}}}$ is 1 and for all others it is 0, for the first *sflow* time layers.
Let π be an operator sequence of length *sflow*. The k-th operator of π is o if the transition in the k-th time layer, for that $F_n^{\boldsymbol{\mathcal{T}}}$ is 1, corresponds to operator o. We will now show that π is a plan via induction over the subpaths.
Let l be the length of π and π_x the subpath of π with length x (note that $0 \leq x \leq l$ and $\pi_l = \pi$) and v is any variable of Π.
Basis: π_0 *is a path in* $\mathcal{T}$ *with state path* $\langle s_0 \rangle$ *and a path in* $(\mathcal{T}^{\alpha_v})^{\tau_n}$ *with state path* $\langle \alpha_v(s_0)_0 \rangle$.
After applying no operators to any planning task the initial state is the only one visited, in this case s_0 and $\alpha_v(s_0)_0$ are the initial states.
Inductive step: *If* π_x *is a path in* $\mathcal{T}$ *with corresponding state path* $\langle s_0, \ldots, s_x \rangle$ *and a path in* $(\mathcal{T}^{\alpha_v})^{\tau_n}$ *with state path* $\langle \alpha_v(s_0)_0, \ldots, \alpha_v(s_x)_x \rangle$, *then* π_{x+1} *is a path in* $\mathcal{T}$ *with state path* $\langle s_0, \ldots, s_{x+1} \rangle$ *and a path in* $(\mathcal{T}^{\alpha_v})^{\tau_n}$ *with state path* $\langle \alpha_v(s_0)_0, \ldots, \alpha_v(s_{x+1})_{x+1} \rangle$.
To show that π_{x+1} is a path in $\mathcal{T}$, we have to show that the $(x+1)$-th operator of π, we will referring to it as o_{x+1}, is applicable in s_x. Through the definition of π we know that there is the transition $\langle \alpha_v(s_x)_x, o_{x+1}, \alpha_v(s_{x+1})_{x+1} \rangle$ in $(\mathcal{T}^{\alpha_v})^{\tau_n}$ (see proof of Theorem 2 to see why the end state can be expressed as stated). Thus we also know that there is a transition $\langle \alpha_v(s_x), o_{x+1}, \alpha_v(s_{x+1}) \rangle$ in $\mathcal{T}^{\alpha_v}$, we then deduce that either $s_x(v) = [pre(o_{x+1})](v)$ or $v \notin vars(pre(o_{x+1}))$. This means that s_x fulfills all preconditions of o_{x+1} and therefore π_{x+1} is a path in $\mathcal{T}$.

By mathematical induction we know that π is a path in Π. We furthermore know that after applying π in $(\mathcal{T}^{\alpha_v})^{\tau_n}$, $\alpha_v(s_l)_l$ is reached. Due to the flow conditions of $F_n^{\boldsymbol{\mathcal{T}}}$ being fulfilled, we know that $\alpha_v(s_l)_l$ is a goal state in the abstract transition system, as there is a incoming flow of 1 and no outgoing flow. This also means that $\alpha_v(s_l)$ is a goal state, from which we deduce that $s_l(v) = s_G(v)$. As this holds true for arbitrary v we deduce that $s_l = s_G$ and π is not only a path, but a plan in $\mathcal{T}$.
Furthermore π is of same cost as $F_n^{\boldsymbol{\mathcal{T}}}$, because the cost of $F_n^{\boldsymbol{\mathcal{T}}}$ is defined as the individual flows times the corresponding operator cost which is of course only the cost of π, as the flows are all 1. That means that there is a plan of same cost for every integer n times synchronized flow.

□

Now that we have both directions, from plan to synchronized flow and from

synchronized flow to plan we prove that $h^{ATU}_{n,\mathbb{N}}(s)$ is a perfect heuristic in a special case.

Theorem 15. *$h^{ATU}_{n,\mathbb{N}}(s) = h^*(s)$ if the length of the cheapest path from s to a goal state is less than or equal to n.*

Proof. Given any integer n times time synchronized flow $F^{\mathcal{T}}_n$. h^* is equal to the cost of the cheapest path and with the theorem's condition as well as Theorem 7 we know that for the cheapest path π_c there also is an integer time synchronized flow F_c. $h^{ATU}_{n,\mathbb{N}}(s)$ is equal to the cost of F_c, because if there would be a cheaper flow, there would also be a cheaper plan (see Theorem 14), which contradicts with π_c being the cheapest plan. □

We furthermore prove that h^{MATU} is a perfect heuristic if the planning task has unit costs.

Theorem 16. *Given a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_I, s_G, cost \rangle$, $h^{MATU}_{\mathbb{N}} = h^*$ if $cost(o) = 1$ for all $o \in \mathcal{O}$.*

Proof. $h^{MATU}_{\mathbb{N}}$ is defined as $h^{ATU}_{n,\mathbb{N}}$ where $n = \text{argmin}_n\, h^{ATU}_{n,\mathbb{N}} \neq \infty$. With Theorem 15 we know that $h^{ATU}_{n,\mathbb{N}}$ is either equal to the cheapest synchronized flow, or there is no synchronized flow for any number of time steps.
In the case where there is a cheapest synchronized flow we know with Theorem 9 that $h^{ATU}_{n,\mathbb{N}}$ is equal to the cost of a plan, this plan also has to be the cheapest, because if there would be a cheaper plan, there would also be a cheaper synchronized flow, as in the unit cost case a cheaper plan also means that it is shorter and still preserved with n time steps.
The case were there is no number of time steps for which there is a synchronized flow, we know with Theorem 7, that there are no plans i.e. $h^{ATU}_{n,\mathbb{N}} = h^* = \infty$. □

For showing dominance relationships between the time heuristics the following theorem will be introduced.

Theorem 17. *Given a transition system $\mathcal{T} = \langle \mathcal{S}, \mathcal{O}, cost, T, s_I, S_G \rangle$ with $S_G = \{s^1_G, \ldots, s^l_G\}$. For every flow in $\mathcal{T}^{\tau^{(r)}_n}$ there is a flow in $\mathcal{T}^{\tau^r_k}$ of same cost if $k \leq n$. Analogously for every time n times time synchronized flow in a set of transition systems $\boldsymbol{\mathcal{T}}_n = \{(\mathcal{T}_1^{\tau^{(r)}_n}, \ldots, \mathcal{T}_w^{\tau^{(r)}_n}\}$ a k times time synchronized flow with the same cost exists in $\boldsymbol{\mathcal{T}}'_k = \{(\mathcal{T}_1^{\tau^r_k}, \ldots, \mathcal{T}_w^{\tau^r_k}\}$.*

