University of Basel

Mutex Based Potential Heuristics

Bachelor thesis

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Artificial Intelligence
https://ai.dmi.unibas.ch

Examiner: Prof. Dr. Malte Helmert
Supervisor: Salomé Eriksson

Salome Müller
salo.mueller@unibas.ch
2017-063-058

06. 11. 2020

Table of Contents

1 Background

The goal of this chapter is to define and explain the terminology in this thesis. For visualization the 8-Tiles problem is used as an example. This is a classical planning problem, in which 8 tiles are arranged in a 3x3-Grid. One spot remains empty, the goal is to bring the tiles in a specific order by sliding them around.

1.1 Planning Tasks

In order to solve a classical planning problem with heuristic search it is represented as a **planning task**. Fišer et al. use the finite domain representation (**FDR**) where Π is specified by a tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ [2].
$\mathcal{V}$ is a finite set of **variables**, each of the variables $V \in \mathcal{V}$ has a finite set of **domains** $\mathrm{dom}(V)$. For 8-Tiles, the variables could be defined as the 9 fields in the grid (v_1 to v_9), and their domains hold the values of all tiles and the blank space (1 to 8 and 0 for the blank tile). A **fact** $f = \langle V, v \rangle$ consists of a variable $V \in \mathcal{V}$ and one of its values $v \in \mathrm{dom}(V)$. The fact for tile number 5 being in the first position would be $\langle v_1, 5 \rangle$. $\mathcal{F}_V$ is the set of all possible facts of variable $V \in \mathcal{V}$ while $\mathcal{F}$ is the set of all facts of this problem.
A **partial state** p of size t contains t facts of t different variables, i.e., it is the variable assignment over the variables $\mathrm{vars}(p) \subseteq \mathcal{V}$ with $|\mathrm{vars}(p)| = t$. $p[V]$ is the value assigned to V in p. In other words, $p = \{\langle V, p[v] \rangle | V \in \mathrm{vars}(p)\}$. A **state** s is not partial, if all variables are assigned, i.e., $\mathrm{vars}(s) = \mathcal{V}$. It **extends** the partial state $p \subseteq s$, if $s[v] = p[v]$ for all $v \in \mathrm{vars}(p)$. The partial state $p = \{v_1 \mapsto 0, v_2 \mapsto 1\}$ represents all states where the first grid in the field of the 8-Tiles puzzle is the blank space while tile number one lies in the second field.
I is the **initial state**, in 8-Tiles this is some specific random order of the tiles. G is a partial state representing the **goal**. s is a **goal state**, if it is an extension of G. In 8-Tiles it is one specific order of the tiles e.g. sorted by number: $s = \{v_1 \mapsto 1, v_2 \mapsto 2, v_3 \mapsto 3, v_4 \mapsto 4, v_5 \mapsto 5, v_6 \mapsto 6, v_7 \mapsto 7, v_8 \mapsto 8, v_9 \mapsto 0\}$.
$\mathcal{O}$ is a finite set of **operators**. Each $o \in \mathcal{O}$ has a precondition $\mathrm{pre}(o)$ and an effect $\mathrm{eff}(o)$ which are both partial states over $\mathcal{V}$, and a cost $\mathrm{c}(o) \in \mathbb{R}_0^+$.
The operator o is **applicable** in state s iff $\mathrm{pre}(o) \subseteq s$, the **resulting state** is $o[\![s]\!]$. $o[\![s]\!][v] =$

$\text{eff}(o)[v]$ holds for all $v \in \text{eff}(o)$ in resulting state $o[\![s]\!]$, while $o[\![s]\!][v] = s[v]$ for all $v \notin \text{eff}(o)$. **reformulate.** In 8-Tiles the operators encode the movement of one tile to the blank space. The precondition assures that the tile is next to the blank space, the effect swaps the values of the corresponding two variables, while all other tiles remain at the same position.
In order to reach the goal multiple operators need to be applied in a specific order. A sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ is called a path, $\pi[\![s]\!] = s_n$. π is a **s-plan**, if π is applicable in s and $\pi[\![s]\!]$ is a an extension of G and therefore a goal state. If it has minimal cost among all s-plans it is called **optimal**.
The set $\mathcal{R}$ is defined as the set of all **reachable** states. A state s is reachable, if a plan π is applicable in I such that $\pi[\![I]\!] = s$. An operator o is reachable, if it is applicable in a reachable state. A state s is a **dead-end state** if it does not extend the goal state, and no s-plan exists.

