BLAST: Bit-Blasting Numbers for Classical Planning (Extended Abstract)*

Luigi Bonassi[1], **Francesco Percassi**[2], **Enrico Scala**[3]
[1]Oxford Robotics Institute, University of Oxford
[2]School of Computing and Engineering, University of Huddersfield
[3]Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia
luigibonassi@robots.ox.ac.uk,f.percassi@hud.ac.uk,enrico.scala@unibs.it

Introduction

Automated planning is a relevant area within Artificial Intelligence that focuses on synthesising sequences of actions to achieve a goal specification given an initial state and a domain model. A notable formulation of automated planning is numeric planning, which introduces numeric state variables that can be modified by actions and used in preconditions and goal state specification. Numeric planning has gained renewed interest due to its expressive power, which makes it well-suited for modelling real-world aspects such as resource consumption or naturally expressing concepts like positions in a grid-based environment.

However, numeric planning, in its general form, is undecidable. This has motivated research into bounded or restricted versions of numeric planning that are decidable. Gigante and Scala (2023) showed that numeric planning can be compiled into classical planning under certain limitations. Specifically, they focused on bounded numeric planning tasks, where the domain of each numeric variable is finite and enumerable. They demonstrated that such tasks can be transformed into equivalent classical planning tasks using only Boolean state variables and conditional effects. The backbone of this compilation consists of representing numeric variables in binary format and encoding numeric effects through conditional effects that simulate logic circuits operating on binary vectors. Most importantly, their compilation is polynomial in size and plan-size preserving, meaning that the length of the resulting classical plan is identical to that of the original numeric plan. This property is relevant in the context of compilation among planning formalisms, as transformations that do not preserve plan size are often impractical or introduce significant overhead in the planning process (Nebel 2000).

In this work, we explore the practical implications of this theoretical framework. While the result by Gigante and Scala (2023) offers valuable insights, it remains unclear whether it provides a viable path toward practical applicability. We address this by instantiating their approach for a restricted class of numeric planning tasks, specifically, simple numeric planning (SNP). This is an important fragment in which numeric variables can only be increased or decreased by a constant value, and numeric conditions are limited to linear combinations. In doing so, we pose the central question: *how far can classical planners go when applied to solve (bounded) numeric planning tasks*?

To this end, we propose two new compilations from SNP into classical planning with conditional effects: BLAST and BLAST$_{\mathcal{X}}$. Both build on the core idea introduced by Gigante and Scala (2023). The former is more closely aligned with their original approach; the latter is tailored to produce more compact planning tasks by combining conditional effects with axioms in a complementary way (Bonassi et al. 2023; Speck, Seipp, and Torralba 2025).

BLAST and BLAST$_{\mathcal{X}}$ Compilations

In the BLAST compilation, each numeric variable is encoded in two's complement using a fixed-width binary vector $\langle v_{\kappa-1}, \dots, v_0 \rangle$, where κ is the number of bits and $v_{\kappa-1}$ is the sign bit. Numeric conditions $\langle \xi \geq 0 \rangle$ are normalised by introducing a fresh variable z_ξ tracking the value of ξ; the condition is then encoded as a Boolean test over z_ξ, using its sign bit.

To model numeric effects like $v{=}v{+}q$, BLAST simulates a full adder using conditional effects. Let $\langle q_{\kappa-1}, \dots, q_0 \rangle$ denote the binary representation of the constant q in two's complement; then the sum bit z_i at position i is defined as $z_i = (v_i \oplus q_i) \oplus c_{i-1}$, where c_{i-1} is the carry from the previous position. The carry bits are defined recursively as $c_i = (v_i \wedge q_i) \vee (c_{i-1} \wedge (v_i \oplus q_i))$, with base case $c_{-1} = \bot$. For each bit i, BLAST introduces two conditional effects that update v_i depending on whether z_i holds.

Overflow detection is handled by monitoring the sign bits: an overflow occurs if the signs of the operands are the same but differ from the result. This is encoded using additional conditional effects that activate a Boolean overflow flag.

While the compilation is sound and, for sufficiently large κ, complete, the recursive structure of carry computation causes the size of the formulae to grow with the bit width, which may affect the scalability of the planning systems.

BLAST$_{\mathcal{X}}$ builds on the BLAST compilation by introducing the use of axioms to simplify the encoding of numeric effects. Instead of directly representing the recursive structure of carry propagation within conditional effects, BLAST$_{\mathcal{X}}$ shifts the calculation of sum and carry bits from the action

*We report results from our previously published work (Bonassi, Percassi, and Scala 2025)

effects into a set of derived variables, whose values are determined by axioms. This modularisation reduces formula size, making it constant regardless of bit width, and improves compilation scalability while maintaining the same theoretical guarantees of BLAST.

A potential drawback of BLAST$_{\mathcal{X}}$ lies in the generation and evaluation of axioms: for each numeric effect in the original task, a corresponding set of axioms must be compiled to compute the associated sum and carry bits. These axioms must then be evaluated at each planning step, potentially increasing the computational load. In the experimental evaluation, we will analyse the practical trade-offs of this approach and assess in which contexts axioms provide a tangible advantage over fully explicit encodings.