Proof. Given goal variables $\boldsymbol{g} = \{g^1, \ldots, g^l\}$, a flow F in $\mathcal{T}^{\tau^{(r)}_n}$ and any transition $t = \langle s_1, o, s_2 \rangle$ out of the transition set of $\mathcal{T}^{\tau^r_k}$, we define the mapping M as

$$M(t) = \begin{cases} \sum_{i=n,\ldots,k-1(,k)} F(t_i) \text{ where } t_i = \langle s_i, o, s' \rangle & \text{if } time(s_1) = n \\ F(t) & \text{if } time(s_1) \leq n \end{cases}$$

$$M(g^m_t) = \begin{cases} \sum_{i=n,\ldots,k-1(,k)} F(g^m_i) & \text{if } t = n \\ F(g^m_t) & \text{if } t \leq n \end{cases}$$

where g^m_t denotes the goal variable corresponding to the goal state of g^m in the t-th time layer of $\mathcal{T}^{\tau^r_k}$. As the flow values of transitions and goal variables are not different in the first n time layers, the flow constraints are fulfilled for all states $s \in \mathcal{T}^{\tau^r_k}$ with $time(s) < n$. For all states s'_n it holds that the flow value of a outgoing transition is equal to the flow value of the corresponding transitions of

the states s'_i where $i \in \{k, \ldots, n\}$. This is similarly true for incoming transitions and goal variables. As the flow constraints are fulfilled for every state s'_i, the flow constraints are fulfilled for s'_n as well.
In the same way a mapping M_2 can be built from a time synchronized flow $F_n^{\mathcal{T}_n}$ and we already now that the flow constraints are fulfilled. Furthermore the time synchronization constraints and the operator flow constraints are fulfilled as well because $\mathit{otflow}_o^{=n}(T_w^{\tau_k^r})$ of M_2 is equal to the sum of $\mathit{otflow}_o^{=i}(T_w^{\tau_n^{(r)}})$ with $i \in \{n, \ldots, k\}$ and as all the $\mathit{otflow}_o^{=i}(T_w^{\tau_n^{(r)}})$ are synchronized individually, $\mathit{otflow}_o^{=n}(T_w^{\tau_k^r})$ are synchronized as well. □

The theorem can now be used to show dominance relationships.

Theorem 18. *$h_n^{ATUR} \leq h_k^{ATUR}$ if $n \leq k$ and $h_k^{ATUR} \leq h^{MATU}$.*

Proof. Both dominance relationships follow directly from Theorem 17 and the fact that h_k^{ATUR} and h^{MATU} are defined via the *cheapest* time synchronized flows.

□

There is the question whether TU and TUR should introduce goal states in every time layer or just a subset of time layers. As goal states act as sinks for flows, reducing the amount of goal states decreases the degrees of freedom of flows. However with the idea of the proof of Theorem 4 we know that reducing the goal states to a set of time layers T, would result in the time unrolled transition system only preserving plans that have length n, if $n \in T$. Therefore limiting the goal states holds the danger, that cheapest plans are no longer preserved, which affects admissibility of heuristics that build up on the time unrolling. If a lower bound for plans is known, it would be possible to omit goal states in time layers that are lower than the lower bound while still preserving the cheapest plan. We can define a new heuristic h_*^{MATU} that is defined just as h^{MATU} but the time unrolling only introduces goal states in the last time layer. $h_{n,*}^{ATU}$ is defined analogously. The reason why h_*^{MATU} is still admissible in the unit cost case is that it still holds that $h_{n,*}^{ATU} \leq n$ for all n. In the special case of goal state only being present in the last time layer the inequality becomes an equality, because there is no sink in the first $n-1$ time layers, this however is not even required for admissibility. With Theorem 7 we know that there is a time synchronized flow for every plan that only uses goal states in the last time layer as sinks. As h_*^{MATU} is per definition equal to the first $h_{n,*}^{ATU} = n$ that is not equal to ∞ it is admissible. There will be no experiments done on h_*^{MATU} due to time constraints of this thesis.

4 Implementation

There are two types of heuristics implemented, one that uses a fixed amount of time steps and one that iteratively increases time steps.

4.1 Implementation with fixed amount of time steps

This version is an implementation of the n times atomic time unrolling heuristic h_n^{ATUR} (note the time unrolling with repetition) with a fixed n. What makes this type of implementation advantageous (compared to the h^{MATU} implementation) is the fact that with a fixed number of time steps the only thing that changes in the LP, with each heuristic computation, are the start states of the time unrolled atomic projections. That means that there are at most $2 \cdot |\mathcal{V}|$ constraints to be changed with each new heuristic computation.
An advantage of working with repetitions in the last layer is that the heuristic is admissible in all cases (see Theorem 9).

4.2 Implementation with dynamic amount of time steps

This version is an implementation of h^{MATU} with the help of h_n^{ATU} with a changing n value. From Theorem 7, we know that if there is no flow found for $n = x$, we know that there does not exist plans with a length equal to or smaller than x. The idea is to start with 0 time steps and then build up as many time steps are needed until a solution is found. This method is therefore adjusting the quality of the heuristic depending on how long the plans are (in contrast to the static implementation). Furthermore dead ends are a problem for this method, as they force an perpetual increase of time steps, making the implementation exceed the memory or time limit during the experiments.

4.3 Optimizations

There were two optimizations considered.

4.3.1 Removing dead states

With Theorem 13 we know that leaving out dead states can not lower heuristic values, but also leaving them out means that there are less LP variables (the LP contains a variable for every transition) and also fewer constraints (the LP contains a constraint for every state). This optimization was only implemented for the dynamic time steps approach, as in the fixed time step case would lead to a need for changing constraints of the LP (the forward-reachability of states in the early time layers changes with every heuristic calculation), that would require a fully reload of the LP into the LP solver with every heuristic value calculation, which leads to a big overhead.

4.3.2 Introducing shared variables

The idea of introducing shared variables is used for both implementations. If there are operators, that have multiple variables, there are multiple $\mathit{otflow}_o^{=t}$

with only one variable. With a fixed t these have to be equal, due to the time synchronization conditions. What we can do is using the same variables for a fixed t if there are multiple $otflow_o^{=t}$ with only one variable. This would remove the requirement to force them to be equal with constraints. This does not only decrease the constraints of the LP, but also the variable count. Note that $otflow_o^{=t}$ of the flow of a time unrolled atomic projection consists of a single variable if the corresponding variable of the atmoic projection is in the preconditions of o. That means that there are saved $n - 1$ constraints and variables for each time step that has to be synchronized, where $n = \sum_o |vars(o)|$.

