Investigating the Grounding Bottleneck for a Large-Scale Configuration Problem: Existing Tools and Constraint-Aware Guessing - Extended Abstract

Veronika Semmelrock[1] and Gerhard Friedrich[1]

Department of Artificial Intelligence and Cybersecurity
University of Klagenfurt, Austria

1 Introduction and Problem Description

This extended abstract summarizes our investigation into the scalability of Answer Set Programming (ASP) for industrial-scale configuration problems, presented as a TC paper at ICLP 2025 and published in EPTCS.
The long-standing AI vision is simple: the user specifies the problem, and the computer solves it automatically. ASP has realized this in many domains, yet scaling to large industrial problems remains a challenge. Since grounding can require super-linear space in the input size, ASP applications often become infeasible due to high memory demands, which is the phenomenon known as the *grounding bottleneck*. While ASP is expressive and powerful, it often depends on expert-crafted encodings and optimizations to scale. Our research addresses a fundamental question: How far can current ASP techniques scale for large configuration problems if we rely solely on the solvers, without manual optimization? As a benchmark, we use the configuration of electronic systems, which in industrial settings may involve more than 30,000 components. To systematically analyze scalability, we adopt the simplified *House Configuration Problem* (HCP), which preserves the essential structural and combinatorial characteristics of such systems. Here, the key configurable elements are *things*. Our study evaluates the limits of current solvers and identifies techniques to push these boundaries.

2 Benchmarking Grounding Strategies in ASP

We benchmarked representative state-of-the-art ASP solvers and grounding strategies: `clingo` [6], `DLV` [1], and `I-DLV+clasp` (standard ground-and-solve), `Alpha` (lazy grounding with domain-specific heuristics) [3], `ProASP` (compilation-based hybrid solving) [4], and `newground` (body-decoupled hybrid grounding) [2]. Experimental instances were systematically scaled, increasing the number of *things* while maintaining proportional resource domains to ensure that a solution can be found. All solvers were tested with a one-hour timeout on identical hardware. The results show severe scalability limitations: `clingo`, `DLV`, and `I-DLV+clasp` reached instance sizes of only about 400 *things* before memory exhaustion;

`ProASP` achieved slightly smaller solvable instances (approx. 350 *things*); `newground` stalled at about 100 *things*; and `Alpha` performed best among non-incremental systems, reaching approx. 2,100 *things* using lazy grounding. None of these solvers approached the real-world scale of industrial configuration tasks. However, simple and effective heuristics for configuring such systems exist [5]. These heuristics enable incremental extension of partial solutions, a key to applying AI technology in large-scale configuration. We therefore investigated the potential of ASP solvers under this favorable assumption, exploring *incremental solving*, where partial configurations are iteratively extended until a complete configuration is reached. We implemented an incremental solving procedure using `clingo`, `DLV`, `ProASP`, `newground`, and `I-DLV+clasp`. Each iteration extended a partial answer set with new facts, controlled by the *persons-per-iteration* (PPI) parameter. All systems benefited from incrementalization: `incremental-clingo` scaled up to 2,850 *things* (approx. 425% improvement), `DLV` and `newground` showed relative gains up to 700%, and `incremental-ProASP` reached approx. 5,900 *things*, which is over 1,500% improvement compared to its one-shot version. Although incremental solving improved runtime, grounding size remained the dominant bottleneck. We therefore developed *Constraint-Aware Guessing* (CAG), a grounding-aware rewriting method that filters inconsistent guesses during grounding. By adding filter conditions derived from existing constraints to the guessing rules, CAG prevents the generation of rule instances that would inevitably violate constraints. CAG was implemented on top of incremental `clingo`, replacing standard guessing rules with filtered versions. When applied to the HCP, CAG achieved up to a 99% reduction in grounding size and solved instances with up to 6,200 *things* within the one-hour time limit, thus surpassing all other methods. The grounding phase dominated the runtime (approx. 94%), while solving required less than 2%.

3 Conclusion

Current ASP technology without specialized encodings cannot yet handle full-scale industrial configurations involving tens of thousands of components. Incremental solving and CAG substantially improve scalability, but even the best-performing approaches still require improvements by several orders of magnitude to reach real-world applicability. At present, carefully engineered, problem-specific, and potentially multi-shot encodings remain necessary to address large-scale configuration problems such as the HCP. This highlights an ongoing challenge for the ASP community: achieving scalability without relying on extensive expert optimization. Future work will therefore focus on integrating symbolic and subsymbolic methods to advance toward this goal and further realize the long-standing vision of automated problem solving in AI.

References

1. Adrian, W.T., Alviano, M., Calimeri, F., Cuteri, B., Dodaro, C., Faber, W., Fuscà, D., Leone, N., Manna, M., Perri, S., et al.: The asp system dlv:

advancements and applications. KI-Künstliche Intelligenz **32**, 177–179 (2018). https://doi.org/10.1007/S13218-018-0533-0
2. Beiser, A., Hecher, M., Unalan, K., Woltran, S.: Bypassing the asp bottleneck: hybrid grounding by splitting and rewriting. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence. pp. 3250–3258 (2024), https://www.ijcai.org/proceedings/2024/360
3. Comploi-Taupe, R., Friedrich, G., Schekotihin, K., Weinzierl, A.: Domain-specific heuristics in answer set programming: A declarative non-monotonic approach. Journal of Artificial Intelligence Research **76**, 59–114 (2023). https://doi.org/10.1613/JAIR.1.14091
4. Dodaro, C., Mazzotta, G., Ricca, F.: Blending grounding and compilation for efficient asp solving. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning. vol. 21, pp. 317–328 (2024). https://doi.org/10.24963/KR.2024/30
5. Fleischanderl, G., Haselböck, A.: Thoughts on partitioning large-scale configuration problems. In: AAAI 1996 Fall Symposium Series. pp. 1–10 (1996)
6. Gebser, M., Kaminski, R., Kaufmann, B., Lindauer, M., Ostrowski, M., Romero, J., Schaub, T., Thiele, S., Wanko, P.: Potassco guide version 2.2.0 (2019), https://github.com/potassco/guide/releases/tag/v2.2.0, retrieved from https://github.com/potassco/guide/releases/tag/v2.2.0.