University of Basel

Optimizations for the Additive Heuristic in Fast Downward

Simona Wittner <simona.wittner@stud.unibas.ch>

Department of Mathematics and Computer Science, University of Basel

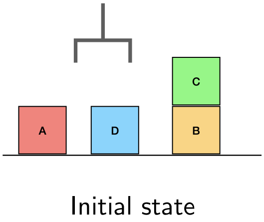
Artificial Intelligence Research Group

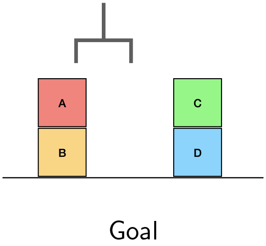
15.07.2024

Motivation

- Unary operators used to calculate values of additive heuristic in Fast Downward
- Reduce number of unary operators for more efficient calculation

Planning Task

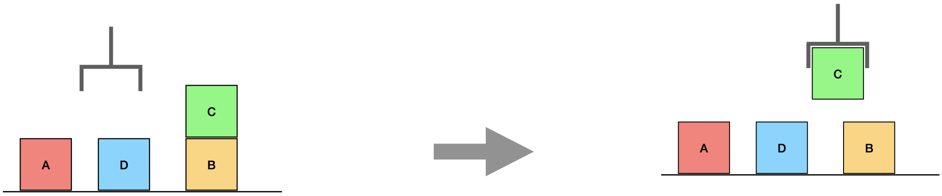




State variables: for example *on*(C, B), *ontable*(A), *handempty*(), *holding*(C), *clear*(A)

Planning Task

Actions: for example action *unstack*(C, B), with cost 1



Preconditions: *on*(C, B), *clear*(C), *handempty*()

Add effects: *holding*(C), *clear*(B)

Delete effects: *on*(C, B), *clear*(C), *handempty*()

Atom

To enable switch between ground & lifted representation of planning task

Definition (Atom)

Atom $P(\langle t_1, \dots, t_n \rangle)$ with

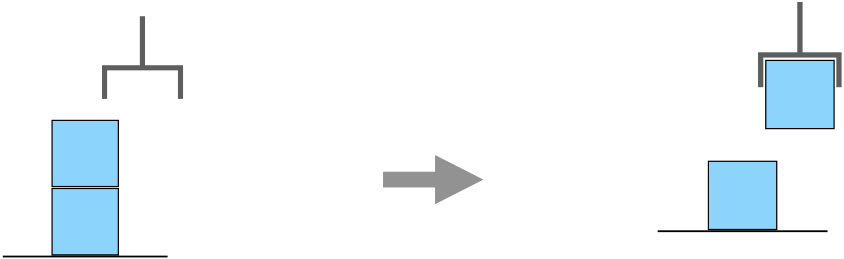
- P n-ary predicate symbol
- $\langle t_1, \dots, t_n \rangle$ tuple, where $t_1, \dots, t_n$ objects or variables

Ground atom:

- All variables replaced by objects
- Variable mapping $\sigma : \mathcal{V} \mapsto O$
- Ground atoms similar to state variables

Lifted Planning Task

Lifted actions: for example action *unstack*(x, y), with cost 1



Preconditions: *on*(x, y), *clear*(x), *handempty*()

Add effects: *holding*(x), *clear*(y)

Delete effects: *on*(x, y), *clear*(x), *handempty*()

Planning

- Find sequence of actions (plan) from initial state to a goal state of planning task
- Use search algorithms
- Search algorithms can use heuristics to enhance efficiency

Heuristics

- Guide the search
- Provide estimates of distance from states to nearest goal state
- Additive heuristic h^{add}

Weighted Datalog Program

Definition (Weighted Datalog Program)

$\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ with

- $\mathcal{F}$ facts (set of ground atoms)
- $\mathcal{R}$ weighted rules
  - Consist of atoms ϕ_i and have form $\phi_0 \leftarrow \phi_1, \ldots, \phi_m$, for $m \geq 0$
  - Weight $w(r)$ of rule $r \in \mathcal{R}$

Reachable Atoms

- Calculate reachable atoms of planning task for grounding of actions
- Use Datalog program for planning task and initial state
- Facts: ground atoms of initial state
- Rules:
  - For goal: goal-reachable ← goal atoms, weight 0
  - For each action a:
    - a-applicable ← precondition atoms, weight is $cost(a)$
    - For each effect: effect atom ← a-applicable, weight 0

Construction of Rules

- Calculate values of h^{add} using weighted Datalog program (in lifted planning)
- From paper *"Delete-relaxation heuristics for lifted classical planning"* by A. B. Corrêa, G. Francès, F. Pommerening and M. Helmert, 2021
- Special construction of rules for efficient calculation:
  1. Action Predicate Removal
  2. Rule Splitting
  3. Duplicate Rule Removal