5 Experiments

The experiments were conducted on the computing cluster maia of the University of Basel. The tasks were run on cores of the Intel® Xeon® CPU E5-2660 at 2.20 GHz. All problem instances were run with a memory limit of 2 GB and a time limit of 30 minutes. The problem domains and instances used were composed from optimal tracks of the IPC from 1998 up until 2014. An automatization tool called downward-lab (Seipp et al., 2017) was used to easier conduct experiments on the cluster and also evaluate them by creating tables and graphs. As LP solver IBM ILOG CPLEX Optimization Studio 12.6.3 was used. The implementation was integrated into the planning system Fast Downward (Helmert, 2006), so that not everything had to be build up from scratch and the implementations could be easily compared to other already implemented heuristics. The coverage of a heuristic is defined as the number of problems solved in all benchmark domains. If numbers in tables are bold they are equal to the maximum row value. The *initial h value* is equal to the heuristic value of the initial state. We will discuss the static number of time steps and the dynamic number of time steps approach in terms of initial h value quality and coverage. Both, the static and the dynamic version compete against the state-of-the-art heuristic LM-cut (Helmert and Domshlak, 2009) and the state equation heuristic h^{SEQ}, which is also interesting, as the time unrolling heuristics dominate it, but it is not clear by how much. As operator costs were integers only we were able to round up heuristic values if they were not integers. Also note that benchmark results are usually rounded to the nearest integer. For analyzing initial heuristic values we are going to normalize them with the perfect heuristic. To get to perfect heuristic values we took the plan costs of $h^{LM\text{-}cut}$. This also means that initial heuristic values could only be normalized if $h^{LM\text{-}cut}$ was able to solve the problem.

5.1 Static number of time steps

As already stated in the optimization section we tested what influence introducing shared variables have on the implementation of h_n^{ATUR}. It turned out that the optimization is beneficial, which is also the reason that all benchmark results in the following sub-sections are based on an implementation with this optimization. The benchmarks of h_n^{ATUR} with $n \in \{0, 2, 4, 6\}$ were however conducted with and without the optimization to analyze the effect of the optimization, what we will do in the following.
We first look at how many LP variables and LP constraints are needed to calculate the heuristic value of the initial state. The optimization reduced the number of constraints on average by 6 %. Whereas the variables were only reduced by 2 %. Remember that for every saved variable there is also a saved time synchronization constraint. The percentages are just different as there are around three times as many variables as constraints. The reduced number of variables and constraints, reduced the solving time of the LP by 14 % on average. This leads to a average improvement of 15 more solved tasks. Figure 8 shows that there is an improvement for any number of time step that was tested.

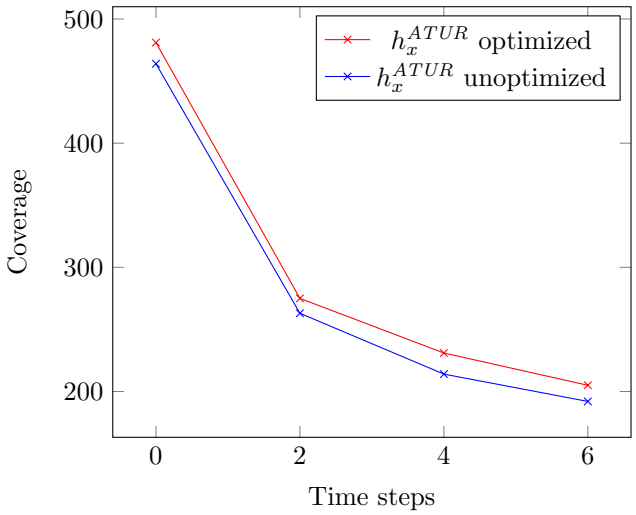

Figure 8: Plot depicts coverage of h_x^{ATUR} with and without shared variable optimization. We can see that there is an improvement for any number of time steps.

5.1.1 Initial heuristic value

We want to compare the quality of heuristic values of h_n^{ATUR} with different values for n against the quality of h^{SEQ} and $h^{LM\text{-}cut}$. The summed initial h values of $h^{LM\text{-}cut}$, h^{SEQ}, h_0^{ATUR}, h_2^{ATUR}, h_4^{ATUR} and h_6^{ATUR} are shown in Table 1. The summed initial heuristic values of h^{SEQ} and h_0^{ATUR} as expected from Theorem 10. We can furthermore see that adding more time steps almost always comes with an increase of heuristic values. There are multiple domains, for instance pegsol and visitall, were the time steps could improve summed values up until they got higher than LM-cut's summed values. Also note that sometimes the benefit of adding additional time steps (which has an dramatic effect on the coverage, see Table 2) can lead to only a minor increase of heuristic value like for the Sokoban domains. Where in some domains like psr-small or mprime the additional time steps lead to significantly better heuristic values.

We want to know much the h^{SEQ} heuristic values can be improved by increasing the time steps. Figure 9 depicts normalized initial h values, where a 1 means being equal to the perfect heuristic value. The perfect heuristic value was acquired by doing A^* with $h^{LM\text{-}cut}$. If $h^{LM\text{-}cut}$ did not solve the problem, there is no point for the respective problem. This means that this comparison was only done for problems that were easy for *LM-cut*. Furthermore whenever we state average improvements in heuristic value they will be calculated via the average difference of normalized initial heuristic values. If for instance the average improvement is x then in our graphs this usually means that a data point is shifted on average x to the right, starting from the diagonal. We can see that all points are in the bottom right triangle, which means that the values of h^{SEQ} were never better than the values of h_2^{ATUR} and h_6^{ATUR}, which confirms our theoretical analysis. We can see that already introducing two time

