Enumeration of Preferred Extensions in Almost Oriented Digraphs

Serge Gaspers
UNSW Sydney, Sydney, Australia
Data61, CSIRO, Australia
sergeg@cse.unsw.edu.au

Ray Li
UNSW Sydney, Sydney, Australia
rayli.main@gmail.com

Abstract

In this paper, we present enumeration algorithms to list all preferred extensions of an argumentation framework. This task is equivalent to enumerating all maximal semikernels of a directed graph. For directed graphs on n vertices, all preferred extensions can be enumerated in $O^*(3^{n/3})$ time and there are directed graphs with $\Omega(3^{n/3})$ preferred extensions. We give faster enumeration algorithms for directed graphs with at most $0.8004 \cdot n$ vertices occurring in 2-cycles. In particular, for oriented graphs (digraphs with no 2-cycles) one of our algorithms runs in time $O(1.2321^n)$, and we show that there are oriented graphs with $\Omega(3^{n/6}) > \Omega(1.2009^n)$ preferred extensions.

A combination of three algorithms leads to the fastest enumeration times for various proportions of the number of vertices in 2-cycles. The most innovative one is a new 2-stage sampling algorithm, combined with a new parameterized enumeration algorithm, analyzed with a combination of the recent monotone local search technique (STOC 2016) and an extension thereof (ICALP 2017).

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning; Theory of computation → Fixed parameter tractability; Mathematics of computing → Enumeration

Keywords and phrases abstract argumentation, exact algorithms, exponential time algorithms, parameterized algorithms, enumeration algorithms, semikernels in digraphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.74

Related Version `https://arxiv.org/abs/1907.01006`

Acknowledgements We thank Oliver Fisher for fruitful discussions and collaboration on preliminary results in the early stages of this work.

1 Introduction

In Dung's theory of abstract argumentation [15], an *argumentation framework* (AF) is a digraph $G = (V, E)$, where each vertex represents an argument, and an arc $(u, v) \in E$ denotes that argument u *attacks* argument v. There are various semantics that express what properties a set of arguments should have for a rational agent to stand by that set of arguments. One of the most central semantics is the *preferred semantics* that was already proposed by Dung in his foundational paper [15]. Let $S \subseteq V$ be a subset of vertices (also called *extension*) of a digraph $G = (V, E)$. The set S is *conflict-free* if no arc has both endpoints in S. A vertex $v \in V$ is *acceptable* with respect to S if for each arc $(u, v) \in E$ there is an arc $(w, u) \in E$ with $w \in S$. In other words, for each argument u that attacks v, there is an argument w in S that attacks u. We say in this case that w *defends* v against u. The set S is *admissible* if it is conflict-free and each argument in S is acceptable with respect to S. The set S is *preferred* if it is an inclusion-wise maximal admissible set.

While we will use the language of abstract argumentation, we remark that such vertex sets have also been studied in graph theory. Neumann-Lara [27] (see also [21]) defined the notion of *semikernels*. Maximal *semikernels* are equal to the preferred extensions in the directed graph where all arcs are reversed. The related notion of *kernels* [32] has the same correspondence with stable extensions in abstract argumentation, and was introduced as an abstract solution concept in cooperative game theory, but has been extensively studied in the theory of directed graphs. In particular, various issues around the enumeration of kernels and semikernels have been considered in previous work [2, 5, 20, 29].

Motivation. A central problem in abstract argumentation is the enumeration of extensions prescribed by a given semantics. The enumeration of preferred extensions is of particular interest, firstly for its own sake, but also in the study of other semantics as it forms the basis of several other semantics refining this set. A number of existing algorithms and implementations enumerate all preferred extensions of a digraph (see, e.g., [6, 7, 9, 10, 11, 12, 14, 25, 28, 30, 31]). The enumeration of preferred extensions is also a part of the biennial International Competition on Computational Models of Argumentation (ICCMA). Computational problems where the enumeration of extensions are used involve answering the questions: is a given argument in some / all preferred extensions and what is the number of preferred extensions containing a given argument / in total. Upper bounds on the number of extensions under various semantics have also been proposed as fundamental characteristics to compare various semantics in abstract argumentation [3, 16].