Action Predicate Removal

- Remove all a-applicable atoms
- For each action:
  - Take rule with preconditions a-applicable $\leftarrow$ precondition atoms
  - For each rule effect atom $\leftarrow$ a-applicable set new rule:

$$\text{effect atom} \leftarrow \text{precondition atoms},$$

  where weight is cost of action

Rule Splitting

- Already in Fast Downward, here adapted for weighted rules
- Split rules in smaller rules
- New auxiliary atoms
- For each rule:
  - One "root" rule with weight of original rule
  - Other rules with weight 0

Duplicate Rule Removal

- Remove rules only different due to naming of variables
- Considers rules that define auxiliary atom
- Keep only one such rule
- Weights not affected

Optimization Idea

- Reformulate unary operators, i.e. operators with one effect
- Use weighted Datalog program with optimized rules instead of actions
- Reduce number of unary operators
- Could lead to more efficient computation of values of h^{add}

Example

Action $a[\Delta]$ with

- $pre(a[\Delta]) = \{\mathrm{P}(x), \mathrm{Q}(x, y), \mathrm{R}(z)\}$
- $add(a[\Delta]) = \{\mathrm{A}(x), \mathrm{B}(y)\}$
- $del(a[\Delta]) = \emptyset$
- $cost(a[\Delta]) = 1$
- $\Delta = \{x, y, z\}$

$O = \{o_1, o_2, o_3\}$ set of objects

Unary operator: consists of one ground atom from add list & all ground atoms from precondition list

If each variable of Δ can be mapped to each object of O:

$$|O|^{|\Delta|} \cdot |add(a[\Delta])| = 3^3 \cdot 2 = 54 \text{ unary operators after grounding}$$

Example

Rules corresponding to action $a[\Delta]$ (without rule construction approaches):

$$\mathrm{A}(x) \leftarrow a\text{-applicable} \qquad \text{weight } 0$$

$$\mathrm{B}(y) \leftarrow a\text{-applicable} \qquad \text{weight } 0$$

$$a\text{-applicable} \leftarrow \mathrm{P}(x), \mathrm{Q}(x, y), \mathrm{R}(z) \qquad \text{weight } 1$$

Example

Rules corresponding to action $a[\Delta]$ (without rule construction approaches):

$$\begin{aligned} A(x) &\leftarrow a\text{-applicable} && \text{weight } 0 \\ B(y) &\leftarrow a\text{-applicable} && \text{weight } 0 \\ a\text{-applicable} &\leftarrow P(x), Q(x, y), R(z) && \text{weight } 1 \end{aligned}$$

With action predicate removal:

$$\begin{aligned} A(x) &\leftarrow P(x), Q(x, y), R(z) && \text{weight } 1 \\ B(y) &\leftarrow P(x), Q(x, y), R(z) && \text{weight } 1 \end{aligned}$$

Example

$$A(x) \leftarrow P(x), Q(x, y), R(z) \qquad \text{weight } 1$$
$$B(y) \leftarrow P(x), Q(x, y), R(z) \qquad \text{weight } 1$$

Example

$$A(x) \leftarrow P(x), Q(x,y), R(z) \qquad \text{weight } 1$$
$$B(y) \leftarrow P(x), Q(x,y), R(z) \qquad \text{weight } 1$$

With rule splitting:

$$A(x) \leftarrow \theta_0(x), \theta_1() \qquad \text{weight } 1$$
$$B(y) \leftarrow \theta_4(y), \theta_5() \qquad \text{weight } 1$$
$$\theta_0(x) \leftarrow \theta_2(x), P(x) \qquad \text{weight } 0$$
$$\theta_2(x) \leftarrow Q(x,y) \qquad \text{weight } 0$$
$$\theta_1() \leftarrow R(z) \qquad \text{weight } 0$$
$$\theta_4(y) \leftarrow Q(x,y), P(x) \qquad \text{weight } 0$$
$$\theta_5() \leftarrow R(z) \qquad \text{weight } 0$$

No atom with θ_3: created & removed again

Example

$$A(x) \leftarrow P(x), Q(x,y), R(z) \quad \text{weight } 1$$
$$B(y) \leftarrow P(x), Q(x,y), R(z) \quad \text{weight } 1$$

With rule splitting:

$$A(x) \leftarrow \theta_0(x), \theta_1() \quad \text{weight } 1$$
$$B(y) \leftarrow \theta_4(y), \theta_5() \quad \text{weight } 1$$
$$\theta_0(x) \leftarrow \theta_2(x), P(x) \quad \text{weight } 0$$
$$\theta_2(x) \leftarrow Q(x,y) \quad \text{weight } 0$$
$$\theta_1() \leftarrow R(z) \quad \text{weight } 0$$
$$\theta_4(y) \leftarrow Q(x,y), P(x) \quad \text{weight } 0$$
$$\theta_5() \leftarrow R(z) \quad \text{weight } 0$$

No atom with θ_3: created & removed again

Example

With duplicate rule removal:

$$
\begin{array}{ll}
A(x) \leftarrow \theta_0(x), \theta_1() & \text{weight } 1 \\
B(y) \leftarrow \theta_4(y), \theta_1() & \text{weight } 1 \\
\theta_0(x) \leftarrow \theta_2(x), P(x) & \text{weight } 0 \\
\theta_2(x) \leftarrow Q(x, y) & \text{weight } 0 \\
\theta_1() \leftarrow R(z) & \text{weight } 0 \\
\theta_4(y) \leftarrow Q(x, y), P(x) & \text{weight } 0
\end{array}
$$