Initial h value	h^{LM-cut}	h^{SEQ}	h_0^{ATUR}	h_2^{ATUR}	h_4^{ATUR}	h_6^{ATUR}
airport (16)	**602**	383	383	389	395	401
barman-opt11 (4)	141	144	144	**148**	**148**	**148**
blocks (34)	574	570	570	601	627	**649**
childsnack-opt14 (1)	10	**18**	**18**	**18**	**18**	**18**
depot (8)	**124**	85	85	89	98	100
driverlog (14)	**195**	162	162	163	169	175
elevators-opt08 (30)	**1025**	0	0	33	33	33
elevators-opt11 (20)	**674**	0	0	21	21	21
floortile-opt11 (16)	**846**	702	702	703	705	707
floortile-opt14 (20)	**1143**	916	916	916	919	924
freecell (8)	37	80	80	80	**81**	**81**
ged-opt14 (15)	14	0	0	**15**	**15**	**15**
gripper (20)	**940**	920	920	920	**940**	**940**
hiking-opt14 (10)	**94**	50	50	70	75	79
logistics00 (28)	**1041**	888	888	888	891	898
logistics98 (7)	**170**	122	122	122	126	129
miconic (103)	**3641**	2242	2242	2339	2352	2357
movie (30)	**210**	**210**	**210**	**210**	**210**	**210**
mprime (5)	**25**	14	14	19	**25**	**25**
mystery (11)	43	24	24	33	44	**45**
nomystery-opt11 (11)	**160**	116	116	125	129	132
openstacks (7)	**117**	90	90	97	104	111
openstacks-opt08 (27)	**27**	0	0	**27**	**27**	**27**
openstacks-opt11 (20)	**20**	0	0	**20**	**20**	**20**
openstacks-opt14 (8)	8	0	0	8	8	**9**
pathways-noneg (5)	**77**	16	16	26	36	40
pegsol-08 (30)	76	87	87	92	96	**103**
pegsol-opt11 (20)	59	72	72	72	76	**79**
pipesworld-notankage (4)	23	20	20	22	**24**	**24**
pipesworld-tankage (6)	33	44	44	44	44	**45**
psr-small (50)	157	310	310	390	418	**432**
rovers (16)	**287**	118	118	140	148	153
satellite (7)	**94**	44	44	53	61	63
scanalyzer-08 (7)	117	114	114	117	121	**122**
scanalyzer-opt11 (5)	100	96	96	97	101	**103**
sokoban-opt08 (27)	**385**	134	134	134	134	134
sokoban-opt11 (19)	**280**	92	92	92	92	92
storage (15)	**118**	90	90	113	**118**	**118**
tpp (12)	261	312	312	325	326	**330**
transport-opt08 (15)	**3999**	120	120	581	797	928
transport-opt11 (15)	**4102**	172	172	438	594	699
transport-opt14 (9)	**2183**	78	78	285	422	518
trucks (8)	146	138	138	143	148	**149**
visitall-opt11 (20)	542	799	799	800	801	**802**
visitall-opt14 (16)	774	1193	1193	1193	1193	**1194**
woodworking-opt08 (11)	**1860**	1565	1565	1665	1717	1726
woodworking-opt11 (6)	**1155**	1035	1035	1095	1121	1128
zenotravel (10)	**96**	75	75	79	83	84
Finite sum (806)	**28805**	14460	14460	16050	16851	17320

Table 1: Initial heuristic values of the static time step approach with 0, 2, 4 and 6 time steps versus the two other competitors. Note that only instances are counted, were every heuristic got an initial h value calculated. Note that the parcprinter domain results are ignored, due to their very high operator costs, that would heavily skew the sums.

steps leads to improved heuristic values (points that do not lie on the diagonal). There were also some cases where h_2^{ATUR} improved heuristic values of h^{SEQ} to the perfect value (represented by points that lie at the gray perpendicular line). The values got improved in 380 out of the 788 data points. On average the normalized value got improved by 6 % (with the non-improvements counted). Improvements were recorded in 38 out of the 57 domains, which means that the heuristic improvements are not limited to only few domain structures.

With the right graph in Figure 9 we can check whether heuristic values could be further improved when introducing more time layers or whether the most improvement was already gathered with 2 time steps. It turns out that increasing the time steps form 2 to 6 brings further improved heuristic values for a lot of problem instances (there are many points shifted more to the right and the points on the diagonal are not as crowded anymore). There are also a lot more problems where the heuristic got improved to the perfect value. The number of problems where the heuristic value got improved changed from 380 out of 788 problems to 444 out of 686 problems. There were also 6 new domains where the heuristic value got improved, which means that there are now 44 out of the 57 domains with improved values. The heuristic value got improved by an average of 10 %. Note how we have tripled the time steps, but only got a 4 % over the initial 6 % with only 2 time steps, which shows that there are diminishing returns of further time steps.

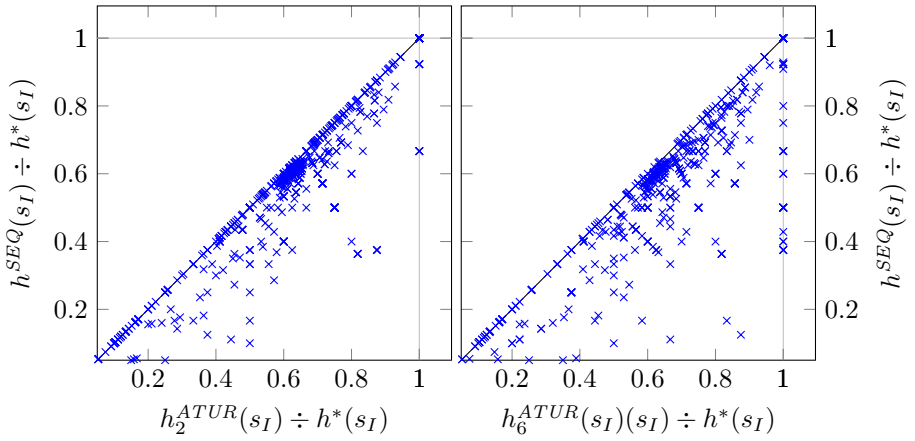

Figure 9: The plots compare normalized initial heuristic values of h_2^{ATUR} and h_6^{ATUR} against the initial value of h^{SEQ}. We can see that increasing the time steps leads overall improvement and also to more perfect values.

We want to know how far we can improve the heuristic values with increasing the number of time steps and whether the diminishing returns can be confirmed. Figure 10 depicts two scatter plots. The left shows the improvement from going from 6 time steps to 10 and the right from 10 to 15. We can see that going from 6 to 10 time steps there is still a noticeable shift to the right. Where for the step from 10 to 15 this is also true, there are far less problems where the

heuristic value got increased (the number of improvements went from 95 out of 568 problems to only 44 out of 461 problems). When increasing the time steps from 6 to 10, we get on average only 1 % improvement (this is not much if we compare it to the 4 % improvement going from 2 to 6 time steps, where also the number of time steps got increased by 4). The number of improvements shrunk to only 95 problems out of the 568, where the initial h value could be computed. If we go from 10 to 15 time steps the improvement was on average only 0.6 %. This shows again that there are diminishing-returns. Also note that the diminishing returns that can be seen in the averages were also caused by the fact that heuristic values were perfect already and could not be improved further. In the 4 experiments the number of already perfect values (represented by points in the upper right corner) are 98, 91, 110 and 106 respectively, whilst the number of data points went down from 788 in the h^{SEQ} vs h_2^{ATUR} plot to 461 in the h_{10}^{ATUR} vs h_{15}^{ATUR} plot. The reduction of data points is a result of the fact that with increasing number of time steps there is more computation needed for the initial heuristic value, because the LP gets bigger.
Benchmarking was also done with 20, 50, 100 and 200 time steps and it turned out that every number of time steps over 20 did not result in better heuristic values. Note that there are problems where the length of the cheapest plan is bigger than 20 where h_{50}^{ATUR} has an initial heuristic value. So the stagnation in heuristical quality is not only due to cheapest plans being shorter than the used number of time steps.
If we compare h_{20}^{ATUR} against h^{SEQ} there are only 31 problems that could not be improved that were not perfect already (out of 367 in total and 68 already being perfect). There are still some domains where h_{20}^{ATUR} has lower values than *LM-cut*, for instance in the Transport domain. This shows that the heuristic cannot be improved after a certain point. This also seems reasonable if we consider, that time unrolling does not change the first $n-1$ time layers. And if there is a solution that does not have a flow in the last time layer, it will persist into time unrolled transition systems with more time steps.

