A Sampling Approach to Planning with Infinite Domain Control Variables

Ángel Aso-Mollar[1], Diego Aineto[2], Enrico Scala[2], Eva Onaindia[1]

[1]Universitat Politècnica de València
[2]Università degli Studi di Brescia
{aaso,onaindia}@vrain.upv.es, {diego.ainetogarcia,enrico.scala}@unibs.it

Abstract

Research in planning has sought to broaden the scope of planning problems by incorporating numeric parameters into action descriptions to condition both continuous and discrete change. Focusing on the latter, this work studies the problem of numeric planning with *control variables*, a reformulation of actions with infinite domain parameters. To tackle the challenge of handling an infinite decision space driven by control variables, we incorporate sampling into a forward state-space search. The resulting search framework (1) partially expands nodes by sampling their successors and (2) implements a re-expansion strategy to sample additional successors if a node shows promise in future evaluations. We perform a deep probe into this concept that materializes into a new algorithm called Sampling Greedy Best-First Search (S-GBFS). Our empirical evaluation of S-GBFS across various domains shows significant improvements over existing planning approaches.

Code — https://github.com/aasomol/samplingCV

Introduction

Classical planning defines a discrete and finite state space, as the number of (grounded) actions is limited and determined by the objects of the problem. In PDDL, action parameters are associated with a particular object type, allowing planners to generate a finite number of action instances and conduct a complete search in the resulting finite decision space.

There has been increasing interest in extending the scope of classical planning by incorporating numeric parameters with infinite domains into action descriptions, referred to as *control parameters* in the literature. This is explored from two perspectives: (1) representing rates of continuous change, as in planners like Konming (Li and Williams 2008) and Scotty (Fernández-González, Karpas, and Williams 2015), which manage the gradual continuous change of variables during execution; and (2) representing instantaneous change, as in planners such as POPCORN (Savaş et al. 2016) and NextFLAP (Sapena, Onaindia, and Marzal 2024), where changes are not influenced by the passage of time.

Focusing on (2), both POPCORN and NextFLAP approach decision spaces induced by control parameters through a forward partial-order planning search. In both approaches, control parameters are defined as variables with dynamic bounds that adjust to satisfy numeric conditions, culminating in a final plan concretization step executed by a SAT or SMT solver in order to minimize makespan. Alternatively, infinite transition systems have been extensively studied in the Monte Carlo Tree Search (MCTS) literature (Couetoux et al. 2011; Lim, Tomlin, and Sunberg 2021), particularly through the concept of Progressive Widening (PW).

In this paper, we provide new insights and perspectives on planning within an infinite decision space, an area still largely underexplored in current literature. We introduce *control variables* as an equivalent reformulation to control parameters that better aligns with our line of work. To manage the complexities of this infinite decision space driven by control variables, we incorporate sampling within a forward state-space search. We draw inspiration from PW, employing a partial node expansion guided by a hyperparameter that adjusts the propensity to prioritize selection among existing children over the creation of new ones.

The resulting framework, Sampling Greedy Best-First Search (S-GBFS), partially expands nodes by sampling their successors and employs a re-expansion strategy to sample additional successors if a node demonstrates potential in future evaluations. Our empirical evaluation of S-GBFS across various domains reveals significant improvements over existing planning approaches. Our results demonstrate that S-GBFS consistently solves more problems than compared baselines, often achieving similar or better quality plans.

Background

We adopt the numeric planning formalization from Scala et al. (2020a) and adapt it to our needs. We consider a set of numeric variables, X, and a set of Boolean variables, F, referred to as *numeric state variables* and *Boolean state variables*, respectively. We define $V = X \cup F$ as the union of these sets. A **valuation** $v : V \rightarrow D$ maps each variable $\lambda \in V$ to a value in its domain D, denoted as $v[\lambda]$. A **numeric expression** is a polynomial expression over X, $\text{Expr}(X)$ being the set of all numeric expressions. A **constraint** involves equalities $f = b$ for variables $f \in F$, where $b \in \mathbb{B}$ ($\mathbb{B} = \{\top, \bot\}$), and inequalities $\xi \bowtie 0$, such that $\bowtie \in \{<, \leq, =, \geq, >\}$, where $\xi \in \text{Expr}(X)$, combined with logical operations $\wedge, \vee, \neg$. The set of constraints over V is

defined as Constr(V). **Propositional assignments**, defined as Assign$_P(F)$, consist of assignments $f := b$, where $f \in F$ and $b \in \mathbb{B}$, while **numeric assignments**, Assign$_N(X_1, X_2)$, are of the form $x := \xi$, where $x \in X_1$ and $\xi \in \text{Expr}(X_2)$.

