Generalized Potential Heuristics for Classical Planning: Additional Material

Guillem Francès, **Augusto B. Corrêa**, **Cedric Geissmann** and **Florian Pommerening**
University of Basel, Basel, Switzerland
{guillem.frances,augusto.blaascorrea,cedric.geissmann,florian.pommerening}@unibas.ch
Technical Report CS-2019-003

This technical report contains proofs and a detailed version of the mixed integer program of the paper *Generalized Potential Heuristics for Classical Planning* [Francès *et al.*, 2019].

We start by proving that the functions we define in in Section 4 of the paper are indeed descending and dead-end avoiding. To show that a function is descending, we have to show that each alive state s has a successor s' with $h(s')+1 \leq h(s)$. We do this in all cases below by a case distinction over the structure of the state s. To show that a function is dead-end avoiding, we have to show that all possible transitions from an alive state s to an unsolvable state s' satisfy $h(s') \geq h(s)$. The domain Spanner is the only of our example domains that has reachable unsolvable states. In the domain VisitAll, visiting a place never hurts and in all domains other than Spanner and VisitAll the actions are invertible so the reachable part of the state space is a strongly connected component.

1 Spanner

The heuristic function we consider for the domain Spanner is

$$|S_1| + |S_2| + \mathit{dist}(S_3, link, S_4) + 2|S_5|$$

with the following interpretation of the features:

- $|S_1|$: number of spanners which have not been picked up
- $|S_2|$: number of untightened nuts
- $\mathit{dist}(S_3, link, S_4)$: distance between the location of the agent and the location of the gate
- $|S_5|$: number of spanners in a cell no longer reachable by any agent

Since the domain Spanner has reachable unsolvable states, we must prove that it is descending in alive states and dead-end avoiding. We first show that the heuristic is descending for every alive state in a case-by-case analysis. We consider the version of the domain where there is a unique gate and distinguish the following cases for an alive state s.

1. The agent is at location l which is not the gate location:
   (a) There is a spanner at l:
   In this case, *picking up the spanner* will always reduce the heuristic value. $|S_1|$ will decrease by 1, while all other features remain constant. Thus, the heuristic value changes by -1.
   (b) There is no spanner at l:
   Moving towards the gate reduces the heuristic estimate. Since there is no spanners at l, $|S_1|$ and $|S_5|$ do not change. Similarly, we do not tighten any nut and hence $|S_2|$ remains constant. Since we can only move in one direction in this domain (which is towards the gate), the distance feature will decrease by 1. Hence the heuristic value changes by -1.
2. The agent is at the gate:
   (a) There is an untightened nut at the gate:
   Tightening a nut leads to a descending transition. Since the agent is already at the gate and the domain forbids spanners at the gate, the distance feature and $|S_1|$ remain unchanged. $|S_5|$ also stays constant if we tighten a nut. The feature $|S_2|$ decreases by 1 since we have one more nut tightened after this action. The total heuristic change is -1.
   (This state might also be unsolvable, see the case w.r.t. dead-end avoidance below.)
   (b) There is no untightened nut at the gate:
   Then s is a goal state and thus not alive.

Now we show that the function is also dead-end avoiding. We have to consider all transitions from alive to unsolvable states. These are all movements away from a location that still has a spanner. Not all of them lead to an unsolvable state, e.g., if there are more spanners available than nuts to tighten. However, proving that none of these transitions is descending is sufficient to prove dead-end avoidance.

Consider a move from a location l that still contains spanners. The number of spanners which have not been picked up and the number of untightened nuts does not change and hence $|S_1|$ and $|S_2|$ remain constant. The agent gets one step closer to the gate and thus the distance feature value decreases by 1. However, $|S_5|$ increases by at least 1, because the agent cannot reach l after the move to pick up the spanner that it left there. Since the weight for $|S_5|$ is 2, the total heuristic change for this transition would be at least $-1 + 2 > 0$, which is always non-descending.