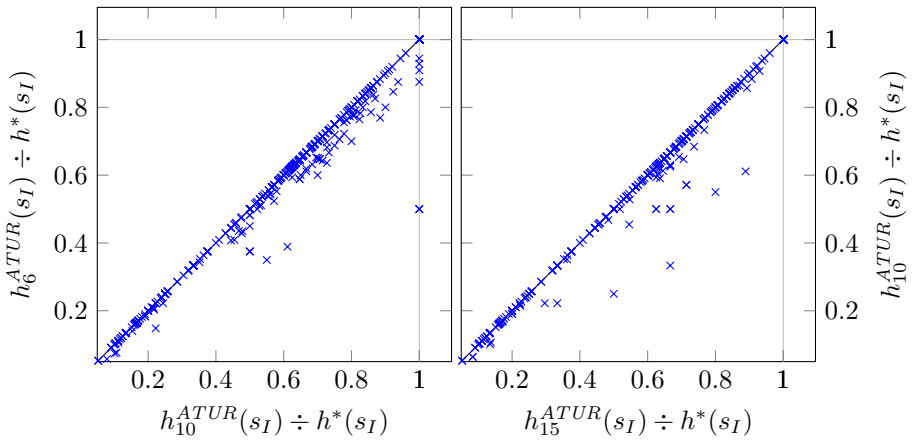

Figure 10: The plots compare normalized initial heuristic values of h_6^{ATUR}, h_{10}^{ATUR} and h_{15}^{ATUR}. We can see that increasing time steps leads to fewer and smaller heuristic value improvements.

We now want to measure how IP versions perform. Figure 11 compares h_2^{ATUR} and h_6^{ATUR} to their IP counterparts. In the left plot we can see that, when using only 2 time steps, restricting to IP variables leads to only few improvements and even fewer heuristic values got improved to the perfect value. The average improvement was an additional 2 % which come from an improvement in only 68 out of 785 problems. If we increase time steps to 6, there we can clearly see that there are way more and way better improvements. With 6 time steps there is an average increase of 5.5 % coming from 263 problems with increased values out of 600 problems in total. This shows that restricting to integers becomes more beneficial with more time steps.
If we compare the time it took the LP and the IP version to compute the initial h values, we can see that calculating h_2^{ATUR} on average takes about 34 % of the time compared to its IP counterpart. Calculation time for h_6^{ATUR} on average was around only 22 % of the time of its IP counterpart. This means that, in the case of 2 time steps, the IP version took roughly 3 times the time and in the case of 6 time steps around 4 times.

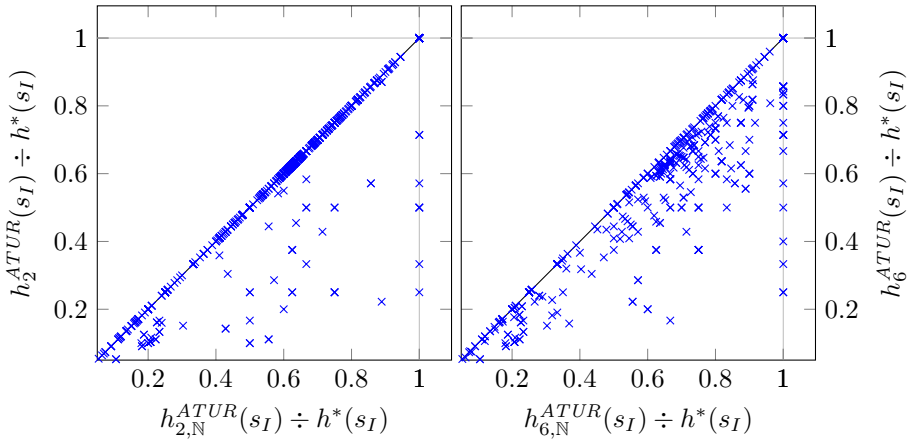

Figure 11: The plots compare normalized initial heuristic values of h_2^{ATUR}, h_6^{ATUR} with their IP counterparts. We can see that using IP variables leads to bigger heuristic value improvements with more time steps.

5.1.2 Coverage

Table 2 shows the coverage of h^{LM-cut}, h^{SEQ}, h_0^{ATUR}, h_2^{ATUR}, h_4^{ATUR} and h_6^{ATUR}. On first glance we can see that the time heuristics cannot keep up, as they generally have less solved problems. Increasing time steps introduces an overhead, that does not seem to pay off through better heuristic values. Also note that h^{SEQ} and h_0^{ATUR} are the same heuristics, with the only difference being the implementation and we can see that the Fast Downward's version has better coverage. This could be explained with the implementation of h_0^{ATUR} being very poorly optimized and the implementation of h^{SEQ} omitting the upper bound constraints of the flow constraints for each state. It has been shown that the upper bound constraints of the flow constraints for h_0^{ATUR} can be omitted without changing the heuristic value (Pommerening et al., 2014). The only do-

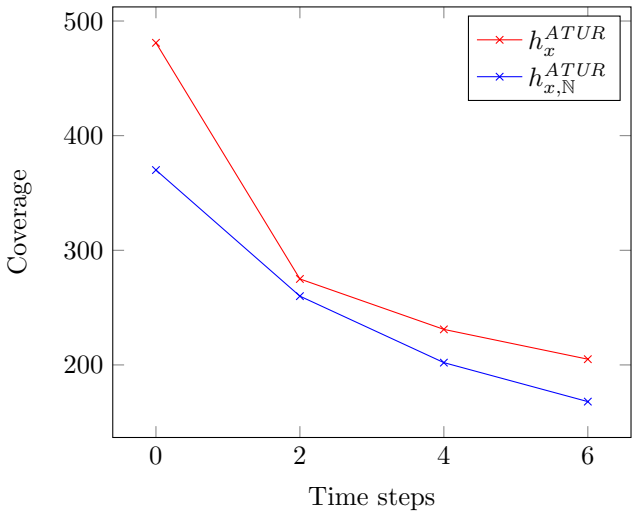

Figure 12: Plot depicts coverage of h_x^{ATUR} and $h_{x,\mathbb{N}}^{ATUR}$ with variable time steps. We can see that the IP coverage is consistently inferior. Introducing few time steps can however lessen the gap.