Definition 1 *A **numeric planning problem** is a tuple $\mathcal{P} = (F, X, A, I, G)$, where:*

- *F is a finite set of Boolean state variables;*
- *X is a finite set of numeric state variables over $\mathbb{Q}$;*
- *A is a finite set of actions such that each action $a \in A$ is a pair $\langle \text{Pre}(a), \text{Eff}(a) \rangle$, where $\text{Pre}(a) \in \text{Constr}(X \cup F)$ and $\text{Eff}(a) \subseteq Assign_P(F) \cup Assign_N(X, X)$;*
- *I is a valuation over $X \cup F$ that describes the initial situation;*
- *$G \in \text{Constr}(X \cup F)$ is the goal condition.*

Definition 2 *The semantics of a **numeric planning problem** $\mathcal{P}$ is the transition system $\mathcal{T}(\mathcal{P}) = (S, s_0, S_G, \rightarrow)$ where:*

- *$S = \mathbb{B}^{|F|} \times \mathbb{Q}^{|X|}$ is the state space; a state $s \in S$ is a valuation over $X \cup F$;*
- *$s_0 \in S$ is the initial state described by I;*
- *$S_G = \{s \in S \mid s \models G\}$ are the goal states;*
- *$\rightarrow$ is the transition relation. A transition $(s, a, s') \in S \times A \times S$ belongs to $\rightarrow$ iff $s \models Pre(a)$ and s' is the result of changing the truth value of variables present in $\text{Eff}(a)$ assignments accordingly.*

The transition system of a numeric planning problem is countably infinite because its state space is a product of countably infinite sets. We define the *decision space* as the set of all possible actions that can be taken from a given state. In a numeric planning problem, this decision space is finite for each state.

Definition 3 *Let $\mathcal{P}$ be a numeric planning problem that induces the transition system $\mathcal{T}(\mathcal{P}) = (S, s_0, S_G, \rightarrow)$. A **plan** $\pi = (a_i)_{i=1}^n$ for $\mathcal{P}$ is a solution iff $(s_0, a_0, s_1), \ldots, (s_{n-1}, a_n, s_n) \in \rightarrow$ and $s_n \in S_G$.*

Numeric Planning with Control Variables

This section presents the main contributions of our work. First, we formulate a *numeric planning problem with control variables*, which reformulates the traditional numeric planning problem with numeric action parameters. Next, we introduce a search algorithm, Sampling-GBFS (S-GBFS), based on Greedy Best-First Search, designed to effectively handle the infinite decision space that arises when incorporating control variables.

Problem Formulation

In a *numeric planning problem with control variables*, we introduce a new set of numeric variables, termed *control variables*, U, alongside the already defined set of numeric state variables X. The evolution of control variables is unconstrained and they can take any value within their (bounded) infinite domain, i.e., an interval. We introduce a *precision* parameter $\rho \in \mathbb{N}$ to optionally limit the number of decimal places of the variable values within the interval. Concretely, being $l, u \in \mathbb{Z}$, $l < u$, we define $[l, u]_\rho = \left\{x \in [l, u] \mid x = \frac{m}{10^\rho}, m \in \mathbb{Z}\right\}$. Note that $[l, u]_0$ is equivalent to the discrete set $\{l, l+1, \ldots, u-1, u\}$ and, abusing notation, we define $[l, u]_\infty := [l, u]$.

In a numeric planning problem with control variables, the values of the state variables X change after the execution of actions, based on the assignments explicitly defined in their effects, which now incorporate control variables: Assign$_N(X, X \cup U)$. Moreover, control variables also appear in action preconditions.