We study the enumeration of preferred extensions in digraphs with no, or relatively few, 2-cycles (i.e., bidirectional arcs). Our aim is to determine how much the presence of 2-cycles affects the number of preferred extensions of an AF. Mutually attacking arguments play a special role in abstract argumentation [24], but this conflict is often resolved rather easily if the strength of the two attacks can be evaluated [4], or the user's preference between the two arguments can be elicited [26, 1]. These methods of resolving conflicts motivate the study of problems, and in particular enumeration problems, for AFs with no or few 2-cycles.

Our results. Define the *resolution order* of a digraph $G = (V, E)$, denoted $r(G)$, as the number of vertices that belong to a 2-cycle in G. We study enumeration algorithms and combinatorial upper bounds on the number of preferred extensions in *oriented graphs* (digraphs without 2-cycles) and digraphs that have small resolution order.

Our main result is an algorithm that, for any $\varepsilon > 0$, enumerates all preferred extensions of a digraph G on n vertices in time

$$O^*\left(\left(\min\left(\varphi^{2r}\cdot\varphi^{1-r},\ \left(\left(1+2^{\frac{1}{4}}-\frac{1}{\sqrt{2}}\right)^{r}\cdot\left(2-\frac{1}{\sqrt{2}}\right)^{1-r}\right)^{1+\varepsilon},\ 3^{1/3}\right)\right)^{n}\right)$$
$$\leq O^*\left(\left(\min\left(1.5180^{r}\cdot 1.2321^{1-r},\ 1.4822^{r}\cdot 1.2929^{1-r},\ 1.4423\right)\right)^{n}\right),$$

where $r = r(G)/n$, and $\varphi \approx 1.2321$ is the positive root of $1 - x^{-1} - x^{-8}$. The O^* notation hides factors that are polynomial in the input size. See Figure 1, which plots the base α of the running time expressed as $O^*(\alpha^n)$ for r varying from 0 to 1. For $r = 1$, this is best possible and follows from the work in [16, 26]. At the other end of the spectrum, i.e., for oriented graphs where $r = 0$, the upper bound is $O^*(\varphi^n) \leq O(1.2321^n)$ and is obtained via a carefully constructed branching algorithm and running time analysis. We also give a lower bound on the largest number of preferred extensions an oriented graph on n vertices may have of $\Omega(3^{n/6}) \geq \Omega(1.2009^n)$. A construction, which we call the *Oriented Translation*, reducing an arbitrary digraph $G = (V, E)$ to an oriented graph with $|V| + r(G)$ vertices,