1.2 Heuristics

A **heuristic** $h : \mathcal{R} \to \mathbb{R} \cup \{\infty\}$ estimates the cost of the optimal plan for a state s. The problem of 8-Tiles has uinform cost, as sliding a tile always costs the same, i.e. 1, and there are no other operators. Therefore, the cost of a s-plan of any state which is not a dead-end equals the amount tiles, which need to be slid in the plan. The **optimal heuristic** $h^*(s)$ maps each state s to its actual optimal cost, or to ∞ if it is a dead-end state. We aim to approach this heuristic.
This thesis uses heuristics in the forward heuristic search where unreachable states are never expanded. Therefore they are defined over $\mathcal{R}$ instead of over all states and the above defined rules hold for reachable states only.
A heuristic is **admissible**, if it never overestimates the optimal heuristic, i.e., $h(s) \leq h^*(s)$. It is **goal-aware** iff $h(s) \leq 0$ for all reachable goal states, i.e., it recognizes a goal sate as such. Further, it is **consistent** iff $h(s) \leq h(o[\![s]\!]) + c(o)$. This rule assures, that the heuristic of the successor of a state is not a a lot lower than the heuristic of the state itself.
A heuristic which is goal aware and consistent is also admissible.
One class of heuristics are potential heuristics which assign a potential to each possible fact of the planning task.

Definition 1. *Let Π denote a planning task with facts $\mathcal{F}$. A* ***potential function*** *is a function $\mathtt{P} : \mathcal{F} \mapsto \mathbb{R}$. A* ***potential heuristic*** *for $\mathtt{P}$ maps each state $s \in \mathcal{R}$ to the sum of potentials of facts in s, i.e., $h^{\mathtt{P}}(s) = \sum_{f \in s} P(f)$.*

Potential heuristics are goal-aware, consistent and admissible [1]. The potentials themselves are obtained through optimization which will be further analyzed in Chapter 2.
One further approach is **ensemble heuristics**. Instead of only one heuristic, this approach uses multiple heuristics and chooses the highest value as heuristic value for each state.

1.3 Mutexes and Disambiguations

Mutex means, that two or more things mutually exclude each other.

Definition 2. *Let Π denote a planning task with facts $\mathcal{F}$. A set of facts $\mathcal{M} \subseteq \mathcal{F}$ is a* ***mutex*** *if $\mathcal{M} \nsubseteq s$ for every reachable state $s \in \mathcal{R}$*

Facts are a mutex if they never appear together in any reachable state. If a partial state p in 8-Tiles holds $p[v_3] = 1$, then tile one may not be in any other spot of the grid, i.e., the fact $\langle v_3, 1\rangle$ is mutex with all other facts $\langle v, 1\rangle$ with $v \in \mathcal{V} \setminus \{v_3\}$.

Definition 3. *Let Π denote a planning task with variables $\mathcal{V}$ and facts $\mathcal{F}$. A set of sets of facts $\mathcal{M} \subseteq 2^{\mathcal{F}}$ is called a* ***mutex-set*** *if the following hold: (a) every $M \in \mathcal{M}$ is a mutex; and (b) for every $M \in \mathcal{M}$ and every $f \in \mathcal{F}$ it holds that $M \cup \{f\} \in \mathcal{M}$; and (c) for every variable $V \in \mathcal{V}$ and every pair of facts $f, f' \in \mathcal{F}_V$, $f \neq f'$, it holds that $\{f, f'\} \in \mathcal{M}$.*

We can say that $s \in \mathcal{M}$ if s contains a subset of facts which are a mutex.
Mutexes can be used to derive disambiguations.

Definition 4. *Let Π denote a planning task with facts $\mathcal{F}$ and variables $\mathcal{V}$, let $V \in \mathcal{V}$ denote a variable, and let p denote a partial state. A set of facts $F \subseteq \mathcal{F}_V$ is called a* ***disambiguation*** *of V for p if for every reachable state $s \in \mathcal{R}$ such that $p \subseteq s$ it holds that $F \cap s \neq \emptyset$ (i.e., $\langle V, s[V]\rangle \in F$).*