Definition 4 *A **numeric planning problem with control variables** extends Definition 1, i.e., it is a tuple $\mathcal{P} = (F, X \cup U, A, I, G, \rho)$, where:*

- *Numeric state variables are enriched with control variables U, which is a finite set of bounded numeric variables over $[l, u]_\rho$, $l, u \in \mathbb{Z}$;*
- *Actions incorporate control variables, i.e., A is a finite set of actions $a = (\text{Pre}(a), \text{Eff}(a))$, where $\text{Pre}(a) \in \text{Constr}(X \cup F \cup U)$ and $\text{Eff}(a) \subseteq Assign_P(F) \cup Assign_N(X, X \cup U)$;*

Next, we characterize the transition system of a numeric planning problem with control variables.

Definition 5 *The semantics of a **numeric planning problem with control variables** $\mathcal{P} = (F, X \cup U, A, s_0, G, \rho)$ extends Definition 2, i.e., it is a transition system defined as $\mathcal{T}(\mathcal{P}) = (S, \mathcal{U}, s_0, S_G, \rightarrow)$ where:*

- *$\mathcal{U} = [l_1, u_1]_\rho \times \ldots \times [l_n, u_n]_\rho$ is the control space, where $\mu \in \mathcal{U}$ is a valuation over U;*
- *Decisions are enriched with valuations of control variables, i.e., a transition $(s, \langle a, \mu \rangle, s') \in S \times A \times \mathcal{U} \times S$ belongs to $\rightarrow$ iff $(s, \mu) \models Pre(a)$ and s' is the result of changing the truth value of variables present in $\text{Eff}(a)$ assignments accordingly.*

The semantics of this approach is equivalent to the one of control parameters, the only difference being that now numeric decisions are treated outside actions. Note that the decision space given by $A \times \mathcal{U}$ is infinite with $\rho = \infty$. Hereinafter, we assume $\rho = \infty$ unless otherwise stated. A plan in this setting is no longer a sequence of actions, but a sequence of pairs action-valuation of control variables:

Definition 6 *Let $\mathcal{P}$ be a numeric planning problem with control variables that induces the transition system $\mathcal{T}(\mathcal{P}) = (S, \mathcal{U}, s_0, S_G, \rightarrow)$. A **plan** $\pi = (\langle a_i, \mu_i \rangle)_{i=1}^n$ for $\mathcal{P}$ is a solution iff $(s_0, \langle a_0, \mu_0 \rangle, s_1), \ldots, (s_{n-1}, \langle a_n, \mu_n \rangle, s_n) \in \rightarrow$ and $s_n \in S_G$.*

Forward Search in an Infinite Decision Space

The main challenge of searching in an infinite decision space is that nodes may have infinitely many successors, meaning a node might never be fully expanded – unlike in finite search spaces. This renders standard forward search algorithms, like GBFS, unsuitable for our problem.

To tackle this challenge, we propose two modifications to a regular best-first search schema. First, we use **partial expansions** to incrementally generate subsets of a state's successors via sampling. Partially expanded states are re-added to the priority queue to maintain completeness, but

Algorithm 1: Sampling Greedy Best-First Search (S-GBFS)

Input: Transition system $\mathcal{T}(\mathcal{P}) = (S, \mathcal{U}, s_0, S_G, \rightarrow)$, sampling function f, heuristic h
Parameters: $\alpha > 0$
Output: Goal reached

```
1: Initialize priority list L and N(s) = 0, ∀s ∈ S
2: L.add(h(s_0), s_0)
3: while L not empty do
4:     s ← L.get_lowest()
5:     N(s) ← N(s) + 1
6:     if s ∈ S_G then
7:         return true
8:     else
9:         s' ← f(s, →)
10:        L.add(h(s'), s')
11:        L.add(h(s) + N(s)^α, s)
12:    end if
13: end while
14: return false
```

with their value adjusted based on a **re-expansion penalty**. This penalty reflects the number of times a state has been partially expanded, lowering the priority of re-expanding the same state versus states expanded fewer times or not at all. We can further adjust this penalty using a hyperparameter α that balances exploration and exploitation by influencing the likelihood of re-expanding the same state in consecutive steps. There are many ways these abstract ideas can be instantiated into a concrete algorithm. Following, we describe one such instantiation, which we will use to evaluate the feasibility of our proposal in the following section.

