Distributed Answer Set Coloring: Stable Models Computation via Graph Coloring

Marco De Bortoli mbortoli@ist.tugraz.at

[1] Institute for Software Technology, Graz University of Technology, Graz, Austria
[2] Dept DMIF, University of Udine, Udine, Italy
[3] Dept CS, New Mexico State University, Las Cruces, NM, USA

Abstract. In this paper we present a distributed solver for ASP, based on the well-known Graph Coloring algorithm, implemented via the Boost and MPI libraries for C++.

1 Introduction and Problem Description

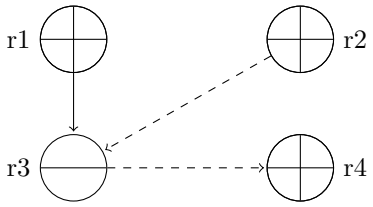
The Answer Set Programming (ASP) language has become very popular in the last years thanks to the availability of more and more efficient solvers (e.g., Clingo [5] and DLV [1]). As described, e.g., in [3], ASP has some important weakness when dealing with real-world complex problems, like planning [4, 8], which generates huge ground programs exceeding the resources of a single machine. Lazy grounding is an attempt to address this problem [2, 9], but it is not effective for every domain.
Our contribution consists into a distributed ASP solver, called Distributed Answer Set Coloring (DASC), which automatically splits a ground program over a network, using the resources of a single computational node to process only a portion of the original program. To accomplish this, we use the Graph Coloring algorithm [7], which represents the program as a graph, and its stable models as different colorings of its vertices.
DASC is not the first attempt of this sort: it was born with the purpose of lowering the implementation level of the mASPreduce solver, a tool developed by Federico Igne for his Master thesis [6], in order to address its performance. mASPreduce is indeed developed with the distribution framework Apache Spark, which, although it is a very powerful and expressive framework for distributed programming, gives to the user very low control over the communication flow, and it is the real performance killer during such kind of distributed computations, as can be seen from the experimental results.
We handled this communication control problem by developing DASC with C++, using the MPI library for messages handling and the Parallel Boost Graph Library to represent the distributed graph to color.

2 Rule Dependency Graph and Graph Coloring Algorithm

DASC is a distributed implementation of the Graph Coloring Algorithm for solving, obtained by splitting over a set of computational nodes the Rule Dependecy Graph (RDG) of the ground program, in order to let each node to work in its partition of the graph, with its own resources, without duplicating any data.

```
r1   p.
r2   q.
r3   r :- p, not q.
r4   s :- not r.
```

Answer Set = {p,q,s};



Above, the reader can see a simple ground program together with its RDG. Executing the Graph Coloring algorithm on that, we obtain (if any) the stable models in terms of colorings, namely maps from the set of rules to the set $\{\oplus, \ominus\}$. Given a total coloring, we deduce the corresponding stable model by taking the heads of all $\oplus$ colored rules, as shown above.

Theorem 1 (Operational answer set characterization). *Let Γ RDG and $\mathcal{C}$ total coloring of Γ, with $\mathcal{P}, \mathcal{V}$ propagation operators, $\mathcal{D}$ as the non-deterministic guess operator and $*$ the fix-point operator (see [7] for more details about the operators). $\mathcal{C}$ is an admissible coloring of Γ iff there exists a sequence $(\mathcal{C}^i)_{0 \leq i \leq n}$ such that:*

- $\mathcal{C}^0 = (\mathcal{PV})^*_\Gamma((\emptyset, \emptyset))$;
- $\mathcal{C}^{i+1} = (\mathcal{PV})^*_\Gamma(\mathcal{D}^\circ_\Gamma(\mathcal{C}^i))$ *for some* $\circ \in \{\oplus, \ominus\}$ *and* $0 \leq i < n$;
- $\mathcal{C}^n = C$.

From $\mathcal{C}^n$ a stable model is deduced.

Table 1. Comparison between mASPreduce and DASC on a Toy Example. Times in seconds.