2 Gripper

The heuristic function we consider for the domain Gripper is

$$8|G_1| + 4|G_2| - 2|G_3| - 1|G_4|$$

with the following interpretation of features (assuming the target room is X)

- $|G_1|$: the number of balls in a room other than X
- $|G_2|$: the number of carried balls
- $|G_3|$: the number of balls carried by robots in room X
- $|G_4|$: the number of robots in non-empty rooms other than X

As mentioned in the paper, we consider the generalization of the domain with an arbitrary number of robots, rooms and grippers. Gripper does not have reachable unsolvable states, so we just have to prove that every state s has a descending successor. We distinguish the following cases for state s.

1. There is a robot carrying a ball in room X:
   dropping the ball reduces $|G_2|$ and $|G_3|$ by 1 while leaving $|G_1|$ and $|G_4|$ constant. The heuristic value thus changes by
   $-4 + 2 = -2$.
2. There is a robot carrying a ball in a room $A \neq X$:
   moving the robot from A to X does not affect $|G_1|$ or $|G_2|$ but will increase $|G_3|$ by the number of carried balls $b > 0$ and might decrease $|G_4|$ by 1 if there are no balls in A. The heuristic value changes by
   $-2b + 1[\text{if } A \text{ is empty}] < 0$.
3. All robots carry nothing and there is a ball in a room $A \neq X$:
   (a) There is a robot in a room A:
   picking up a ball in room A increases $|G_1|$ by 1 and increases $|G_2|$ by 1 and might decrease $|G_4|$ by 1 if this was the last ball in A. The heuristic value changes by
   $-8 + 4 + 1[\text{if this was the last ball in } A] < 0$
   (b) There is a robot in a room X:
   moving the robot from X to A does not affect $|G_1|$, $|G_2|$, or $|G_3|$ but increases $|G_4|$ by 1. The heuristic value thus changes by -1.
   (c) There is a robot in a room $B \notin \{A, X\}$:
   We can assume that B is empty (otherwise, case 3a defines a descending action). When *moving from B to A* the features $|G_1|$, $|G_2|$, and $|G_3|$ remain constant and $|G_4|$ increases by 1. The heuristic value thus changes by -1.
4. All robots carry nothing and there are no balls in rooms $A \neq X$: In this case all balls must be in room X and the state s is a goal state (and thus not alive).

3 Blocksworld

The heuristic function we consider for the domain Blocksworld is

$$-4|B_6| - |\textit{holding}| - 2|\textit{ontable}| - 2|B_7|$$

with the following interpretation of the features:

- $|B_6|$: number of well-placed blocks
- $|\textit{holding}|$: number of blocks currently held
- $|\textit{ontable}|$: number of blocks currently on the table
- $|B_7|$: number of blocks currently held while their target block is well-placed and clear

Blocksworld does not have reachable unsolvable states, so we just have to prove that every state s has a descending successor. We distinguish the following cases for state s.

1. We are currently holding a block b:
   (a) Block b has no target (is not mentioned in the goal):
   blocks that are not mentioned in the goal are well-placed by definition even if they are held, so when *putting b on the table* $|B_6|$ does not change. However, $|\textit{holding}|$ is reduced by 1 and $|\textit{ontable}|$ is increased by 1. Since b has no target, $|B_7|$ also does not change. The heuristic value changes by:
   $+1 - 2 < 0$.
   (b) The target of block b is the table:
   block b cannot be well-placed while holding it and is well-placed on the table, so *putting b on the table* increases $|B_6|$ and $|\textit{ontable}|$ while it decreases $|\textit{holding}|$. Since b should not be *on* a block in the goal, $|B_7|$ stays constant. The heuristic value changes by: $-4 + 1 - 2 < 0$.
   (c) The target of block b is block c which is misplaced or not clear:
   putting b on the table increases $|\textit{ontable}|$ by 1 and decreases $|\textit{holding}|$ by 1. $|B_6|$ remains constant because b is misplaced before and after the action and $|B_7|$ remains constant because c is misplaced or not clear. The heuristic value changes by: $+1 - 2 < 0$.
   (d) The target of block b is block c which is well-placed and clear:
   stacking b on c increases $|B_6|$ by 1 but decreases $|\textit{holding}|$ and $|B_7|$ by 1. The number of blocks on the table does not change. The heuristic value changes by: $-4 + 1 + 2 < 0$.
2. We are not holding a block but there is a misplaced block b (since blocks above misplaced blocks are also misplaced, we can assume that b is on top of a stack):
   (a) Block b is clear and on another block:
   unstacking b increases $|\textit{holding}|$ by 1 and does not affect $|\textit{ontable}|$. It might also increase $|B_7|$ if b target is clear and well-placed. If b is not mentioned in the goal then it counts as misplaced before but counts as well-placed while it is held, so $|B_6|$ can increase as well. The heuristic value changes by: $-4[\text{if } b \text{ has no target}] - 1 - 2[\text{if target of } b \text{ is clear and well-placed}] < 0$
   (b) Block b is clear and on the table but its target is well-placed and clear:
   picking up block b increases $|\textit{holding}|$ and $|B_7|$ by 1 and decreases $|\textit{ontable}|$ by 1. Since b is misplaced before and after the action, $|B_6|$ does not change. The heuristic changes by: $-1 + 2 - 2 < 0$.
   (c) Block b is clear and on the table but its target is misplaced or not clear:
   this cannot be the case for all misplaced blocks,