The Sampling Greedy Best-First Search (S-GBFS) algorithm incorporates partial expansions and re-expansion penalties in a GBFS scheme. Algorithm 1 outlines S-GBFS. It starts by initializing a priority list L and a table N to track partial expansions (line 1). The initial state is added to L (line 2), and the main loop begins (line 3). In each iteration, the state with the lowest heuristic value is selected (line 4), and its partial expansion count is incremented (line 5). If the state is a goal state (line 6), the algorithm returns that the goal has been reached (line 7). If not (line 8), one successor s' is generated by partially expanding s, using a sampling function f (line 9), added to L (line 10), and s being reinserted with its value penalized by $N(s)^\alpha$ (line 11).

Completeness Several studies analyze the completeness of algorithms using random exploration, such as exploration-based GBFS and RRTs (Valenzano and Xie 2016; Kleinbort et al. 2019). Since both operate on infinite graphs, completeness cannot be guaranteed in finite steps. Instead, sampling-based planning introduces **probabilistic completeness**, meaning the probability of finding a solution (assuming at least one exists) approaches 1 as samples increase indefinitely. In our case, a proof can be constructed to guarantee the probabilistic completeness of S-GBFS if (1) the sampling function f assigns a nonzero probability to each subset of the sampling interval – recall that, although in continuous distributions the probability of sampling a specific point is zero, the purely continuous problem is computationally infeasible, as computers operate with finite precision in decimal numbers; when we assume $\rho = \infty$, we are trully assuming $0 \ll \rho < \infty$ which is machine-specific. And (2) observing that the re-expansion penalty is monotonically increasing with respect to the number of times a state has been visited. (1) guarantees that every state will eventually enter the priority list (the probability of never sampling a subset $[x, x + \varepsilon]$ tends to zero $\forall \varepsilon > 0$) and (2) guarantees that no state is endlessly prioritized.

Experiments

We empirically evaluate S-GBFS in two different scenarios:

1. **Control parameter domains**: we utilize the three domains introduced in POPCORN: CASHPOINT, PROCUREMENT, and TERRARIA, adapted to control variables with continuous domains ($\rho = \infty$). Unfortunately, POPCORN has been confirmed not to be in a compilable state after consultation with its authors, so we can only establish a baseline with respect to NextFLAP.
2. **Numeric planning domains**: by setting the number of decimals to zero ($\rho = 0$), we can handle environments with actions with integer parameters. This flexibility enables us to redefine certain type of numeric domains to accommodate different size increments instead of unitary. We tested four domains from the latest numeric IPC[1]: COUNTERS, BLOCKS-GROUPING, DRONE, and SAILING. As baseline, we solved the original instances with standard GBFS using ENHSP (Li et al. 2018), a state-of-the-art heuristic numeric planning system.

Experimental setup We implemented prototype versions of each domain in Python using the goal-counting heuristic h_{GC} and uniform sampling as f. Hyperparameter α is fixed to 0.01 except in the DRONE domain, which is fixed to 2. We conducted 10 different runs of S-GBFS for each problem in order to have a representative sample, and we show average results with a 95% confidence interval. The purpose of this experimentation is to explore the algorithm's performance even when employing such a straightforward heuristic. We defined 20 problems for each domain, collecting information about the number of expansions, cpu-time, number of actions of the solution plans, and coverage of each approach. All experiments were run on a 12th Gen Intel(R) Core(TM) i9-12900KF CPU and Ubuntu 22.04 LTS operating system, with a 10-minute timeout and an 8 GB memory limit for each problem, the same for all approaches.

Coverage analysis Firstly, it has been noticed that both baselines solve much less problems than S-GBFS. We analyse this aspect by looking at the coverage of each approach considering the time taken to solve each problem and also problem size, measured by the number of objects. As shown in Figure 1, S-GBFS solves more problems than the other approaches (top section) in less time, except for GBFS-h_{MRP}, which momentarily outperforms S-GBFS. All baseline methods struggle with larger problem instances, as indicated in the bottom section of the figure, unlike S-GBFS.