	1 cp unit		2 cp units		3 cp units		4 cp units		5 cp units	
inst	mASPr	DASC	mASPr	DASC	mASPr	DASC	mASPr	DASC	mASPr	DASC
2	56.330	0.003	42.190	0.010	40.160	0.011	41.177	0.013	35.405	0.014
3	95.697	0.048	64.315	0.14	61.767	0.151	62.845	0.16	54.144	0.172
4	150.82	0.36	88.043	1.11	89.145	1.393	89.695	1.115	78.178	1.232
5	error	1.83	error	6.118	error	6.18	error	6.226	error	5.256
6	stopped	7.03	stopped	22.513	stopped	20.765	stopped	20.881	stopped	18.511
7	stopped	21.99	stopped	71.07	stopped	81.55	stopped	66.43	stopped	65.83
8	stopped	58.90	stopped	185.46	stopped	185.45	stopped	182.33	stopped	191.27)

3 Conclusion and Future Work

DASC represents a step forward in building a tool capable of exploiting distributed system resources in order to manage huge-size programs. Yet, we are still far from achieving the goal of handling large problems, and a lot of work has to be done to make our tool competitive with state-of-the-art solvers.
Moreover, the performance difference between our tool and mASPreduce is pretty huge, so lowering the level of implementation paid off, together with the development of a different propagation technique, the so-called *notify_change* approach.
To confirm C++ boost improvement with respect to Spark, in the instances in which Clingo exceeds 10ms, the minimum machine time, DASC is about 500 times slower; STRASP [10] instead, the distributed grounder developed by Pietro Totis in his thesis using Spark, capable of solving stratified programs (non-definite programs solvable without non-determinism in polynomial time), is about 2000 times slower than Clingo [5].

Finally, we present below the roadmap for DASC:

- improving the initial distribution or changing the redistribution algorithm with a more sophisticated one;
- implementing local multithreading;
- implementing heuristics, like the strong techniques used by Clingo, namely clause learning and backjumping.

References

1. Weronika T. Adrian, Mario Alviano, Francesco Calimeri, Bernardo Cuteri, Carmine Dodaro, Wolfgang Faber, Davide Fuscà, Nicola Leone, Marco Manna, Simona Perri, Francesco Ricca, Pierfrancesco Veltri, and Jessica Zangari. The ASP system DLV: advancements and applications. *KI*, 32(2-3):177–179, 2018.
2. Jori Bomanson, Tomi Janhunen, and Antonius Weinzierl. Enhancing lazy grounding with lazy normalization in answer-set programming. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, pages 2694–2702. AAAI Press, 2019.
3. Alessandro Dal Palù , Agostino Dovier, Enrico Pontelli, and Gianfranco Rossi. GASP: answer set programming with lazy grounding. *Fundam. Inform.*, 96(3):297–322, 2009.
4. Agostino Dovier, Andrea Formisano, and Enrico Pontelli. An empirical study of constraint logic programming and answer set programming solutions of combinatorial problems. *JETAI*, 21(2):79–121, 2009.
5. Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694, 2014.
6. Federico Igne. Analysis and development of a distributed asp solver using mapreduce. Master's thesis, University of Udine, 2017.

7. Kathrin Konczak, Thomas Linke, and Torsten Schaub. Graphs and colorings for answer set programming. *TPLP*, 6(1-2):61–106, 2006.
8. Tran Cao Son and Enrico Pontelli. Planning for biochemical pathways: A case study of answer set planning in large planning problem instances. In Marina De Vos and Torsten Schaub, editors, *Proceedings of the First International SEA'07 Workshop, Tempe, Arizona, USA*, volume 281 of *CEUR Workshop Proceedings*, pages 116–130, 01 2007.
9. Richard Taupe, Antonius Weinzierl, and Gerhard Friedrich. Degrees of laziness in grounding - effects of lazy-grounding strategies on ASP solving. In Marcello Balduccini, Yuliya Lierler, and Stefan Woltran, editors, *Logic Programming and Nonmonotonic Reasoning - 15th International Conference, LPNMR 2019, Philadelphia, PA, USA, June 3-7, 2019, Proceedings*, volume 11481 of *Lecture Notes in Computer Science*, pages 298–311. Springer, 2019.
10. Pietro Totis. A distributed asp solver for stratified programs. Master's thesis, University of Udine, 2018.