The disambiguation of a variable V for a partial state p is the set of facts $F \in \mathcal{F}_V$ which occur in all reachable extended states of p. This means, that each fact of V which is not in F is a mutex with p. If F contains exactly one fact then p can be safely extended with that fact, as it is the only non-dead-end extension of the state. If F is the empty set every extended state of p is a dead-end. This knowledge can be used to prune operators o for which $p \subseteq \text{pre}(o)$ and unreachable states $s \subseteq p$. If the goal state G is one of this states, the problem is unsolvable.
If a partial state s of the 8-Tiles problem holds $p[v_3] = 1$ and $p[v_2] = 1$, then it is a dead-end, as these facts are a mutex. If $p = \{v_1 \mapsto 1, v_2 \mapsto 2, v_3 \mapsto 3, v_4 \mapsto 4, v_5 \mapsto 5, v_6 \mapsto 6, v_7 \mapsto 7, v_8 \mapsto 8\}$ then p is not a dead-end and v_9 can safely be assigned with 0, as it is the only fact in $\mathcal{F}_{v_9}$ which does not form a mutex with any of the already assigned facts.
The set $\mathcal{M}_p = \{f | f \in \mathcal{F}, p \cup \{f\} \in \mathcal{M}\}$ is the set of facts which are mutex with p. All facts of a variable $f \in \mathcal{F}_V$ not contained in $\mathcal{M}_p$ build the disambiguation F of V for p. In 2 we will use this to improve potential heuristics by narrowing down possible extensions of partial states.

2 Strengthening Potential Heuristics

Fišer et al. propose a method to improve potential heuristics with mutexes and disambiguations. This chapter contains the changes which are required to do so, regarding the transformation of a planning task into TNF and the adaption of the optimization functions. It shows how the equations which were later implemented (Section Implementation) are derived.

2.1 Potential Heuristics

When Pommerening et al. first introduced potential heuristics, they showed that two inequalities are sufficient to proof admissibility.

Theorem 5. *Let* $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ *denote a planning task,* $\mathtt{P}$ *a potential function, and for every operator* $o \in \mathcal{O}$*, let* $\mathrm{pre}^*(o) = \{\langle V, \mathrm{pre}(o)[V] \rangle | V \in \mathrm{vars}(\mathrm{pre}(o)) \cap \mathrm{vars}(\mathrm{eff}(o))\}$ *and* $\mathrm{vars}^*(o) = \mathrm{vars}(\mathrm{eff}(o)) \setminus \mathrm{vars}(\mathrm{pre}(o))$*. If*

$$\sum_{f \in G} \mathtt{P}(f) + \sum_{V \in \mathcal{V} \setminus \mathrm{vars}(G)} \max_{f \in \mathcal{F}_V} \mathtt{P}(f) \leq 0 \tag{2.1}$$

and for every operator $o \in \mathcal{O}$ *it holds that*

$$sum_{f \in \mathrm{pre}^*(o)} \mathtt{P}(f) + \sum_{V \in \mathrm{vars}^*(o)} \max_{f \in \mathcal{F}_V} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f) \leq c(o) \tag{2.2}$$

then the potential heuristic for $\mathtt{P}$ *is admissible.*

Eq. (2.1) of the Theorem 5 assures goal-awareness of the potential heuristic. As all variables are assigned in the goal state, the potential of one fact per variable has to be summed up. For the variables $v \in \mathrm{vars}(G)$ we can simply use the potentials of their respective facts. Meanwhile we assume the worst case for the other variables, by using the maximal potential over their facts, as we do not know what fact they are assigned.
Eq. (2.2) assures consistency. Recall the general heuristics consistency equation $h(s) \leq h(o[\![s]\!]) + \mathrm{c}(o)$. It can be rewritten as $h(s) - h(o[\![s]\!]) \leq \mathrm{c}(o)$. As the facts which do not occur in the effect are the same in both s and $o[\![s]\!]$ we can leave them aside. For s we know what facts of the variables of the preconditions are assigned and sum the potentials of the

facts which are in the effect as well. For the variables which are in the effect but not in the precondition we proceed similarly to (2.1), as we do not know their values. The potentials of the facts in the effect can be used without modification for $o[\![s]\!]$.

The advantage of these equations is that they are not state-dependent, even though they do not tell us explicitly what the potentials should be. However, they can be used as the constraints for a linear program, the solution of which is a potential function that forms an admissible potential heuristic. More about this in Section 2.2.

2.1.1 Generalization with Mutexes

Mutexes can be used to reduce the domain of variables, which are not yet assigned in a partial state p. This property is very helpful, as it minimizes the amount of facts which are candidates for the not assigned variables in Equations (2.1) and (2.2) of Theorem 5.

make this algorithm look a little nicer. . .