[1] https://github.com/ipc2023-numeric

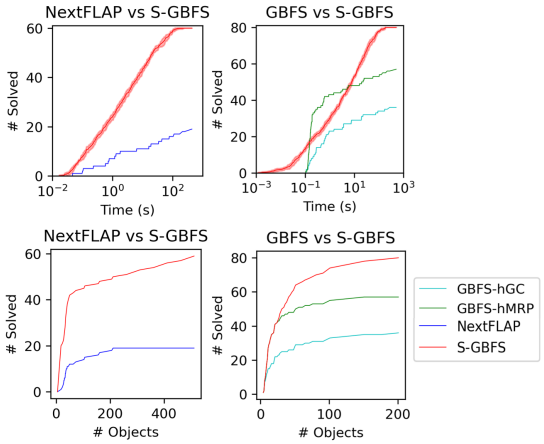

Figure 1: Cumulative plot of number of problems solved against time spent (top) and against problem size (bottom), for baselines 1 (left) and 2 (right)

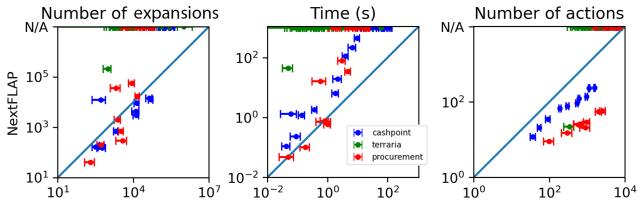

Figure 2: Number of expansions, time and number of actions of S-GBFS w.r.t NextFLAP planner

Control Parameters Benchmarks Results for control parameters domains are shown in Figure 2. In this comparison, we see that both NextFLAP and S-GBFS require a comparable number of expansions, but S-GBFS is generally faster than NextFLAP. Further evidence is given by the large number of points in the upper edge, denoting NextFLAP timeouts. In problems solved by both approaches, NextFLAP finds shorter plans, which we speculate is due to the fact that the heuristic employed, which is based on h_{FF} (Hoffmann 2003), is more informed than h_{GC}. There is also a crucial difference in plans, since NextFLAP/POPCORN guarantee a minimization of the values of control parameters, which is not contemplated in our approach. This may be an interesting line work to consider new sampling functions.

Numeric Planning Benchmarks For the numeric planning setting, we first conducted a comparison against GBFS equipped with the h_{GC} heuristic, to analyse both algorithms on level ground, and then we equipped GBFS with the more informed h_{MRP} heuristic (Scala et al. 2020b), which has demonstrated strong performance in numeric planning. Results are shown in Figure 3. We observed that S-GBFS with h_{GC} outperforms both GBFS-h_{GC} and GBFS-h_{MRP} in the number of expansions and in time. Results in number of actions are aligned with GBFS-h_{GC} but worse compared to GBFS-h_{MRP}, which makes sense as the heuristic is more informed. We noticed that S-GBFS showed somewhat lower performance in the SAILING domain, especially with respect

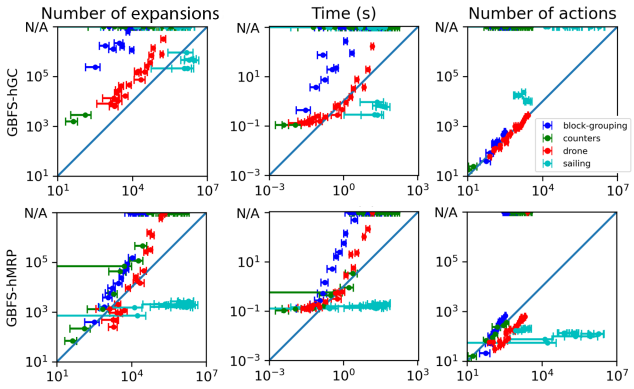

Figure 3: Number of expansions, time and number of actions of S-GBFS w.r.t GBFS-h_{GC} and GBFS-h_{MRP}

to GBFS-h_{MRP}. This is likely due to the fact that numeric variables are unbounded in SAILING. We believe that probabilistic completeness could be compromised in this type of unbounded domains, due to the possible presence of infinite cycles inherent to this type of problems that translate into infinite graphs. In future work, we plan to investigate this anomaly further, but our preliminary intuition suggests that it may be the result of a combination of a poorly-informed heuristic and an overly simplistic sampling technique.