otherwise all misplaced blocks would be clear and on the table and the target of at least one block would be available. Thus there is at least one block that falls in one of the previous two cases, showing the existence of a descending action.

3. We are not holding a block and there is no misplaced block:
   In this case all blocks are well-placed and the state s is a goal state (and thus not alive).

4 VisitAll

The heuristic function considered for the domain VisitAll is

$$k|V_1| + \mathit{dist}(\mathit{at\text{-}robot}, \mathit{connected}, V_1)$$

with the following interpretation of features:

- $|V_1|$: places not yet visited.
- $\mathit{dist}(\mathit{at\text{-}robot}, \mathit{connected}, V_1)$: distance between the location of the robot and the closest unvisited location.

VisitAll has no reachable unsolvable state, thus we just need to prove that every alive state s has a descending successor. The function will only generalize to graphs with a diameter up to k so for the proof we assume the diameter is less than or equal to k. We distinguish the following cases for state s.

1. The robot is at location l, there is at least one unvisited location but all locations connected to l are already visited:
   Let l' be one of the unvisited locations closest to l. There is at least one movement which will move the robot closer to l', decreasing the value of $\mathit{dist}(\mathit{at\text{-}robot}, \mathit{connected}, V_1)$ by 1. The number of visited places does not change because all neighbors of l were already visited and hence $|V_1|$ remains constant. The heuristic value change is: $0 - 1 < 0$.
2. The robot is at location l and there is at least one unvisited location l' which is connected to l:
   Moving the robot from l to l' turns l' visited and hence decreases $|V_1|$ by 1. Since l' was previously unvisited and connected to l, the feature $\mathit{dist}(\mathit{at\text{-}robot}, \mathit{connected}, V_1)$ has value 1 in s. After moving to l', the value of this feature can increase up to $k - 1$, since the graph has diameter k and the previous value of the feature was 1. If l' is the last unvisited location in the graph then, accordingly to our definition, $\mathit{dist}(\mathit{at\text{-}robot}, \mathit{connected}, V_1) = 0$ after moving the robot to l'. Hence, the minimum heuristic change possible is: $-k + k - 1 < 0$.
3. The robot is at location l and all other locations in the graph are visited:
   In this case, s is a goal state (and thus not alive).

5 Logistics

The heuristic function we consider for the domain Logistics is

$$\sum_{i=1}^{12} i|P_i|$$

with the following interpretation of features

- $|P_1|$: the number of packages in a truck at the right location for that package
- $|P_2|$: the number of packages in a truck in the right city but wrong location
- $|P_3|$: the number of packages on the ground in the right city with an available truck
- $|P_4|$: the number of packages on the ground in the right city without an available truck
- $|P_5|$: the number of packages in an airplane in the right city
- $|P_6|$: the number of packages in an airplane in the wrong city
- $|P_7|$: the number of packages on the ground in an airport of the wrong city with an available plane
- $|P_8|$: the number of packages on the ground in an airport of the wrong city without an available plane
- $|P_9|$: the number of packages in a truck at the airport of the wrong city
- $|P_{10}|$: the number of packages in a truck at a non-airport location of the wrong city
- $|P_{11}|$: the number of packages on the ground of a non-airport location of the wrong city with an available truck
- $|P_{12}|$: the number of packages on the ground of a non-airport location of the wrong city without an available truck