mains were the time heuristics could almost keep up with LM-cut (for instance Freecell or visitall-opt14-strips) were also the domains were h^{SEQ} surpasses LM-cut. This means that the time heuristics exploits problem structure which are also exploited by the state equation heuristic, which is also reasonable as the time heuristics are an extension of the state equation heuristic. The extension however, at least evaluated on the coverage, does not pay off. This can be explained by the diminishing returns of adding further time steps, which we have seen in the initial h value section.
We now want to see what influence solving the IP versions has on the coverage. Figure 12 depicts coverage of h_0^{ATUR}, h_2^{ATUR}, h_4^{ATUR} and h_6^{ATUR} in red against their IP counterparts in blue. We can see that the IP versions have worse coverage in all cases. The difference in coverage are 111, 15, 29 and 37 respectively. The biggest difference is with no time steps at all. The minimum gap is either with 1 or 2 time steps. Increasing time steps over 2 increases the gap again. This seems to contradict with the improve in heuristic value that be seen in Figure 11. The bigger increase in heuristic values does therefore come with a time overhead, which has more influence on the coverage than the increased heuristic values.
The average time spent calculating initial h values of the LP versions took only 30 % of the time of their IP counterparts. Apparently this overhead, of more than triple the calculation time on average does not pay off in terms of coverage.

5.2 Dynamic number of time steps

With Theorem 12 we know that h^{MATU} is only admissible if the planning task has unit costs. Therefore we will enforce unit costs of our benchmark domains when testing this implementation.

Coverage (Number of Instances)	h^{LM-cut}	h^{SEQ}	h_0^{ATUR}	h_2^{ATUR}	h_4^{ATUR}	h_6^{ATUR}
airport (50)	**28**	22	14	7	7	7
barman-opt11 (20)	**4**	**4**	0	0	0	0
barman-opt14 (14)	**0**	**0**	**0**	**0**	**0**	**0**
blocks (35)	**28**	**28**	21	15	13	12
childsnack-opt14 (20)	**0**	**0**	**0**	**0**	**0**	**0**
depot (22)	**7**	**7**	2	2	1	1
driverlog (20)	**13**	12	8	3	2	2
elevators-opt08 (30)	**22**	9	3	0	0	0
elevators-opt11 (20)	**18**	7	1	0	0	0
floortile-opt11 (20)	**7**	4	2	0	0	0
floortile-opt14 (20)	**6**	2	0	0	0	0
freecell (80)	15	**37**	20	1	1	0
ged-opt14 (20)	**15**	13	7	5	5	3
grid (5)	**2**	1	1	0	0	0
gripper (20)	**7**	**7**	5	3	2	2
hiking-opt14 (20)	**9**	**9**	7	3	2	2
logistics00 (28)	**20**	16	12	10	6	6
logistics98 (35)	**6**	4	2	2	0	0
miconic (150)	**141**	51	40	30	26	25
movie (30)	**30**	**30**	**30**	**30**	**30**	**30**
mprime (35)	**22**	20	9	1	1	1
mystery (30)	**17**	13	8	4	4	4
nomystery-opt11 (20)	**14**	10	8	2	2	2
openstacks (30)	**7**	**7**	5	5	5	0
openstacks-opt08 (30)	**19**	16	8	3	2	2
openstacks-opt11 (20)	**14**	11	3	0	0	0
openstacks-opt14 (20)	**3**	1	0	0	0	0
parcprinter-08 (30)	18	**28**	**28**	17	11	8
parcprinter-opt11 (20)	13	**20**	**20**	13	7	4
parking-opt11 (20)	2	**3**	0	0	0	0
parking-opt14 (20)	**3**	**3**	0	0	0	0
pathways-noneg (30)	**5**	4	4	2	1	1
pegsol-08 (30)	27	**28**	25	10	9	6
pegsol-opt11 (20)	17	**18**	15	2	1	1
pipesworld-notankage (50)	**17**	15	8	2	1	1
pipesworld-tankage (50)	**12**	11	7	2	2	2
psr-small (50)	49	**50**	47	42	40	38
rovers (40)	**7**	6	4	4	4	4
satellite (36)	**7**	6	4	3	3	3
scanalyzer-08 (30)	**15**	14	8	4	4	3
scanalyzer-opt11 (20)	**12**	11	5	1	1	1
sokoban-opt08 (30)	**30**	19	7	3	1	0
sokoban-opt11 (20)	**20**	16	4	0	0	0
storage (30)	**15**	**15**	12	7	7	7
tetris-opt14 (17)	6	**12**	2	0	0	0
tidybot-opt11 (20)	**13**	6	1	0	0	0
tidybot-opt14 (20)	**7**	0	0	0	0	0
tpp (30)	6	**8**	6	5	5	5
transport-opt08 (30)	**11**	**11**	7	4	3	3
transport-opt11 (20)	**6**	**6**	2	0	0	0
transport-opt14 (20)	**6**	4	1	0	0	0
trucks (30)	**10**	9	4	2	1	1
visitall-opt11 (20)	11	**17**	16	12	9	9
visitall-opt14 (20)	5	**13**	11	6	4	3
woodworking-opt08 (30)	**17**	13	7	3	3	2
woodworking-opt11 (20)	**12**	9	2	0	0	0
zenotravel (20)	**13**	9	8	5	5	4
Sum (1667)	**866**	725	481	275	231	205

Table 2: Coverage of h^{LM-cut}, h^{SEQ}, h_0^{ATUR}, h_2^{ATUR}, h_4^{ATUR} and h_6^{ATUR}. The bigger the time steps of h_n^{ATUR} are chosen, the smaller the coverage gets.

When there is no flow found for the first t time steps (i.e. $h_i^{ATU} = \infty$ for all $i \in \{1, \ldots, t\}$), we know that there are no plans of length less than or equal to t (see Theorem 7). Therefore we will use $max(h^{MATU}, t)$ where $t = \text{argmin}_i\, h_i^{ATU} \neq \infty$ as heuristic value. An example showing that this can actually lead to an improvement is the problem "p01" in the Sokoban domain from 2008, where this increases the initial heuristic value from 14 to 16.
As baseline for comparisons, also in Section 5.2.1 and 5.2.2, the algorithm was tested with both optimizations as described in Section 4.3. They were however tested individually.
We will now analyze what influence the optimizations have on the LP computed for the initial heuristic value.
The dead states removal reduced the number of constraints by 2 % and variables by 17 %. This lead to a 9 % peak memory reduction and 19 % less time spent calculating the first heuristic value. The coverage however only went up from 158 to 165 problems.
Introducing shared variables decreased the number of constraints by 11 % and variables by 4 %. Again, as constraints and variables get decreased by the same amount the percentage difference results from having almost 3 times as many variables than constraints. Peak memory got reduced by 3 % and time spent calculating the first heuristic value reduced by 11 %. The result on the coverage is very similar to the dead state optimization – it went up from 158 to 166 problems.
The last comparison is a comparison between both optimization enabled and none. There were 11 % fewer constraints, 20 % fewer variables, 12 % less peak memory and 42 % less time spent solving the first LP with a solution. The resulting coverage is 174.