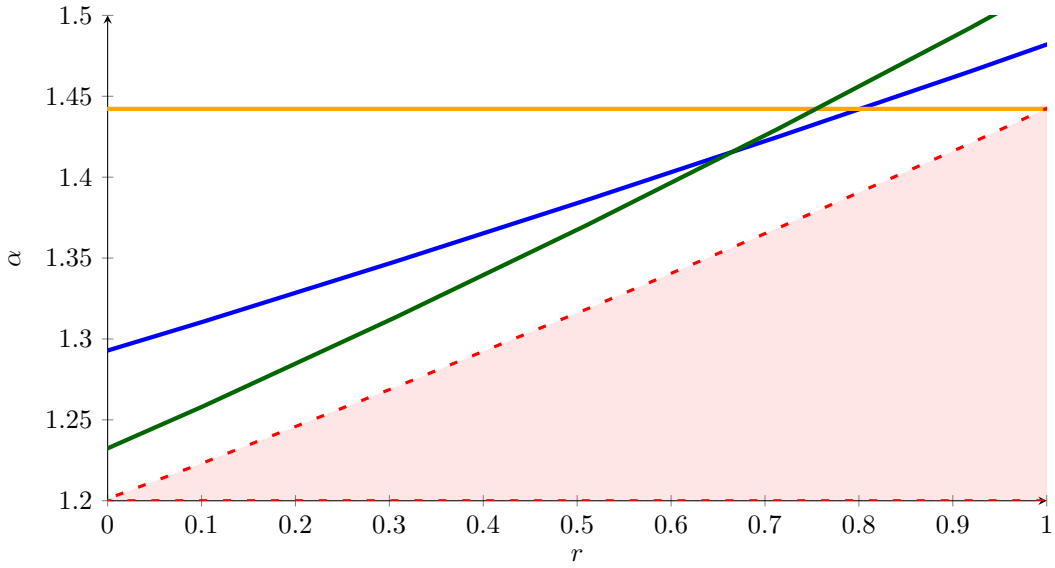

Figure 1 The graph depicts the base of the exponential running times $O^*(\alpha^n)$ of the three enumeration algorithms, according to $r = r(G)/n$. When $r < 0.6684$, our branching algorithm for oriented graphs together with the Oriented Translation (dark green) gives the fastest algorithm. For $r > 0.8005$, the algorithm based on previous work [16, 26] (orange) is fastest. In the middle range, the combination of the 2-phase monotone local search with the parameterized enumeration algorithm (blue) is fastest. Our lower bound on the largest number of preferred extensions is drawn with a dashed red line.

such that there is a bijection between their preferred extensions, allows us to generalize these upper and lower bounds to $O^*\left(\varphi^{2\cdot r(G)} \cdot \varphi^{n-r(G)}\right) \leq O(1.5180^{r(G)} \cdot 1.2321^{n-r(G)})$ and $\Omega(3^{r(G)/3} \cdot 3^{(n-r(G))/6}) \geq \Omega(1.4422^{r(G)} \cdot 1.2009^{n-r(G)})$, respectively.

Our main technical contribution is the third algorithm. It relies on a parameterized enumeration algorithm and extensions of the recent monotone local search framework [17]. The parameterized enumeration algorithm has as input a digraph $G = (V, E)$, a set of arguments S, and a non-negative integer k, and it enumerates all maximal admissible extensions $T \subseteq S$ of G within distance k of S. Its running time can be upper bounded by $O^*(2^{k/2+r(G[S])/4})$. This is optimal, since there are instances for which the solution consists of $\Omega(2^{k/2+r(G[S])/4})$ preferred extensions at distance at most k from S. Furthermore, under the Strong Exponential Time Hypothesis, the corresponding decision problem has no $O^*(2^{(1-\varepsilon)(k/2+r(G[S])/4)})$ time solution for any $\varepsilon > 0$. We use this parameterized enumeration algorithm in a new 2-phase monotone local search procedure, where we separately sample vertices from B, the set of vertices in at least one 2-cycle, and $V \setminus B$ and then apply the parameterized enumeration algorithm. The running time analysis is a new combination of the results in [17] for the first sampling phase and [23] for the second sampling phase, combined with the parameterized subroutine. From a technical point of view, this is the most innovative part of this paper. (From a conceptual point of view, the most innovative contribution is probably the synergy between modern enumeration algorithmics and the theory of abstract argumentation.) This results in an algorithm enumerating all preferred extensions of a given digraph G in time $O^*\left(\left(1 + 2^{1/4} - 2^{-1/2}\right)^{(1+\varepsilon)\cdot r(G)} \cdot \left(2 - 2^{-1/2}\right)^{(1+\varepsilon)\cdot(n-r(G))}\right)$.

Interpretation of results. Figure 1 indicates we have narrowed the gap between best known lower and upper bounds for a wide range of r and improved algorithms and combinatorial upper bounds whenever $r \leq 0.8004$. The result for $r = 0$ shows that for oriented graphs, our

new algorithm allows to handle instances with 75% more arguments, compared with the previous best $O(3^{n/3})$ upper bound. (We have that $\log_{1.2321}(3^{1/3}) \approx 1.7545$.)