Experiments

We conducted an experimental evaluation to assess the practical effectiveness of two binary compilations, available at https://github.com/LBonassi95/BitBlast. On the classical side, we tested them with LAMA (Richter and Westphal 2010), chosen for its native support for axioms. The evaluation focused on computing the first solution only, referred to as LAMA-F. Since the compilations are natively bounded, the number of bits κ was set per domain based on the maximum constant appearing in it. On the numeric side, we considered three state-of-the-art planners, selected to represent diverse planning architectures: NLM-CUTPLAN with the SAT2 configuration (Kuroiwa et al. 2022); ENHSP (Scala et al. 2020), with the M$(3h||3n)$ configuration (Chen and Thiébaux 2024); and PATTY (Cardellini, Giunchiglia, and Maratea 2024), using its default configuration. A timeout of 1800 seconds was imposed on all planners, including compilation time, where applicable.

Our benchmarks were drawn from the latest International Planning Competition (IPC-23), and feature different SNP domains that we classify as *strongly numeric* (SN) or *mildly numeric* (MN) based on the proportion of numeric variables relative to the total number of variables, indicated by $\mathfrak{N}$.

The results in terms of coverage (i.e., number of problems solved) are reported in Table 1. The analysis suggests that the compilations can be competitive in certain MN domains, whereas in SN domains, the use of native numeric planners is clearly preferable. Moreover, the results highlight the critical role of axioms in achieving better performance.

Future Works

In conclusion, BLAST and BLAST$_{\mathcal{X}}$ offer promising compilations for simple numeric planning, with competitive performance in mildly numeric domains. Future work includes extending the approach to richer numeric formalisms and designing compilations optimised for bit-space requirements, e.g., via bound estimation techniques.

Acknowledgements

Luigi Bonassi was supported by the joint UKRI and AISI-DSIT Systemic Safety Grant [grant number UKRI854]. Francesco Percassi was supported by a UKRI Future Leaders Fellowship [grant number MR/Z00005X/1]. Enrico Scala was supported by MUR projects PRIN2020 RIPER and PNRR PE0000013-FAIR.

Domain (#)	$\mathfrak{N}$	κ	LAMA-F B	LAMA-F B$_{\mathcal{X}}$	S	P	M
BGrouping (20)	1.0	8	2	2	0	**20**	16
Counters (20)	1.0	8	5	5	12	**20**	10
Watering (20)	1.0	9	0	2	19	6	**20**
Farmland (20)	1.0	15	0	4	15	**20**	**20**
MTrader (20)	0.94	15	0	0	2	7	**20**
Sailing (20)	0.88	12	0	5	10	19	**20**
Pathways (20)	0.7	15	14	14	1	**20**	3
Sugar (20)	0.64	7	12	11	6	**20**	13
Σ_{SN} (160)	—	—	33	43	65	**132**	122
Settlers (20)	0.41	5	**15**	**15**	2	*n/a*	2
Expedition (20)	0.39	11	0	3	4	3	**9**
HPower (20)	0.32	17	0	1	18	**20**	**20**
Rovers (20)	0.12	8	4	15	8	**18**	16
MPrime (20)	0.08	5	**20**	**20**	12	17	19
Delivery (20)	0.02	6	**20**	**20**	9	5	**20**
Σ_{MN} (120)	—	—	59	74	53	63	**86**
Σ (280)	—	—	92	117	118	195	**208**

Table 1: Coverage. $\mathfrak{N}$: domain numericity; κ: number of bits. NLM-CUTPLAN (S), PATTY (P), ENHSP (M). Bold indicates the overall best result; underline marks the best among compilations. *n/a* denotes planner-domain incompatibility.

References

Bonassi, L.; Giacomo, G. D.; Favorito, M.; Fuggitti, F.; Gerevini, A. E.; and Scala, E. 2023. FOND Planning for Pure-Past Linear Temporal Logic Goals. In *ECAI*, 279–286. IOS Press.

Bonassi, L.; Percassi, F.; and Scala, E. 2025. Towards Practical Classical Planning Compilations of Numeric Planning. In *AAAI*, 26472–26480.

Cardellini, M.; Giunchiglia, E.; and Maratea, M. 2024. Symbolic Numeric Planning with Patterns. In *AAAI*, 20070–20077.

Chen, D. Z.; and Thiébaux, S. 2024. Novelty Heuristics, Multi-Queue Search, and Portfolios for Numeric Planning. In *SOCS*, 203–207. AAAI Press.

Gigante, N.; and Scala, E. 2023. On the Compilability of Bounded Numeric Planning. In *IJCAI*, 5341–5349.

Kuroiwa, R.; Shleyfman, A.; Piacentini, C.; Castro, M. P.; and Beck, J. C. 2022. The LM-Cut Heuristic Family for Optimal Numeric Planning with Simple Conditions. *J. Artif. Intell. Res.*, 75: 1477–1548.

Nebel, B. 2000. On the Compilability and Expressive Power of Propositional Planning Formalisms. *J. Artif. Intell. Res.*, 12: 271–315.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.*, 39: 127–177.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2020. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *J. Artif. Intell. Res.*, 68: 691–752.

Speck, D.; Seipp, J.; and Torralba, Á. 2025. Symbolic Search for Cost-Optimal Planning with Expressive Model Extensions. *J. Artif. Intell. Res.*, 82: 1349–1405.