It is easy to see that the concepts all describe sets of packages, that no package can be in two of these sets at the same time and that packages that are in none of the sets are delivered. We can thus show that the function is descending if in every state s there is an action that only moves packages from concepts P_i to concepts P_j with $j \leq i$. We do this by a case distinction over the state s.

We first consider states where there is at least one package in one of the odd-numbered sets. In those situations a package can be loaded or unloaded which only affects the concepts where this package is before and after the action application.

1. $|P_1| > 0$:
   unloading one of the packages in P_1 moves it from the truck to the correct location, so it leaves P_1 without entering any other set.
2. $|P_3| > 0$:
   loading one of the packages in P_3 moves them from P_3 to P_2.
3. $|P_5| > 0$:
   unloading one of the packages in P_5 moves them from P_5 either to P_4 or to P_3 depending on whether there is an available truck at this airport.
4. $|P_7| > 0$:
   loading one of the packages in P_7 moves them from P_7 to P_6.

5. $|P_9| > 0$:
   unloading one of the packages in P_9 moves them from P_9 either to P_8 or to P_7 depending on whether there is an available airplane at this airport.
6. $|P_{11}| > 0$:
   loading one of the packages in P_{11} moves them from P_{11} to P_{10}.

We now assume that all odd-numbered concepts are empty and distinguish cases where one of the concepts P_2, P_6, and P_{10} is non-empty. In these situations the descending action is moving a vehicle and multiple packages can be affected.

In all cases below let *src* and *dst* be the source and destination of the move in each case. Let P_{src} be the undelivered packages on the ground at *src*, let P_{dst} be the undelivered packages on the ground at *dst*, and let P_v be the packages in the moved vehicle. All packages that are in other vehicles or at locations other than *src* and *dst* are unaffected.

When moving a truck, packages in P_{src} have to be from the concept P_8 because P_3, P_7, and P_{11} are empty by assumption, packages in P_4 and P_{12} cannot have a truck next to them, and all other packages on the ground are delivered. When moving a plane, they have to be in P_4 or P_{12} for the same reasons. We use $P_{src,4}$, $P_{src,8}$ and $P_{src,12}$ to describe the relevant parts of P_{src}.

Packages in P_{dst} have to be from one of the concepts P_4, P_8 or P_{12} for the same reasons. We use $P_{dst,4}$, $P_{dst,8}$ and $P_{dst,12}$ to describe the relevant parts of P_{dst}.

Likewise, packages in a truck can only be in the concepts P_2 or P_{10}, and packages in an airplane can only be in P_6. We define $P_{v,2}$, $P_{v,10}$, and $P_{v,6}$ to describe the relevant parts of P_v.

1. $|P_2| > 0$:
   driving a truck with a loaded package from P_2 to the target of that package is a descending action.

   First consider the packages in $P_v = P_{v,2} \cup P_{v,10}$. Some packages in $P_{v,2}$ (at least 1) move from P_2 to P_1, while the others remain in P_2. All packages in $P_{v,10}$ move from P_{10} to P_9 if *dst* is an airport and otherwise stay in P_{10}.

   The packages in $P_{src} = P_{src,8}$ are and remain in P_8.

   Finally, consider the packages in $P_{dst} = P_{dst,4} \cup P_{dst,8} \cup P_{dst,12}$. Packages from $P_{dst,4}$ move from P_4 to P_3 and packages from $P_{dst,12}$ move from P_{12} to P_{11} by moving a truck next to them. Packages in $P_{dst,8}$ remain in P_8.
2. $|P_6| > 0$:
   flying an airplane with a loaded package from P_6 to the the target city of that package is a descending action.

   First consider the packages in $P_v = P_{v,6}$. Some of them (at least 1) move from P_6 to P_5, the others remain in P_6.