Algorithm 1 Multi-fact fixpoint disambiguation.

```
Input: A planning task $\Pi$ with variables $\mathcal{V}$ and facts $F$, a partial state $p$, and a mutex-set $\mathcal{M}$.
Output: A set of disambiguations $\mathcal{D}_p$ of all variables $\mathcal{V}$ for $p$.
 1: $D_v \leftarrow F_V$ for every $V \in \mathcal{V}$
 2: $A \leftarrow \mathcal{M}_p$
 3: change $\leftarrow$ True
 4: while change do
 5:     change $\leftarrow$ False
 6:     for all $V \in \mathcal{V}$ do
 7:         if $D_V \setminus A \neq D_V$ then
 8:             $D_V \leftarrow D_V \setminus A$
 9:             $A \leftarrow A \cup \bigcap_{f \in D_V} \mathcal{M}_{p \cup \{f\}}$
10:             change $\leftarrow$ True
11:         end if
12:     end for
13: end while
14: $\mathcal{D}_p \leftarrow \{D_V | V \in \mathcal{V}\}$
```

At the beginning, the set D_V contains all possible values for the variable $V \in \mathcal{V}$, while A contains all facts which are a mutex with any fact in p. In each iteration of the while-loop, all $f = \langle v, V \rangle$ which are in A and in D_V are removed from the corresponding D_V. On line 9, A is extended with all facts that form a mutex with all facts remaining in D_V, i.e., which are a mutex with $p \cup \{f\}$ for all $f \in D_V$.

In conclusion, after applying mutli-fact fixpoint disambiguation p can be extended with any fact in $\mathcal{D}_p$ without reaching a dead-end state. If any $D_V \in \mathcal{D}_p$ is empty, then p is already a dead-end itself.

This algorithm is used for several applications during the remainder of this chapter. In this section, it can be used to generalize Theorem 5 by the following theorem.

Theorem 6. *Let* $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ *denote a planning task with facts* $\mathcal{F}$*, and let* $\mathtt{P}$ *denote a potential function, and* *(i) for every variable $V \in \mathcal{V}$, let $G_V \subseteq \mathcal{F}_V$ denote a disambiguation of V for G s.t. $|G_V| \geq 1$, and*

(ii) for every operator $o \in \mathcal{O}$ and every variable $V \in \mathrm{vars}(\mathrm{eff}(o))$, let $E^o_V \subseteq \mathcal{F}_V$ denote a disambiguation of V for $\mathrm{pre}(o)$ s.t. $|E^o_V| \geq 1$.

If

$$\sum_{V \in \mathcal{V}} \max_{f \in G_V} \mathtt{P}(f) \leq 0 \tag{2.3}$$

and for every operator $o \in \mathcal{O}$ it holds that

$$\sum_{V \in \mathrm{vars}(\mathrm{eff}(o))} \max_{f \in E^o_V} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f) \leq \mathrm{c}(o) \tag{2.4}$$

then the potential heuristic $\mathtt{P}$ *is admissible.*

Fišer et al. prove the theorem by showing that Equations (2.3) and (2.4) are generalizations of Equations (2.1) and (2.2), respectively.
The disambiguation G_V equals $D_V \in \mathcal{D}_G$ with $V \in \mathcal{V}$, which is generated by applying Algorithm 1 on the goal state. If it is empty for any of the variables, then the problem is unsolvable, as the goal contains a mutex and is therefore not a reachable state. E^o_V is equal to $D_V \in \mathcal{D}_{\mathrm{pre}(o)}$. o is not applicable in any (partial) state if E^o_V is empty for any $V \in \mathrm{vars}(\mathrm{eff}(o))$.
To show the reader why this property is useful in practice, we first introduce the Transition Normal Form.

2.2 Transition Normal Form

Planning tasks can be in Transition Normal Form (TNF). A planning task in TNF has a fully defined goal ($\mathrm{vars}(G) = \mathcal{V}$) and all variables of the effect are also in the precondition for each operator $o \in \mathcal{O}$ ($\mathrm{vars}(\mathrm{pre}(o)) = \mathrm{vars}(\mathrm{eff}(o))$). These properties are essential to form the Linear Program, which we will do in the next section. Any planing task $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ can be transformed into TNF with the following rules cited from Fišer et al.:

- Add a fresh value U to the domain of every variable.
- For every variable $V \in \mathcal{V}$ and every fact $f \in \mathcal{F}_V$, $f \neq \langle V, U \rangle$, add a new *forgetting* operator o_f with $\mathrm{pre}(o_f) = \{f\}$ and $\mathrm{eff}(o_f) = \{\langle V, U \rangle\}$ and the cost $\mathrm{c}(o_f) = 0$.
- For every operator $o \in \mathcal{O}$ and every variable $V \in \mathcal{V}$:
  - If $V \in \mathrm{vars}(\mathrm{pre}(o))$ and $V \notin \mathrm{vars}(\mathrm{eff}(o))$, then add $\langle V, \mathrm{pre}(o)[V] \rangle$ to $\mathrm{eff}(o)$.
  - If $V \in \mathrm{vars}(\mathrm{eff}(o))$ and $V \notin \mathrm{vars}(\mathrm{pre}(o))$, then add $\langle V, U \rangle$ to $\mathrm{pre}(o)$.
- For every $V \in \mathcal{V} \setminus \mathrm{vars}(G)$ add $\langle V, U \rangle$ to G.

