dynPARTIX 2.0 - Dynamic Programming Argumentation Reasoning Tool

Günther CHARWAT [1] and Wolfgang DVOŘÁK
Institute of Information Systems E184, Vienna University of Technology, A-1040 Vienna, Austria

Abstract. Most reasoning tasks in abstract argumentation are in general computationally hard. One approach of dealing with such problems stems from the field of parameterized complexity theory. For so called fixed-parameter tractable algorithms, one identifies problem parameters, e.g. the graph parameter tree width, such that the run-time of algorithms heavily scales with the parameter but only polynomially with the input size. The dynPARTIX system turns these fixed-parameter tractability results into practice by implementing dynamic programming algorithms for the graph parameter tree width.

Aim of Work. The tool dynPARTIX 2.0 aims at the efficient evaluation of problems that are defined for abstract argumentation frameworks (AFs). By implementing algorithms that are based on fixed-parameter tractability we want to provide a system that turns complexity-theoretic results into practice. An important parameter of graphs (and hence for argumentation frameworks) is the tree width which is defined on certain tree decompositions of graphs. The fixed-parameter tractable algorithms that underlie dynPARTIX are defined on such tree decompositions. Our goal is to develop an easy-to-use tool that performs especially good for instances of small tree width and that is capable of handling large input instances.

Capabilities. First algorithms using tree decompositions for the evaluation of problems in this field where developed in [3]. The original prototype of dynPARTIX was developed in 2011 by Nopp et.al. [4]. It supported the computation of admissible as well as preferred semantics. The enhanced system dynPARTIX 2.0 significantly increases the overall run-time performance and furthermore introduces support for stable as well as complete semantics. A description of these novel algorithms can be found in [1].
The following features are provided by dynPARTIX 2.0:

- Support of *admissible*, *preferred*, *stable* and *complete* semantics.
- *Enumeration* of extensions.
- *Counting* the number of extensions (without explicit computation of extensions).
- Deciding *credulous* and *skeptical* acceptance.
- Simple command-line interface.
- Support of *normalized* as well as *semi-normalized* tree decompositions.

The system dynPARTIX 2.0 and a detailed documentation are publicly available at:
`www.dbai.tuwien.ac.at/research/project/argumentation/dynpartix`

[1] Corresponding Author: Günther Charwat, E-mail: `gcharwat@dbai.tuwien.ac.at`.

System Interface. As input, dynPARTIX expects an *argumentation framework* (AF) as introduced by Dung [2]. This AF is given as directed graph consisting of arguments (vertices) and an attack relation (directed edges in the graph). One may specify this AF using the ASPARTIX[2] input format:

```
arg(a). arg(b). % Arguments a and b.
att(a,b).       % Attack: a attacks b.
```

A possible program call may be of the following format:

```
./dynpartix -f inputfile.af -n semi -s admissible --cred a
```

Technical Background. After parsing the input (specified with `-f inputfile.af`) a normalized (`-n norm`) or semi-normalized (`-n semi`) tree decomposition is generated with help of the SHARP[3] framework. A *tree decomposition* is a mapping from an AF to a tree where the nodes in the tree contain bags of arguments from the AF. Each argument appears in at least one bag, adjacent arguments are together in at least one bag and bags containing the same argument are connected.

The width of the tree decomposition is defined as the maximal number of arguments in the bags, minus one. The *tree width* is the minimal width over all tree decompositions. Semi-normalized tree decompositions solely consist of binary branch nodes and exchange nodes where the latter allows for arbitrarily many arguments to be added to or removed from the bag of the node. Normalized tree decompositions are composed of binary branch nodes as well. Instead of exchange nodes they consist of introduction and removal nodes (where exactly one argument is added or removed) and leaf nodes.

Given a tree decomposition we apply a dynamic programming algorithm [1,3] corresponding to the specified semantics (e.g. `-s admissible`) and reasoning mode (e.g. `--cred a`). The tree is traversed in bottom-up order where at each node possible candidate solutions are computed. After a full traversal we obtain our result at the root node.

Performance. Our benchmarks show that the run-time performance of dynPARTIX heavily depends on the tree width, which reflects the theoretical results from [3]. Comparing our tool with a state-of-the-art reasoning system, i.e. ASPARTIX (in conjunction with dlv[4]), we achieve that, for instances of small tree width, dynPARTIX outperforms ASPARTIX. However, for AFs of high tree width the ASPARTIX system still performs better. Finally, our benchmarks show that using semi-normalized instead of normalized tree decompositions significantly improves the performance of dynPARTIX 2.0.

References

[1] Günther Charwat. Tree-Decomposition based Algorithms for Abstract Argumentation Frameworks. Master's thesis, Vienna University of Techology, 2012.

[2] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321 – 357, 1995.

[3] Wolfgang Dvořák, Reinhard Pichler, and Stefan Woltran. Towards fixed-parameter tractable algorithms for argumentation. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *KR 2010*, pages 112 – 122. AAAI Press, 2010.

[4] Wolfgang Dvořák, Michael Morak, Clemens Nopp, and Stefan Woltran. dynPARTIX - A dynamic programming reasoner for abstract argumentation. In *INAP 2011*, *CoRR*, abs/1108.4804, 2011.

[2] `http://www.dbai.tuwien.ac.at/research/project/argumentation/systempage/`

[3] `http://www.dbai.tuwien.ac.at/research/project/sharp/`

[4] `http://www.dlvsystem.com/`