   The packages in $P_{src} = P_{src,4} \cup P_{src,12}$ are unaffected by moving an airplane.

   Finally, consider the packages in $P_{dst} = P_{dst,4} \cup P_{dst,8} \cup P_{dst,12}$. Packages from $P_{dst,4}$ and $P_{dst,12}$ remain in P_4 and P_{12} respectively because no truck moved. Packages in $P_{dst,8}$ change from P_8 to P_7.
3. $|P_{10}| > 0$:
   driving a truck with a loaded package from P_{10} to the airport is a descending action.

   First consider the packages in $P_v = P_{v,2} \cup P_{v,10}$. Some packages from $P_{v,2}$ move from P_2 to P_1 while the others remain in P_2. All packages (at least 1) in $P_{v,10}$ move from P_{10} to P_9.

   There are no packages in $P_{src} = P_{src,8}$ because *src* is not an airport.

   Finally, consider the packages in $P_{dst} = P_{dst,4} \cup P_{dst,8} \cup P_{dst,12}$. Packages from $P_{dst,4}$ move from P_4 to P_3, packages in $P_{dst,8}$ remain in P_8, and $P_{dst,12} = \emptyset$ because *dst* is an airport.

We can now assume that all concepts except P_4, P_8, and P_{12} are empty and deal with the remaining cases. We use the same notation as the previous case. Let *src* and *dst* be the source and destination of the move in each case. In particular, all cases below will contain at least one package on the ground of *dst*. The descending action for these cases is to move a vehicle next to the package location. When moving a vehicle we know $P_v = P_{v,2} \cup P_6 \cup P_{v,10} = \emptyset$ by assumption.

1. $|P_4| > 0$:
   driving a truck from src to dst is a descending action.

   For packages in $P_{src} = P_{src,8}$ all packages remain in P_8.

   For packages in $P_{dst} = P_{dst,4} \cup P_{dst,8} \cup P_{dst,12}$. Packages from $P_{dst,4}$ (at least 1) move from P_4 to P_3, packages in $P_{dst,8}$ remain in P_8, and packages in $P_{dst,12}$ move from P_{12} to P_{11} if *dst* is not an airport and stay in P_{12} otherwise.
2. $|P_8| > 0$:
   flying an airplane from src to dst is a descending action.

   The packages in $P_{src} = P_{src,4} \cup P_{src,12}$ are unaffected by moving an airplane.

   Finally, consider the packages in $P_{dst} = P_{dst,4} \cup P_{dst,8} \cup P_{dst,12}$. Packages from $P_{dst,4}$ and $P_{dst,12}$ remain in P_4 and P_{12} respectively because no truck moved. Packages in $P_{dst,8}$ (at least 1) change from P_8 to P_7.
3. $|P_{12}| > 0$:
   driving a truck from src to dst is a descending action.

   For packages in $P_{src} = P_{src,8}$ all packages remain in P_8.

   For packages in $P_{dst} = P_{dst,4} \cup P_{dst,8} \cup P_{dst,12}$. Packages from $P_{dst,4}$ move from P_4 to P_3, packages in $P_{dst,8}$ remain in P_8. There are packages in $P_{dst,12}$ (at least 1), which means that *dst* is not an airport. They move from P_{12} to P_{11}.

We now discussed all cases where at least one concept is non-empty. In case all concepts are empty all packages are delivered and s is a goal state (and thus not alive).

6 MIP Model

Let $\mathcal{S}$ be a set of states and $\mathcal{F}$ be a set of features. As in the paper we define $\mathcal{S}_A$ as the subset of alive states in $\mathcal{S}$ and $\mathcal{T}$ as the set of transitions starting in an alive state. We additionally define $\mathcal{T}_D$ as the subset of $\mathcal{T}$ of transitions from an alive state to an unsolvable state. The MIP $\mathcal{M}(\mathcal{S}, \mathcal{F})$ is:

$$\min_w \sum_{f\in\mathcal{F}} [w_f \neq 0]\mathcal{K}(f) \quad \text{subject to}$$

$$\bigvee_{s'\in succ(s)} h(s') + 1 \leq h(s) \qquad \text{for } s \in \mathcal{S}_A \tag{1}$$

$$h(s') \geq h(s) \qquad \text{for } (s,s') \in \mathcal{T}_D \tag{2}$$

The heuristic value of a state is a linear combination of weights multiplied with (known) constant feature values: $h(s) = \sum_{f\in\mathcal{F}} w_f \cdot f(s)$, so constraint (2) is linear. However, the disjunctions in constraints (1) and the condition in the objective function are nonlinear.