Each Variable $V \in \mathcal{V}$ gets a new value U in its domain, which can be seen as a sort of placeholder. The fact $\langle V, U \rangle$ can be assigned with cost 0, as the forgetting operator o_f which assigns it has no cost, regardless of the current state and especially the current assignment

of V. The next point is to assure that for each operator the variables which are in the precondition are also in the effect.
If V is present in the preconditions of an operator $o \in \mathcal{O}$ but not in the effect, then we can simply add the variable and the value it is already assigned to the effect. This is a formal change, but does not change the effect of the operator at all, as it would not have changed this fact anyway.
The case of an operator $o \in \mathcal{O}$, where V is in the effect but not in the precondition, is a little more complicated. Here, the precondition is changed such that it contains also the fact $\langle V, U \rangle$. If o was applicable in s before, then, after transforming the plan into TNF, the corresponding o_f needs to be applied beforehand in order to forget the value of V. This change of the variable is insignificant, as the value then gets changed by applying the operator anyways.
Last, all variables which were not included in the partial state G need to be added into it. If they are assigned the fresh value U, then the goal state can be reached from every state which expanded it before. Without creating more cost, the values of all unimportant variables are changed to the fresh value. The compilation into TNF can produce a plan twice the size of the original task in worst case [4].

2.2.1 Generalization with Mutexes

Similar to Section 2.1.1, these rules can be generalized with disambiguations. Therefore, to replace U we introduce the fresh values U_{G_V} and $U_{E_V^o}$. Instead of adding forgetting operators from every fact in every $V \in \mathcal{V}$, we use the disambiguation sets G_V and E_V^o.

- Add fresh value U_{G_V} to the domain of every $V \in \mathcal{V}$.
- For every variable $V \in \mathcal{V}$ and every fact $f \in G_V$, $f \neq \langle V, U_{G_V} \rangle$, add new *forgetting* operators o_{f_G} with $\mathrm{pre}(o_{f_G}) = \{f\}$ and $\mathrm{eff}(o_{f_G}) = \{\langle V, U_{G_V} \rangle\}$ and the cost $\mathrm{c}(o_{f_G}) = 0$.
- For every $V \in \mathcal{V} \setminus \mathrm{vars}(G)$ add $\langle V, U_{G_V} \rangle$ to G.
- For every operator $o \in \mathcal{O}$ add fresh value $U_{E_V^o}$ to the domain of every $V \in \mathcal{V}$:
  - If $V \in \mathrm{vars}(\mathrm{pre}(o))$ and $V \notin \mathrm{vars}(\mathrm{eff}(o))$, then add $\langle V, \mathrm{pre}(o)[V] \rangle$ to $\mathrm{eff}(o)$.
  - If $V \in \mathrm{vars}(\mathrm{eff}(o))$ and $V \notin \mathrm{vars}(\mathrm{pre}(o))$, then add $\langle V, U_{E_V^o} \rangle$ to $\mathrm{pre}(o)$.
- For every variable $V \in \mathcal{V}$, every operator$o \in \mathcal{O}$ and every fact $f \in E_V^o$, $f \neq \langle V, U_{E_V^o} \rangle$, add new forgetting operators $o_{f,o}$ with $\mathrm{pre}(o_{f,o}) = \{f\}$ and $\mathrm{eff}(o_{f,o}) = \{\langle V, U_{E_V^o} \rangle\}$ and the cost $\mathrm{c}(o_{f,o}) = 0$.

For the goal state, forgetting operators are only created for the facts in F_V which are not a mutex with any $f \in G$ for every $V \notin \mathrm{vars}(G)$. Similarly, facts in F_V which are a mutex with any $f \in \mathrm{pre}(o)$ are not taken into account for all $o \in \mathcal{O}$ and every $V \in \mathrm{vars}(\mathrm{eff}(o))$. This creates at most $|\mathcal{O}| * |\mathcal{V}|$ $U_{E_V^o}$ and $|\mathcal{V}|$ U_{G_V} and the amount of forgetting operators is in the worst case the same, multiplied with the sum of the cardinalities of the domains of all $V \in \mathcal{V}$. In practice, Fišer et al. show that the transformation with disambiguations is never bigger than without disambiguations.