Outline. Sections 3 and 4 describe our monotone local search algorithm. Section 5 describes our branching algorithm for oriented graphs.

First we introduce the parameterized enumeration problem that will form the subroutine of our Monotone Local Search.

Maximal Admissible Subset Enumeration (MASE)	
Input:	Graph G, set $S \subseteq V(G)$, integer k
Parameter:	k
Output:	Enumerate all maximal admissible sets $T \subseteq S$ such that $\lvert S \setminus T \rvert \leq k$.

There is a subtlety here with how we define maximal. We say T is a maximal admissible subset of S if there does not exist an admissible set U such that $T \subsetneq U \subseteq S$. Notably T is not necessarily a preferred extension (though T is if $S = V(G)$).

In Section 3, we present an algorithm for MASE, parameterized by k and $r(G[S])$, the resolution order of the subgraph of G induced by S.

▶ **Theorem 1.** *For an instance $I = (G, S, k)$ of MASE, let $\mu(I) := \frac{k}{2} + \frac{r(G[S])}{4}$. Then MASE can be solved in $O^*(2^{\mu(I)})$ time. Furthermore, there are at most $2^{\mu(I)}$ maximal admissible subsets of S within distance k of S. Hence, there are at most $2^{\mu(I)}$ preferred extensions that are subsets of S with size $\geq |S| - k$.*

Our algorithm is a standard parameterized branching algorithm. Compared to the enumeration of independent sets, the primary additional tool we have is a powerful simplification rule, **(Undefendable)**, for vertices with in-degree 0.

In Section 4, we extend our parameterized algorithm into a general enumeration algorithm through a novel 2-phase application of Monotone Local Search. Since 2-cycles increase the run time of our MASE subroutine, we modify Monotone Local Search to sample separately between a set of "bad vertices" (ones contained in a 2-cycle) and "good vertices" (ones not contained in any 2-cycle). This presents a speed up compared to a more direct application of the Monotone Local Search framework. We believe this may be useful for other problems.

Separately, Section 5 presents a branching algorithm for oriented graphs. Again, **(Undefendable)** plays a critical role. By carefully setting up the state we also have a simplification rule for vertices with out-degree 0. We tailor our branching rules to take full advantage of these two simplification rules. This is combined with a lot of careful case analysis and ad-hoc methods (including a graph classification theorem for Case 4).

Each of our algorithms also provide a corresponding combinatorial upper bound on the number of preferred extensions. The various enumeration algorithms and bounds are collected in Section 6. To extend the results from Section 5 to general graphs we require the following result which we call the *Oriented Translation*.

▶ **Theorem 2.** *There is a linear time algorithm that transforms any AF G into an oriented AF G' with $|V(G')| = |V(G)| + r(G) + 3$ such that there is a bijection between the preferred extensions of G and the preferred extensions of G' that can be applied in linear time.*

The basic idea of the construction is carefully converting 2-cycles into 4-cycles by duplicating vertices contained in at least one 2-cycle. The construction can be found in the full version.

In the full version we include all omitted proofs (including, in particular, the case analysis of our branching algorithms) and a few extra results:

We apply Theorem 2 to derive complexity results on oriented graphs by extending constructions for directed graphs. We show that:

- Unless P=NP, no algorithm enumerates the admissible or preferred extensions of an AF in output-polynomial time, even when the AF is an oriented graph.
- Assuming the Strong Exponential Time Hypothesis, our algorithm for the decision variant of MASE is optimal.
- The enumeration bound in Theorem 1 is tight (i.e: there exist instances with $2^{\mu(I)}$ maximal admissible subsets within distance k).

We also briefly justify the choice of the measure μ we use for our MASE algorithm by comparing it against similar measures.

