ProGraph: towards enacting bipartite graphs for abstract argumentation frameworks

Serban Groza and Adrian Groza

Department of Computer Science
Technical University of Cluj-Napoca
Baritiu 28, 400391, Cluj-Napoca, Romania
EMAIL: Serban.Groza@student.utcluj.ro

Abstract. ProGraph was developed in Prolog and relies on bipartite graphs to partition the set of arguments in two classes: *in* and *out*. The current version of ProGraph is able to determine some extension and decide whether a given argument is credulously inferred, both with respect to the stable semantics.

1 Bipartite graphs

Bipartite graphs have been succesfully applied for several classes of problems (i.e., coverings, combinatorial applications, optimal spanning trees, general assignement problems) and within various domains (i.e, chemistry, communication networks, computer science) [1]. A graph G is bipartite is the vertex set $\mathcal{V}(G)$ can be partitioned into two sets V_1 and V_2 such that no vertices v_i from the same set are adjacent. The special case of bipartite argumentation frameworks admit polynomial time algorithms for preferred and stable semantics [2, 3].

2 Implementation details

The task to determine an extension which attacks every argument which is not in that extension can be reduced to a relaxed partitioning problem in which the initial set of arguments is split in two partitions: V_{in} and V_{out} with the arguments from the second partition being free to attack each other. Given the argumentation framework (A, R), we denote by $\{x\}^-$ the subset of A containing those arguments that attack argument x, and by $\{x\}^+$ the set of arguments from A that are attacked by x. The steps of the medhod are listed in algorithm 1.

Before the partitioning algorithm starts, the arguments are sorted such that they will be placed from the one who attacks the most to the one who attacks the less arguments. Consequently the first argument picked in each step of the partitioning algorithm is the one with the largest influence on the others. The algorithm picks a non-attacked argument y (line) adds y in the attackers extension (line) and then checks if any of the arguments attacked by y is in partition V_{in}. If this is the case, the algorithm starts backtracking. Otherwise, the arguments attacked by the current argument are added in V_{out} and the arguments attacked

Algorithm 1: Partitioning algorithm.

Input: (A, R) - argumentation framework;
Output: V_{in}, V_{out} - partition of A with *in* and *out* arguments;

```
1  V_in ← ∅, V_out ← ∅;
2  A' ← sort(A) s.t. ∀ y_i, y_j ∈ A' with i < j → |{y_i}^+| > |{y_j}^+|;
3  while ∃y ∈ A \ (V_in ∪ V_out) do
4      if ∃y ∈ A' s.t.{y}^- = ∅ then
5          select first y ∈ A'
6      else
7          select y ← first(A)
8      if {y}^+ ∩ V_in ≠ ∅ then
9          go to 4
10     else
11         V_in ← V_in + {y}
12         V_out ← V_out ∪ {y}^+
13     foreach a ∈ {y}^+ do
14         update({a}^+)
```

by them are updated in order to know how many possibly valid (i.e. members of A or V_{in}) arguments attack them.(lines 13-14). The steps are repeated until all arguments are partitioned or until all paths were tried and none succeeded.

If there are only attacked arguments left the algorithm will chose one of them and suppose it is not attacked (i.e. suppose its attacker will be placed in V_{out}). The mechanism that stops this from producing bad results is the verification step(lines 8-9), which stops the algorithm if at some point the attacker is to be placed in V_{out} .

3 Discussion and future work

ProGraph was developed as a semester project for the undergraduate level. We are currently investigating how metric properties and matrix characterisations of bipartite graphs can be exploited to develop heuristics for searching the preferred extensions [4] of an argumentation framework.

References

1. Asratian, A.S., Denley, T.M.J., Haggkvist, R.: Bipartite graphs and their applications. Volume 131. Cambridge University Press (1998)
2. Dunne, P.E.: The computational complexity of ideal semantics. Artificial Intelligence **173**(18) (2009) 1559–1591
3. Dunne, P.E.: Computational properties of argument systems satisfying graph-theoretic constraints. Artificial Intelligence **171**(10) (2007) 701–729
4. Nofal, S., Atkinson, K., Dunne, P.E.: Algorithms for decision problems in argument systems under preferred semantics. Artificial Intelligence **207** (2014) 23–51