2.3 Linear Program

The formulas and rules which were defined in the previous two sections can now be used to form a Linear Program (LP).
An LP consists of LP-variables which are constrained by multiple inequalities (constraints) and which are part of an optimization function. An LP-solver then assigns each LP-variable a value, such that all constraints are satisfied and the optimization function is optimized. We will look at different optimization functions in the next section (2.4).

Definition 7. *Let f be a solution to the following LP:*
Maximize opt *subject to* $\sum_{V \in \mathcal{V}} \mathtt{P}_{\langle V, s[V] \rangle} \leq 0$ *and* $\sum_{V \in \text{vars}(\text{eff}(o))} (\mathtt{P}_{\langle V, \text{pre}(o)[V] \rangle} - \mathtt{P}_{\langle V, \text{eff}(o)[V] \rangle}) \leq \text{c}(o)$ *for all $o \in \mathcal{O}$, where the objective function* opt *can be chosen arbitrarily.*
Then the function $\text{pot}_{\text{opt}}(\langle V, v \rangle) = f(\mathtt{P}_{\langle V, v \rangle})$ *is the* potential function optimized for opt *and* $h^{\mathtt{P}}$ *is the* potential heuristic optimized for opt*.*

In order to find a potential heuristic, the LP-variables are the potentials of the facts, $P(f)$. Further we define the constraints as Equation (2.3) for the goal state and Equations (2.4) for every operator. Since these two formulas assure admissibility, any solution of the LP builds an admissible $h^{\mathtt{P}}$. We introduce the the LP-variables $X_f = \mathtt{P}(f)$ for every $f \in \mathcal{F}$ and M_{G_V} and $M_{E_V^o}$ corresponding to U_{G_V} and $U_{E_V^o}$, respectively, with the constraint that $X_f \leq M_{G_V}$ for every $f \in G_V$ and $X_f \leq M_{E_V^o}$ for every $f \in E_V^o$. This gives the constraints

$$\sum_{f \in G} X_f + \sum_{V \in \mathcal{V} \setminus \text{vars}(G)} M_{G_V} \leq 0 \tag{2.5}$$

and

$$\sum_{f \in \text{pre}^*(o)} X_f + \sum_{V \in \text{vars}^*(o)} M_{E_V^o} - \sum_{f \in \text{eff}(o)} X_f \leq c(o) \tag{2.6}$$

which are coherent to the formulas in Definition 7 (Pommerening et al.). M_{G_V} and $M_{E_V^o}$ correspond to U_{G_V} and $U_{E_V^o}$, respectively, since these are the facts which are assigned if a variable is not defined in the goal state or in a precondition of an operator.
The solution of the LP might differ vastly depending on the used optimization function. We will look at multiple different possibilities for optimization functions.

2.4 Optimization Functions

An optimization function opt is a linear combination of the LP-variables. In our case, the goal is to have best possible heuristic value for as many states as possible. Using different optimization functions optimizes different aspects of a heuristic. The perfect heuristic would be achieved, if we optimized the potentials for each single state, but this is computationally too expensive.
In the first proposal for potential heuristics from Pommerening et al., the optimization for the initial state was used,

$$\text{opt}_I = \sum_{f \in I} \mathtt{P}(f). \tag{2.7}$$

It optimizes the heuristic value for the initial state. The drawback of is that facts which do not appear in the initial state are not taken into account.

Alternatively, we could optimize the potentials for all reachable states, with the all-states-potentials optimization function:

$$\mathrm{opt}_{\mathcal{R}} = \frac{1}{|\mathcal{R}|} \sum_{s \in \mathcal{R}} \sum_{f \in s} \mathtt{P}(f). \tag{2.8}$$

It calculates the weighted sum of all facts, i.e., it multiplies the potential of a fact with the amount of reachable states containing this fact and normalizes it with the total amount of reachable states. The potentials generated with this optimization function would result in the heuristic with the maximal average heuristic value over all reachable states. Unfortunately, calculating this is, again, computationally expensive or even infeasible, if the planning task and therefore the size of $\mathcal{R}$ are big, as the set of reachable states is not known. To avoid this, we could sample some states $S \subseteq \mathcal{R}$, and calculate Equation (2.8) over these states, instead of $\mathcal{R}$:

$$\mathrm{opt}_{\hat{\mathcal{S}}} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \sum_{f \in s} \mathtt{P}(f). \tag{2.9}$$

The optimization function could also assume uniform distribution and give all facts the same weight. Instead of going over all facts of all states, we would sum over the domains of all variables:

$$\mathrm{opt}_{\mathcal{S}} = \sum_{\langle V,v \rangle \in \mathcal{F}} \frac{1}{|\mathrm{dom}(V)} \mathtt{P}(\langle V, v \rangle). \tag{2.10}$$

These two approaches for the all-states-potential optimization function can be strengthened with mutexes and disambiguations, which we will describe in the next sections.