5.2.1 Initial heuristic value

How h^{MATU} compares against h^{SEQ} and $h^{LM\text{-}cut}$ can be seen in Figure 13. In the left plot we can see that h^{MATU} dominates h^{SEQ} as there is no point beyond the diagonal. Furthermore almost all values of h^{SEQ} could be further improved. If we compare h^{MATU} with $h^{LM\text{-}cut}$, we can see that h^{MATU} can improve on problems where $h^{LM\text{-}cut}$ is particularly bad, whereas $h^{LM\text{-}cut}$ mostly improves where values of h^{MATU} are already pretty good.
In both plots we can see that points move along diagonals that meet in certain points. These patterns however do not stem from domain or heuristic properties, but they are coming from the distribution of a set of rational numbers in a real grid. This set is defined by rationals that can be expressed as fractions with the same denominator. Both coordinates x and y are coming from integer divisions, because heuristic values are rounded up. Furthermore the denominator is the same as h^* stays fixed as one data point represents one problem. In the left plot we can see the pattern even stronger as there is a further condition. One numerator is bigger than the other, i.e. h^{MATU} dominates h^{SEQ} which makes the points less spread and the pattern more apparent. Appendix A shows a plot of only rational numbers with same denominator that shows exactly the same patterns of diagonals and sparse regions. The only difference is that in the heuristic plots they are crowded depending on the quality of the heuristic values.
The left plot in Figure 14 confirms that $h_{\mathbb{N}}^{MATU}$ is in fact equal to the perfect

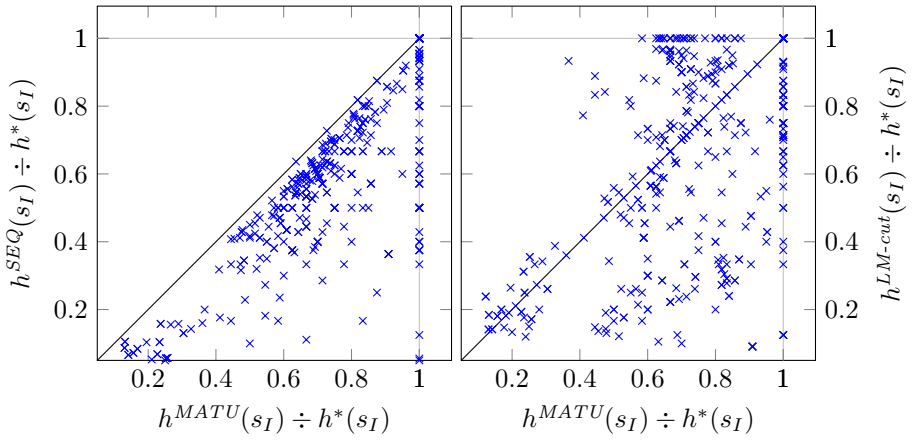

Figure 13: Plots show normalized heuristic values from h^{MATU} against h^{SEQ} and $h^{LM\text{-}cut}$. We can see that h^{MATU} dominates h^{SEQ} and values are improved in almost all times. Whereas the plot with the state of the art heuristic $h^{LM\text{-}cut}$ seems a little more balanced, however h^{MATU} has higher values in most cases.

heuristic, at least in these cases. The number of time steps it took to calculate $h^{MATU}_{\mathbb{N}}$ is equal to the perfect heuristic value. This follows from the fact that $h^{MATU}_{\mathbb{N}}$ is the perfect heuristic and that there can only be a maximum flow of 1 for every time layer (see Theorem 11). For the implementation of the LP version, the average time steps needed for the first heuristic value is only 11, while the average plan length is 21. This shows that there is still a quite big discrepancy between them in terms of utilizing additional time steps.
The right plot confirms that h^{MATU} dominates h^{ATUR}_n. The average increase in initial heuristic value was 3 %. h^{MATU} had higher values in 148 out of 377 problems.

5.2.2 Coverage

Table 3 shows that h^{MATU} and $h^{MATU}_{\mathbb{N}}$ solve far less tasks than $h^{LM\text{-}cut}$ and h^{SEQ}, in fact they solved only around one fifth amount of tasks. One factor that influences the search time is the fact that every time the implementation increases a time step, a new LP is created and loaded into memory. This is an overhead compared to just adding the new variables and new constraints. The frequent LP recreation can also be seen in the calculation time of the initial heuristic value which made up from 30 % initializing the LP (70 % of the time was spent solving it) and for the IP version initializing was even 33 % of the time. It also might not be beneficial to let the solver solve dozens of LPs, it seems to be more beneficial to use the LP solver for just one big LP for each heuristic computation (see coverage results of h^{ATUR}_n). On average for every heuristic value 10 time steps were needed until a solution was found. This means on average 10 there were LPs created loaded and solved for every heuristic value calculated.

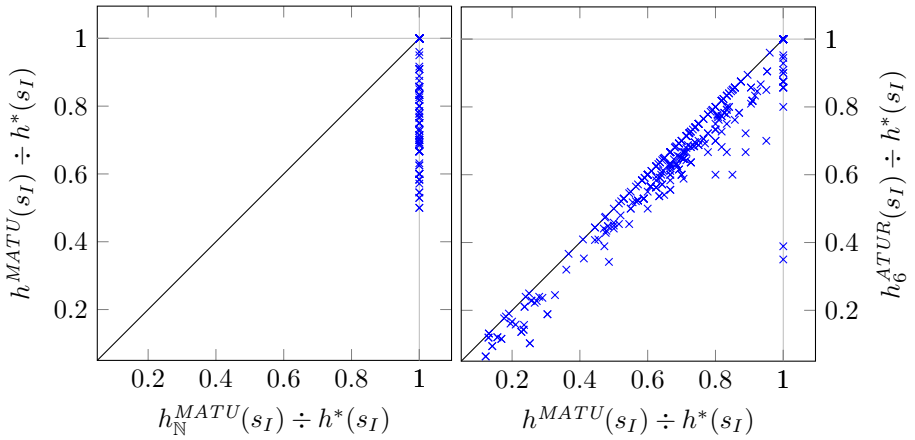

Figure 14: The left plot shows normalized initial heuristic values of h^{MATU} and $h^{MATU}_{\mathbb{N}}$. We can see that $h^{MATU}_{\mathbb{N}}$ is always equal to the perfect heuristic and h^{MATU} dominates h^{ATUR}_6.