Notation. An *oriented graph* is a digraph with no 2-cycles.

We will use the notation $N(v)$ to denote the set of vertices adjacent to v and $N[v]$ to denote $N(v) \cup \{v\}$.

We will assume throughout that AFs have no self loops. In the full version we present the Loopless Translation which, with a (small) constant overhead, transforms any AF into a loopless AF with the same lattice of admissible extensions (under inclusion).

2 Background on Analysis of Branching Algorithms

All results here can be found in standard textbooks. We mostly follow Fomin & Kratsch [19].

To analyze our branching algorithms we use Measure & Conquer [18]. A *measure* is a function assigning a non-negative number to each instance I.

We define the following standard terminology (e.g: see [19, 13]).

▶ **Definition 3** (Branching Vector and Number [19]). *Let μ be a measure. Let b be a branching rule that for any input instance, say I, branches into r instances with measures $\mu(I) - t_1, \mu(I) - t_2, \ldots, \mu(I) - t_r$ such that for all i, $t_i > 0$. Then we call $\mathbf{b} = (t_1, t_2, \ldots, t_r)$ the* branching vector *of branching rule b.*

Let α be the unique positive real root of $x^n - x^{n-t_1} - x^{n-t_2} - \ldots - x^{n-t_r}$.

We call α the branching number *of the branching vector $\mathbf{b}$.*

The following lemma from [22] forms the basis for the analysis of our branching algorithms.

▶ **Lemma 4** (Combine Analysis Lemma [22]). *Let A be an algorithm for a problem which for each instance, say I, either directly solves the instance in $O^*(\alpha_0^{\mu(I)})$ time or after polynomial time, applies one of r branching rules b_i, the i-th of which has branching number α_i. Then A has running time $O^*(\alpha^{\mu(I)})$ where $\alpha = \max_{0 \leq i \leq r}(\alpha_i)$.*

The following lemma forms the basis of our enumeration bounds. The search tree of a branching algorithm is the tree formed by the recursive calls, with the leaves being cases that can be solved directly. In all our applications the number of leaves is an upper bound on the number of objects being enumerated.

▶ **Lemma 5** (Combine Analysis Lemma for Enumeration). *Let A be an algorithm for a problem which for each instance, say I, either directly solves the instance with a branching algorithm that generates at most $\alpha_0^{\mu(I)}$ leaves or applies one of r branching rules b_i, the i-th of which has branching number α_i. Then the search tree for applying A to I has at most $\alpha^{\mu(I)}$ leaves where $\alpha = \max_i(\alpha_i)$.*

3 Parameterized Enumeration Problems

Our algorithm for MASE follows a standard template for parameterized branching algorithms (see Section 2 for notation). We will consider the measure $\mu(I) = \frac{k}{2} + \frac{b}{4}$ where k is the number of vertices we are allowed to remove and $b := r(G[S])$ is the resolution order of $G[S]$. We will design a branching algorithm with run time $O^*(2^{\mu(I)})$ which recurses into subinstances and collates their results to obtain the maximal admissible subsets of S.

However, there is a technical difficulty in the collation step arising from the fact that a maximal admissible subset of $S' \subsetneq S$ may not be a maximal admissible subset of S. This gives rise to the following subproblem:

Maximal Subset Collation	
Input:	Graph G, c pairs (S_i, C_i) where for each i, $S_i \subseteq V(G)$ and C_i is a set containing only maximal admissible subsets of S_i
Output:	A set containing all maximal elements of $\bigcup_{i=1}^{c} C_i$

The main result we need is:

▶ **Lemma 6.** *Maximal Subset Collation can be solved in $O(c \sum_{i=1}^{c} |C_i| \cdot \text{poly}(|V|))$ time.*

An algorithm for Maximal Subset Collation can be found in the full version. It is a consequence of Lemma 7.