2.4.1 Strengthening All State Potentials

To estimate the amount of reachable states containing $f = \langle V, v \rangle$, we will calculate the upper bound of these states, and try to lower it for each $f \in \mathcal{F}$. The product of the domains of all variables except V, since V is already assigned, is the total amount of states, reachable and non-reachable, which contain f. With Algorithm 1 from Section 2.1.1 we can remove all facts from all domains which are mutex with f. These facts could never be assigned in any reachable state containing f. Therefore, the product over all domains in the resulting disambiguation set is again an upper bound of appearances of f.
This holds not only for a single fact, but can be applied to estimate the appearances of any partial state p. As the product of all domains in the disambiguation set is taken, the value is zero, if p is a mutex itself.
If we define $\mathcal{M}$ as the superset of all mutexes and $\mathcal{M}_p$ as the set of all facts which form a mutex with patial state p, then an upper bound for the amount of reachable states of size k containing f is

$$\mathcal{C}_f^k(\mathcal{M}) = \sum_{p \in \mathcal{P}_k^{\{f\}}} \prod_{V \in \mathcal{V}} |F_V \setminus \mathcal{M}_p|. \tag{2.11}$$

The Equation has the constraint to only extend states to size k, as it would get computationally too expensive to compute all reachable states for which f holds.

We will now use Equation (2.11) to calculate the weights for the potential of each $f \in \mathcal{F}$ and form the following optimization function:

$$\text{opt}^k_{\mathcal{M}} = \sum_{f=\langle V,v\rangle \in \mathcal{F}} \frac{\mathcal{C}^k_f(\mathcal{M})}{\sum_{f' \in \mathcal{F}_V} \mathcal{C}^k_{f'}(\mathcal{M})} \mathtt{P}(f). \tag{2.12}$$

The result is a modification of $\text{opt}_{\hat{\mathcal{S}}}$ (Eq. (2.9)) that des not assume uniformly distributed facts. Each fact is weighted according to its estimated appearance in all reachable states of size k in relation to the other facts of the corresponding variable. The sum of the weights for all facts of one variable is 1. We can use this not only as a single heuristic, but also in ensemble heuristics as we will show in the next section.

2.4.2 Strengthening Conditioned Ensemble Potentials

If we divide $\mathcal{R}$ into multiple subsets $S_i \subset \mathcal{R}$, with $i = 1, \ldots, n$ and $S_1 \cup \cdots \cup S_n = \mathcal{R}$, the solution of the LP with optimization function $\text{opt}_{\hat{S}_i}$ gives better heuristic values for the states in S_i than $\text{opt}_{\mathcal{R}}$. If we calculate the potentials for the optimization of all S_i, the resulting heuristics can be used as ensemble heuristics. This gives a result which lies somewhere between the all-states-potential heuristic and calculating the potentials for each individual state.

We will choose S_i as the states extending one particular partial state t, and use the potentials generated with an adjusted $\text{opt}^k_{\mathcal{M}}$ as one of multiple ensemble heuristics. For this we adapt Equation (2.11), such that it counts how many reachable states of size $|t| + k$ extend t:

$$\mathcal{K}^k_f(\mathcal{M}, t) = \sum_{p \in \mathcal{P}^{t \cup \{f\}}_{|t|+k}} \prod_{V \in \mathcal{V}} |\mathcal{F} \setminus \mathcal{M}_p|. \tag{2.13}$$

The corresponding optimization function uses the weights calculated with $\mathcal{K}^k_f(\mathcal{M}, t)$,

$$\text{opt}^{t,k}_{\mathcal{M}} = \sum_{f=\langle V,v\rangle \in \mathcal{F}} \frac{\mathcal{K}^k_f(\mathcal{M}, t)}{\sum_{f' \in \mathcal{F}_V} \mathcal{K}^k_f(\mathcal{M}, t)} \mathtt{P}(f). \tag{2.14}$$

For this thesis, the partial states t were sampled uniformly at random, as did Fišer et al.. Future research could be to investigate other ways to choose them.