In summary, S-GBFS has consistently solved a substantially larger number of problems across all baselines. Additionally, we have examined the potential of this approach in classical numeric planning domains, where incorporating numeric action parameters has shown to be advantageous. The choice of α has shown to be domain-sensitive. A small α only slightly penalizes the parent node's heuristic value, promoting breadth over depth-first exploration, which was sufficient to solve the majority of domains except from DRONE, which is the only domain with a high number of dead-ends. In that particular case, a quadratic penalty delays revisiting parent nodes, avoiding situations in which the algorithm searches hopelessly between sibling nodes. This indicates that further research on penalty functions is needed.

Conclusion

Our work highlights the potential of leveraging numeric action parameters to expand the expressiveness of planning problems. By redefining control parameters as control variables and introducing S-GBFS, we provide a novel approach that effectively navigates infinite decision spaces driven by numeric action parameters. S-GBFS experimentally outperforms existing planners, both in control parameters and numeric domains.

We will analyze S-GBFS with more sophisticated heuristics, exploring the impact of re-expansion criteria and sampling functions. We also aim to examine how the hyperparameter α affects performance across specific domains. Following recent work (Wissow and Asai 2024), we intend to implement S-GBFS as a MCTS by directly utilizing Progressive Widening. This work represents an exploratory yet promising approach.

Acknowledgements

This work was partially supported by the project I+D+i AEI PID2021-127647NB-C22 funded by MICIU/AEI/10.13039/501100011033 and by FEDER/UE; and by the GENERALITAT VALENCIANA project PROMETEO CIPROM/2023/23. This work was also partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU, and by the Climate Change AI project (No. IG-2023-174). Angel Aso-Mollar is partially supported by the FPU21/04273.

References

Couetoux, A.; Hoock, J.-B.; Sokolovska, N.; Teytaud, O.; and Bonnard, N. 2011. Continuous Upper Confidence Trees. In *LION 2011*. Italy.

Fernández-González, E.; Karpas, E.; and Williams, B. C. 2015. Mixed Discrete-Continuous Heuristic Generative Planning Based on Flow Tubes. In *IJCAI 2015*, 1565–1572. AAAI Press.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J. Artif. Intell. Res.*, 20: 291–341.

Kleinbort, M.; Solovey, K.; Littlefield, Z.; Bekris, K. E.; and Halperin, D. 2019. Probabilistic Completeness of RRT for Geometric and Kinodynamic Planning With Forward Propagation. *IEEE Robotics and Automation Letters*, 4(2): i–vii.

Li, D.; Scala, E.; Haslum, P.; and Bogomolov, S. 2018. Effect-Abstraction Based Relaxation for Linear Numeric Planning. In *IJCAI 2018*, 4787–4793. International Joint Conferences on Artificial Intelligence Organization.

Li, H.; and Williams, B. 2008. Generative Planning for Hybrid Systems Based on Flow Tubes. In *ICAPS 2008*, 206–213.

Lim, M. H.; Tomlin, C. J.; and Sunberg, Z. N. 2021. Voronoi Progressive Widening: Efficient Online Solvers for Continuous State, Action, and Observation POMDPs. In *CDC 2021*, 4493–4500.

Sapena, O.; Onaindia, E.; and Marzal, E. 2024. A Hybrid Approach for Expressive Numeric and Temporal Planning with Control Parameters. *Expert Systems with Applications*, 242: 122820.

Savaş, E.; Fox, M.; Long, D.; and Magazzeni, D. 2016. Planning Using Actions with Control Parameters. In *ECAI 2016*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 1185–1193. IOS Press.

Scala, E.; Haslum, P.; Thiebaux, S.; and Ramirez, M. 2020a. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *Journal of Artificial Intelligence Research*, 68: 691–752.

Scala, E.; Saetti, A.; Serina, I.; and Gerevini, A. E. 2020b. Search-Guidance Mechanisms for Numeric Planning Through Subgoaling Relaxation. In *ICAPS 2020*, volume 30, 226–234.

Valenzano, R.; and Xie, F. 2016. On the Completeness of Best-First Search Variants That Use Random Exploration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

Wissow, S.; and Asai, M. 2024. Scale-Adaptive Balancing of Exploration and Exploitation in Classical Planning. arXiv:2305.09840.