Four rules depend on one variable & two rules depend on two variables

If each variable can be mapped to any object of $O = \{o_1, o_2, o_3\}$:

$3^1 \cdot 4 + 3^2 \cdot 2 = 30$ unary operators after grounding (instead of 54 with actions)

Implementation (in Translate Component)

Translate component: responsible for translating planning task from PDDL representation into FDR representation, grounding of planning task

Changes:

- Build second Datalog program (weighted):
  - Use existing code from paper for construction of rules, only slightly modified
  - Remove duplicate preconditions in actions
  - Allow rules with no preconditions and only effect
- Compute reachable atoms of weighted Datalog program

Implementation (in Translate Component)

Changes (continued):

- New grounding algorithm for rules:
  - Use reachable atoms
  - For each rule: work on precondition list of rule, then on effect, build variable mappings
  - Consider atoms removed by translator:
    - Atom in initial state $\Rightarrow$ atom true in every reachable state $\Rightarrow$ ignore atom
    - Atom not in initial state $\Rightarrow$ atom false in every reachable state $\Rightarrow$ remove operator
- Write unary operators (ground rules) to new output file

Implementation (in Search Component)

Search component: responsible for finding a plan for ground planning task

Changes:

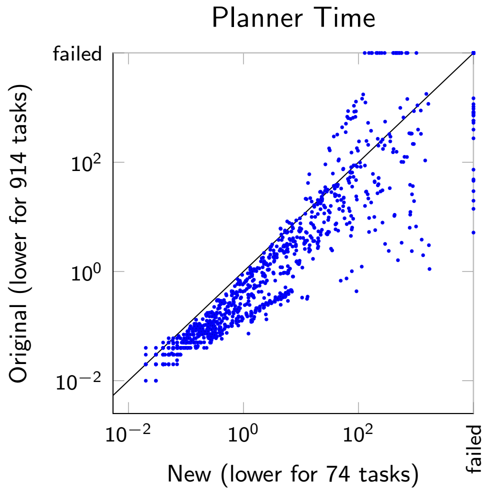
- Constructor of class `RelaxationHeuristic`:
  - Parse ground rules from new output file
  - Use new structs and vectors
  - Change `build_unary_operators` function:
    - Use parsed ground rules as unary operators
    - Remember new propositions in own vector (propositions similar to state variables)
    - Check proposition ID and set it with new function for new propositions

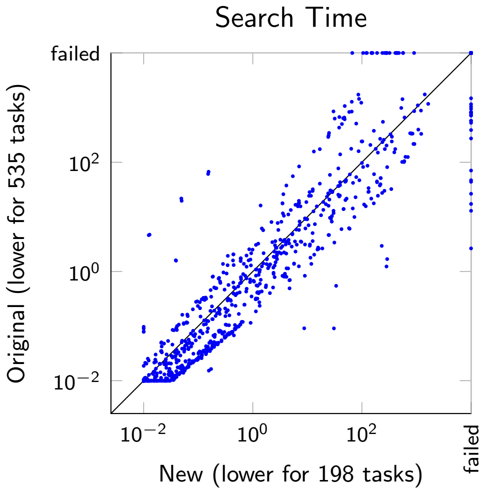
Experiments

	New	Original
Coverage – Sum	1'030	1'030
Unary operators – Sum	12'112'234	42'903'919

- Search algorithm: eager best-first search
- h^{add}
- Without preferred operators

Experiments





Experiments

Coverage per domain (where different value of coverage):

	New	Original
driverlog (20)	18	**19**
logistics98 (35)	18	**27**
parcprinter-08-strips (30)	23	**24**
parking-sat11-strips (20)	**20**	18
parking-sat14-strips (20)	**20**	5
pipesworld-notankage (50)	26	**27**
pipesworld-tankage (50)	20	**21**
rovers (40)	25	**29**
satellite (36)	**34**	30
storage (30)	16	**17**
thoughtful-sat14-strips (20)	12	**15**

Possibly beneficial domain properties: less preconditions, more effects and only few variables in actions $\Rightarrow$ many variables omitted & many duplicate rules removed

Conclusion

- Optimizations to reduce number of unary operators used to calculate values of h^{add}
  - Use weighted Datalog program with specially constructed rules
  - Algorithm for grounding the rules
  - Ground rules as unary operators
- Number of unary operators significantly reduced
- Search time generally not improved
- For specific domains: reduced search time & in some cases even planner time

Future work

Get mapping between ground rules & FDR operators used for the search

- Store transformations of Datalog program with a similar approach to annotated Datalog programs as presented in "*The FF heuristic for lifted classical planning*" by A. B. Corrêa, F. Pommerening, M. Helmert and G. Francès, 2022

$\Rightarrow$ To get preferred operators for efficient search

$\Rightarrow$ To support domains with cost depending on parameter

$\Rightarrow$ To allow domains with negative preconditions