2.4.3 Adding Constraint on Initial State

Both $\text{opt}^k_{\mathcal{M}}$ and $\text{opt}^{t,k}_{\mathcal{M}}$ optimize the potentials in regard of the entire reachable state space. However, the importance of states may vary strongly, depending on where we start and what path is taken from there towards the goal. For example, a planning task with multiple goal states may have smaller heuristic values, even though the constraints enforce goal-awareness. This is why we now look at a third approach which gives the initial state more weight and adds the constraint

$$\sum_{f \in I} \mathtt{P}(f) = h^{\mathtt{P}}_I(I) \tag{2.15}$$

to the Linear Program, where $h^{\mathtt{P}}_I$ denotes the potential heuristic optimized for the initial state (Eq. (2.7)). By calculating the potentials for this heuristic first and then taking it into account for the actual potentials, we force the solver to find potentials which guarantee high heuristic values for the initial state. This approach can be combined with all previously discussed optimization functions.

2.5 Implementation

Our ~~The~~ implementation of mutex based potential heuristics is embedded in the planning system Fast Downward and written in C++.

We first implemented Algorithm 1, which uses the Fast Downward hm-heuristic with m=2 to build a mutex table. The computation of this ~~This~~ heuristic is very slow, which is unfortunate. **i made it better (hopefully)**.

Fast Downward has a potential optimizer, which initializes and constructs an LP-solver. We implemented a new LP-constructor which builds the constraints according to ~~as~~ Equations (2.3) and (2.4). For both the non-mutex and the mutex constructors, we added the option ~~possibility~~ to use the additional constraint on the initial state.

Next, we implemented the optimization function $\mathrm{opt}^k_{\mathcal{M}}$. All partial states of cardinality one ($p_f = \{f\}$ for all $f \in \mathcal{F}$) are generated, and then recursively all partial states of size k extending p_f are created, using the disambiguation set $\mathcal{D}_{p_f}$. Even for small tasks the amount of ~~the~~ extended states can grow very fast ~~high~~. For memory efficiency ~~issues~~, partial states are implemented as maps, containing the assigned facts only. Using these states, we calculate the weight of each fact with $\mathcal{C}^k_f(\mathcal{M})$. The weights are stored as vectors and passed to the potential optimizer, which uses them to generate the optimization function.

The optimization function $\mathrm{opt}^{t,k}_{\mathcal{M}}$ was implemented implicitly. As all former mentioned methods ~~{things}~~ can handle partial states of a cardinality ies ≥ 1 ~~higher than one,~~ only one new method was needed. It generates n mutex based potential heuristics, each of which uses a random state of size t. **same procedure as before** with each of these we perform the same procedure as above with P_f.

Running Fast Downward, the mutex based potential heuristic can be used with the command `--search "astar(mutex_based_potential())"`. The size to which p_f should be extended can be set with k, the default value is k=2.

The command `--search "astar(mutex_based_ensemble_potential())"` uses the mutex based ensemble heuristics. The variables ~~which~~ can be set manually and their respective default values are * t=1, k=2 and n=50. k must be greater ~~bigger~~ than t, but smaller than the size of a fully extended state. n can be chosen arbitrarily, however, the ~~following~~ evaluation in chapter 3 shows that the results are best with **n=50**.

Further, all potential heuristics can be solved using LP-constraints built with mutexes through the option * mutex = 1. The default value mutex= is 0 for all potential heuristics, except for the mutex based ones. The constraint for the initial state is added through init-const=1, the default value for this is 0 for all potential heuristics.

* spacing

3 Experimantal Evaluation

Some things that I tested my code with.

Bibliography

[1] Daniel Fišer, Rostislav Horčík, and Antonín Komenda. Strengthening potential heuristics with mutexes and disambiguations. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 124–133, 2020.

[2] Malte Helmert. Concise finite-domain representations for pddl planning tasks. *Artificial Intelligence*, 173(5-6):503–535, 2009.

[3] Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. From non-negative to general operator cost partitioning. 2015.

[4] Jendrik Seipp, Florian Pommerening, and Malte Helmert. New optimization functions for potential heuristics. 2015.

Declaration on Scientific Integrity
Erklärung zur wissenschaftlichen Redlichkeit

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

Author — Autor
Salome Müller

Matriculation number — Matrikelnummer
2017-063-058

Title of work — Titel der Arbeit
Mutex Based Potential Heuristics

Type of work — Typ der Arbeit
Bachelor 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, 06. 11. 2020

Signature — Unterschrift