3.1 An $O^*(2^{\mu(I)})$ algorithm for Maximal Admissible Subset Enumeration

The overall structure of our algorithm is described in Algorithm 1.

The following table lists the branching rules in application order with the first row being the base case. The second column describes the instances each rule is applicable to. After applying simplification rules, our branching algorithm will always apply the first rule that is applicable to the input instance.

Case	Requirement to apply	Running Time
Base	S is conflict-free.	Solves in $O^*(1)$ time, returns ≤ 1 set.
1	$G[S]$ is oriented with maximum total degree ≤ 2.	Branching vector $(1, 1)$, branching number 2.
2	There is a 2-cycle in $G[S]$.	Branching vector $(1, 1)$, branching number 2.
3	$G[S]$ has maximum total degree ≥ 4.	Branching vector $(2, \frac{1}{2})$, branching number ≈ 1.91.
4	$G[S]$ contains a vertex with total degree 3.	Branching vector $(1, \frac{3}{2})$, branching number ≈ 1.76.

We note that these requirements are exhaustive, hence there will always be at least one applicable rule in any instance.

The base case follows trivially from the following more general result:

▶ **Lemma 7.** *Suppose $S \subseteq V(G)$ induces a DAG in G. Then S has exactly one maximal admissible subset and it can be found in polynomial time.*

The main tool for this is the following Simplification Rule which we will use repeatedly:

▶ **Simplification Rule 1 (Undefendable).** *Let $u \in S$ be a vertex such that there exists a vertex $a \in V(G)$, a attacks u and no vertex in S attacks a. Then there is no admissible subset of S that contains u so we can safely set $S \leftarrow S \setminus \{u\}$.*

Algorithm 1 Structure of MASE branching algorithm.

Ensure: Returns all maximal admissible subsets $T \subseteq S$ such that $|S \setminus T| \leq k$
  function MASE(S, k)
    if $k < 0$ **then**
      return $\emptyset$
    while (Undefendable) applies **do**
      Apply **(Undefendable)**
    if Base Case applies **then**
      Solve the instance directly through the base case subroutine.
    else
      Let b_i be the first branching rule that applies.
      Let $(S_1, k_1) \dots (S_r, k_r)$ be the subinstances obtained from applying b_i.
      Let $C_i = \text{MASE}(S_i, k_i)$, for all $1 \leq i \leq r$.
      return Maximal_Subset_Collation$((S_1, C_1), \dots, (S_r, C_r))$

Lemma 7 follows from applying **(Undefendable)** to S until it is no longer applicable. Then S is acceptable with respect to S (all vertices in V attacking S are also attacked by S) and conflict-free (else **(Undefendable)** would be applicable to any vertex attacked by a maximal vertex in $G[S]$). Hence, by definition, S is admissible. It is the unique maximal admissible subset as **(Undefendable)** only removes vertices in no admissible subsets of S.

The above results are essentially all well known (see [14, 8] for earlier applications of **(Undefendable)**). The full version contains the above argument in more detail.

Proofs of the branching cases can be found in the full version. A few short remarks:

Case 1 follows from noting $G[S]$ must be a family of cycles.

Cases 2 and 3 follow from picking a specific vertex v then applying a 2-way branch, in one branch enumerating all maximal admissible subsets that include v, in the other branch enumerating all that do not include v. In case 2 we pick the vertex in the 2-cycle with higher total degree. In case 3 we pick any vertex with total degree at least 4.

Case 4 is probably the most instructive. It best showcases the power of **(Undefendable)**. First we show a vertex v exists with in-degree 1, out-degree 2. Then a 2-way branch is applied to the vertex attacking v, using **(Undefendable)** to improve the branch where the vertex is excluded and v is left with in-degree 0.

3.2 Running Time Analysis

The base case is solvable in polynomial time. Each of our branching rules has branching number ≤ 2. Hence, if we ignore the calls to Maximal Subset Collation, by the Combine Analysis Lemma our algorithm has running time $O^*(2^{\mu(I)})$.