	Coverage	Coverage ÷ Coverage of h^{MATU}
$h^{LM\text{-}cut}$	832	4.8
h^{SEQ}	757	4.4
h^{MATU}	174	1
$h^{MATU}_{\mathbb{N}}$	127	0.7

Table 3: This table is showing the coverage of $h^{LM\text{-}cut}$, h^{SEQ}, h^{MATU} and $h^{MATU}_{\mathbb{N}}$. Both h^{MATU} and $h^{MATU}_{\mathbb{N}}$ perform far worse.

6 Future Work

The constraints of the time synchronized flows fit very well into the operator counting framework (Pommerening et al., 2014). For every operator o there can be introduced a new variable $Count_o$ and a constraint $Count_o = otflow_o^{>0}$. We are then able to combine our time synchronized flow constraints with other types of constraints for instance the landmark constraints that result from landmarks found by LM-cut. In our experiments we have seen that even for our LP time unrolling heuristic with the best initial heuristic values h^{MATU} there are many benchmark problems were the initial heuristic value is lower than the initial heuristic value of LM-cut, therefore combining them might result in a strong heuristic.
The time unrolling heuristics work with arbitrary abstractions. In this thesis we have benchmarked the heuristics that resulted when fixing the abstractions to atomic projections. But it could also be tested with bigger projections. Bigger projections would of course lead to bigger abstract transition systems. h^{MATU} might not be well suited for this, because it often uses many time steps, when we have seen in our experiments that most benefit could already be gathered with introducing few time steps. The time steps of h_n^{ATUR} however can be set accordingly lower if bigger projections are chosen.
There might be more efficient ways to get to n times time synchronizable transition systems than by using time unrollings. The time unrollings have the states copied into every time layer. This however might not be necessary. If there is a state s in a transition system that already fulfills the requirement that all paths to it have length $d^*(s)$ it might just only be represented in the $d^*(s)$-th time layer.

7 Conclusion

In this thesis we first observed that cycles in transition system can be a hindrance for flow heuristics, leading to lower heuristic values. We then established time unrolling with which we were able to get rid of the cycles. When using time unrolling there might be plans lost in the process, which lead us to time unrolling with repetition. We found out that the flow constraints of multiple abstract transition systems can be combined more strongly if they are time unrolled, or more generally, time synchronizable. If the abstract transition systems are t times time synchronizable we can synchronize the summed flow for each operator for the first t time steps. On these concepts we built general heuristics that combine the information of multiple abstract transition systems with time synchronized flows.
Fixing the abstractions to atomic projections lead us to heuristics that use time unrolling with a fixed amount of time steps h_n^{ATUR} and the heuristic h^{MATU} with a number of time steps depending on the problem. In our experiments we found out that h_n^{ATUR} can either have a focus on solving many problems or on having good heuristic values or even a combination of both. There are however diminishing returns when increasing the time steps. h^{MATU} on the other hand is in general a heuristic with pretty strong heuristic values as usually a lot time steps are used. This means that h_n^{ATUR} is preferred over h^{MATU} if coverage is of importance. Finally it holds that $h_{\mathbb{N}}^{MATU} = h^*$ if there are no dead states in the abstract transition systems.

A Appendix

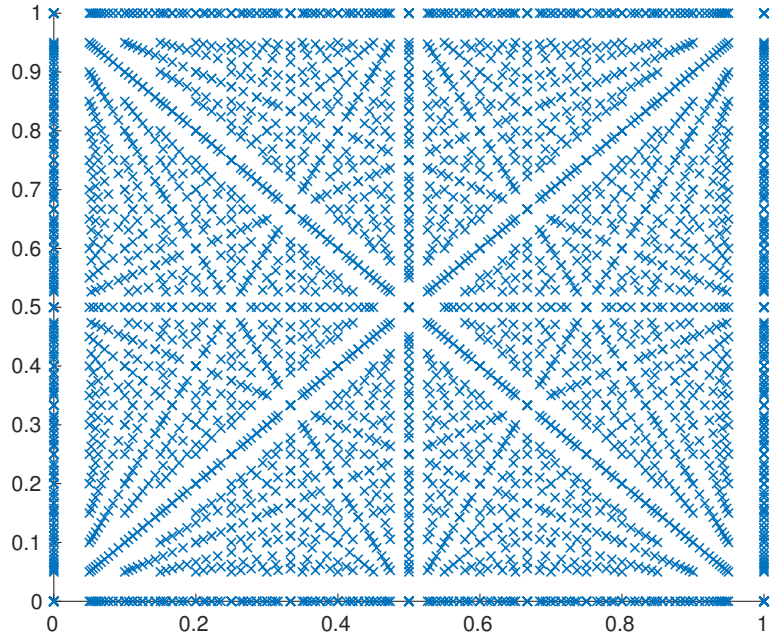

Figure 15: Plot of all points (x, y) where x and y can be expressed as $x = a/c$ and $y = b/c$ with a, b and c being integers in $\{0, \ldots, 20\}$. Note the visible diagonals and the points where they come together.

References

Blai Bonet, Menkes van den Briel, et al. Flow-Based Heuristics for Optimal Planning: Landmarks and Merges. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: what's the difference anyway? In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 162–169, 2009.

Malte Helmert, Patrik Haslum, and Jörg Hoffmann. Explicit-state abstraction: A new method for generating heuristic functions. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference*, pages 1547–1550, 2008.

Florian Pommerening and Malte Helmert. A Normal Form for Classical Planning Tasks. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 188–192, 2015.

Florian Pommerening, Gabriele Röger, Malte Helmert, and Blai Bonet. LP-Based Heuristics for Cost-Optimal Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 226–234, 2014.

Jendrik Seipp, Florian Pommerening, Silvan Sievers, and Malte Helmert. downward-lab 2.0. `https://doi.org/10.5281/zenodo.399255`, 2017. URL `https://doi.org/10.5281/zenodo.399255`.

Menkes van den Briel, J Benton, Subbarao Kambhampati, and Thomas Vossen. An LP-based heuristic for optimal planning. *Principles and Practice of Constraint Programming (CP)*, pages 651–665, 2007.

Declaration on Scientific Integrity
Erklärung zur wissenschaftlichen Redlichkeit

includes Declaration on Plagiarism and Fraud
beinhaltet Erklärung zu Plagiat und Betrug

Author — Autor
Philipp Oldenburg

Matriculation number — Matrikelnummer
13-061-064

Title of work — Titel der Arbeit
Time Unrolling Heuristics

Type of work — Typ der Arbeit
Master Thesis

Declaration — Erklärung
I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Hiermit erkläre ich, dass mir bei der Abfassung dieser Arbeit nur die darin angegebene Hilfe zuteil wurde und dass ich sie nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst habe. Ich habe sämtliche verwendeten Quellen erwähnt und gemäss anerkannten wissenschaftlichen Regeln zitiert.

Basel, 03.03.2018

P. Oldenburg

Signature — Unterschrift