Progressive Generalized Planner

Javier Segovia-Aguas[1], Sergio Jiménez Celorrio[2], Laura Sebastiá [2] and Anders Jonsson[1]

[1]Universitat Pompeu Fabra
[2]VRAIN - Valencian Research Institute for Artificial Intelligence, Universitat Politècnica de València
javier.segovia@upf.edu, serjice@dsic.upv.es, anders.jonsson@upf.edu, lsebastia@dsic.upv.es

Abstract

Progressive Generalized Planner (PGP) is a Best-First Search (BFS) algorithm that searches in the solution space of planning programs as Random-Access Machines, with at most $|Z|$ pointers and n program lines. The contributions of this generalized planner are twofold. On the one hand, it adapts the landmark counting heuristic to generalize planning including the novel concept of *pointer landmarks*. On the other hand, it progressively processes the set of training instances, which describe the generalized planning problem, proposing a candidate solution to them and validating it over the rest of instances. A candidate solution is considered a generalized plan iff is valid for all training instances, otherwise the first instance where it fails is included as a counter-example for the next iteration.

Generalized Planning

Generalized planning (GP) addresses the computation of algorithmic solutions that are valid for a set of classical planning instances from a given domain (Jiménez, Segovia-Aguas, and Jonsson 2019). In the worst case, each classical planning instance may require a completely different solution but in practice, many planning domains are known to have polynomial algorithmic solutions (Helmert 2006; Fern, Khardon, and Tadepalli 2011). GP is however a challenging computation task; specifying an algorithmic solution for a set of classical planning instances often requires features that are not explicitly represented in those instances and hence, they must be discovered (Bonet and Geffner 2021).

A GP problem $\mathcal{P}$ is formalized as a finite and non-empty set of T classical planning instances I_t, s.t. $1 \leq t \leq T$, from a given domain $\mathcal{D}$, i.e. $\mathcal{P} = \{P_1, \ldots, P_T\}$ where $P_1 = \langle \mathcal{D}, \mathcal{I}_1 \rangle, \ldots, P_T = \langle \mathcal{D}, \mathcal{I}_T \rangle$.

A solution to a GP problem is named a generalized plan Π, and it is a valid solution iff for every for every classical planning instance $P_t \in \mathcal{P}$, the sequential plan π that results from applying Π on P_t, is a solution to P_t.

Progressive Generalized Planning with Landmark Counting Heuristic

The *Progressive Generalized Planner* (PGP) (Segovia-Aguas et al. 2022) follows a *GP as heuristic search* approach (Segovia-Aguas, Jiménez, and Jonsson 2021), where the solution space is represented with a Random-Access Machine (RAM), that uses up to n program lines and $|Z|$ pointers. This representation to GP is denoted as $\mathcal{P}_{n,Z}$, and requires to update the shared domain with new actions from the RAM, and new state variables that stand for the pointers as introduced in Segovia-Aguas et al. (2022).

The two orthogonal contributions of this planner, compared to previous GP as heuristic search approaches, are the landmark counting heuristic to GP, and a method that progressively processes the planning instances in $\mathcal{P}_{n,Z}$.

Regarding the heuristic, we implement the back-chaining LAMA algorithm for finding fact landmarks and orderings between landmarks (Richter and Westphal 2010). Briefly we start from a set of known landmarks, and find new landmarks that hold in any plan before an already known landmark may become true. The landmark counting heuristic for a given state s and trace π is formalized as:

$$f_{LM}(s,\pi) = |(LM \setminus Reached(s,\pi)) \cup RAgain(s,\pi)|, \tag{1}$$

where LM is whole set of landmark in the given instance, $Reached(s,\pi)$ is how many landmarks have been reached with the given ordering constraints, and $RAgain(s,\pi)$ is the set of previously reached landmarks that are required again for solving the problem. Then, Eq. 1 can be used for guiding a search in the GP problem $\mathcal{P}_{n,Z}$ by defining a new evaluation function:

$$f_{LM}(\Pi, \mathcal{P}_{n,Z}) = \sum_t f_{LM}(\Pi, P_t), \tag{2}$$

for each $P_t \in \mathcal{P}_{n,Z}$, where $f_{LM}(\Pi, P_t) = f_{LM}(s,\pi)$ such that π is the sequential plan and s is the last state reached that results from executing Π on P_t, and $f_{LM}(s,\pi)$ is the landmark counting heuristic defined in Eq. 1. The evaluation function defined in Eq. 2 is enhanced with the concept of *pointer landmarks*, which adds new landmarks and ordering constraints to progress in the landmark graph (check Segovia-Aguas et al. (2022) for more details).

Regarding with processing planning instances progressively, PGP keeps a subset of the classical planning instances called the *active instances*, that initially contains only the first classical planning instance of the GP problem. PGP finds a program that solves the full set of *active instances*, it validates that program on the remaining instances of the GP problem, and augments the set of *active instances* with the first instance for which the program fails. The procedure is repeated until PGP finds a program that solves all the instances in the GP problem. PGP can be understood as a variant of *counterexample-guided search* (Seipp and Helmert 2018).

The main advantage of PGP is that solutions correspond to programs that are deterministic finite automata, which make them very suitable to solve very large planning instances. However, the submission to the *IPC 2023 - Learning Track* was just posed as a submission example and has some strong limitations like, the number of lines, that is fixed for all domains ($n = 15$), so if a solution requires more lines it will not be found; the input language does not accept certain language features, e.g., hierarchical typing or the use of constants in the action schemas; assumes that if goal information is relevant to generalize, this will be provided in the initial state, which is not the case by default planning instances; as well as it assumes that all domains have polynomial -time solutions.

References

Bonet, B.; and Geffner, H. 2021. General policies, representations, and planning width. In *AAAI*.

Fern, A.; Khardon, R.; and Tadepalli, P. 2011. The first learning track of the international planning competition. *Machine Learning*, 84(1-2): 81–107.

Helmert, M. 2006. New Complexity Results for Classical Planning Benchmarks. In *ICAPS*.

Jiménez, S.; Segovia-Aguas, J.; and Jonsson, A. 2019. A review of generalized planning. *KER*, 34.

Richter, S.; and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *JAIR*, 39: 127–177.

Segovia-Aguas, J.; Celorrio, S. J.; Sebastiá, L.; and Jonsson, A. 2022. Scaling-up generalized planning as heuristic search with landmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 15, 171–179.

Segovia-Aguas, J.; Jiménez, S.; and Jonsson, A. 2021. Generalized Planning as Heuristic Search. In *ICAPS*.

Seipp, J.; and Helmert, M. 2018. Counterexample-guided Cartesian abstraction refinement for classical planning. *JAIR*, 62: 535–577.