Modern MIP solvers support *indicator constraints* that are enabled only if a given binary variable has the value 1. We use such constraints to encode the disjunction in constraint (1). For each transition $(s,s') \in \mathcal{T}$, we add the binary variable $y_{s>s'}$ that indicates that the disjunction of s is satisfied for s'. We then replace constraint (1) with the following two constraints:

$$[y_{s>s'} = 1] \rightarrow (h(s') + 1 \leq h(s)) \quad \text{for } (s,s') \in \mathcal{T} \tag{3}$$

$$\sum_{s'\in succ(s)} y_{s>s'} \geq 1 \quad \text{for } s \in \mathcal{S}_A \tag{4}$$

Constraint (3) enforces the semantics of $y_{s>s'}$ and constraint (4) ensures that at least one part of the disjunction holds.

To encode the objective function, we add two binary variables x_f^+, x_f^- and the constraints

$$M_w \cdot x_f^+ \geq w_f \tag{5}$$

$$M_w \cdot x_f^- \geq -w_f \tag{6}$$

$$-M_w \leq w_f \leq M_w \tag{7}$$

for each feature $f \in \mathcal{F}$. Since weights are bounded (7), all solutions satisfy $(x_f^+ + x_f^-) = 1$ iff they satisfy $w_f \neq 0$ and we can rewrite the objective as $\sum_{f\in\mathcal{F}} \mathcal{K}(f)(x_f^+ + x_f^-)$. Bounding the weights limits the set of potential functions that we can synthesize but since the examples we have seen in Section 4 of the main paper all used relatively low weights, we think this is an acceptable price to pay for being able to reduce the number of used features.

The resulting MIP consists of

- $|\mathcal{F}|$ continuous variables w_f for weights,
- $2|\mathcal{F}|$ binary variables x_f^+, x_f^- for the objective function,
- $|\mathcal{T}|$ binary variables $y_{s,s'}^<$ for the indicator constraints,
- $|\mathcal{S}_A|$ constraints of type (2)
- $|\mathcal{T}|$ constraints of type (3),
- $|\mathcal{S}_A|$ constraints of type (4), and
- $2|\mathcal{F}|$ constraints of type (5) and (6).

The full MIP can be simplified a bit by replacing $h(s)$ with its definition:

$$\min_{w,x} \sum_{f\in\mathcal{F}} (x_f^+ + x_f^-)\mathcal{K}(f) \quad \text{subject to}$$

$$[y_{s>s'} = 1] \rightarrow \left(\sum_{f\in\mathcal{F}} w_f \cdot (f(s) - f(s')) \geq 1\right) \quad \text{for } (s,s') \in \mathcal{T}$$

$$\sum_{s'\in succ(s)} y_{s>s'} \geq 1 \quad \text{for } s \in \mathcal{S}_A$$

$$\sum_{f\in\mathcal{F}} w_f \cdot (f(s) - f(s')) \leq 0 \quad \text{for } (s,s') \in \mathcal{T}_D$$

$$M_w \cdot x_f^+ \geq w_f \quad \text{for } f \in \mathcal{F}$$

$$M_w \cdot x_f^- \geq -w_f \quad \text{for } f \in \mathcal{F}$$

$$x_f^+, x_f^-, y_{s>s'} \in \{0,1\} \quad \text{for } (s,s') \in \mathcal{T}_D$$

$$-M_w \leq w_f \leq M_w, \qquad \text{and } f \in \mathcal{F}$$

References

[Francès *et al.*, 2019] Guillem Francès, Augusto B. Corrêa, Cedric Geissmann, and Florian Pommerening. Generalized potential heuristics for classical planning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*. AAAI Press, 2019. To appear.