Maximal Subset Collation is applied to each admissible subset encountered by the MASE algorithm (i.e: each leaf in the search tree) at most d times, where d is the maximum depth of the search tree. By Lemma 6 each application incurs a $O(\text{poly}(|V|))$ cost ($c \leq 2$ for our branching rules). Hence the overall cost incurred by the Maximal Subset Collation step is

$$O(A \cdot d \cdot \text{poly}(|V|))$$

where A is the total number of admissible subsets encountered by the MASE algorithm (equivalently, the number of leaves in the search tree). By the Combine Analysis Lemma for Enumeration, A is $O(2^{\mu(I)})$. The maximum depth d is $O(\mu(I))$ which we may take to be $O(|V|)$. Hence Maximal Subset Collation incurs an overhead cost of $O^*(2^{\mu(I)})$.

As noted, our algorithm also satisfies the requirements for applying the Combine Analysis Lemma for Enumeration. We summarize all the above results in the following theorem.

▶ **Theorem 8.** *Let $\mu(I) = \frac{k}{2} + \frac{b}{4}$, where b is the resolution order of $G[S]$. Then MASE can be solved in $O^*(2^{\mu(I)})$ time. Furthermore, there are at most $2^{\mu(I)}$ maximal admissible subsets of S within distance k of S. Hence, there are at most $2^{\mu(I)}$ preferred extensions that are subsets of S with size $\geq |S| - k$.*

4 Monotone Local Search

In this section, we apply Monotone Local Search to our $O^*(2^{\mu(I)})$ algorithm for Maximal Admissible Subset Enumeration. A basic exposition of Monotone Local Search will be provided here, see [17] for additional background. We adopt the notation of [17].

The framework normally applies to extension problems. However we can just as easily apply it to removal problems (formally by focusing on the complement of each set, we can turn any removal problem into an extension problem). Hence, we will freely use the framework with removal problems instead.

For our application, the instance I is the graph G and the family we are looking to enumerate, $\mathcal{F}_I$, is the set of all preferred extensions of G. We will apply Monotone Local Search using MASE as our subroutine.

For a MASE instance $I' = (G, X, k)$, let $\mathcal{F}^k_{I,X}$ denote the set of all maximal admissible sets $T \subseteq X$ such that $|X \setminus T| \leq k$. Hence $\mathcal{F}^k_{I,X}$ contains all preferred extensions that are subsets of X within distance k, but may also contain admissible extensions that are not preferred extensions.

Because of this, our Monotone Local Search will enumerate F_I, however, it may also enumerate some non-maximal admissible extensions. Our result, Theorem 10, will account for this, however, for simplicity we will just speak of enumerating F_I throughout this section.

A naive application of the Monotone Local Search framework with our $O^*\left(2^{\frac{k}{2}+\frac{b}{4}}\right)$ algorithm for enumerating $\mathcal{F}^k_{I,X}$ yields an $O^*\left(2^{\frac{b}{4}}\left(2-\frac{1}{\sqrt{2}}\right)^{n+o(n)}\right) \approx O^*(1.1893^b \cdot 1.2929^n)$ time algorithm that enumerates all preferred extensions of G. To improve this we need a basic understanding of how Monotone Local Search works.

4.1 Basic Overview of Monotone Local Search

We need the following definition from [17] (slightly modified to account for our preference for removal problems):

▶ **Definition 9** ([17])**.** *Let U be a universe of size n and let $0 \leq p \leq q \leq n$. A family $C \subseteq \binom{U}{q}$ is an (n, p, q)-set-containing-family if for every set $S \in \binom{U}{p}$, there exists a $Y \in C$ such that $S \subseteq Y$.*

For any fixed s, a value $t \geq s$ will somehow be chosen. Then, a (n, s, t)-set-containing-family C_s is constructed. For each set in C_s, its subsets from $\mathcal{F}_I$ obtained by removing at most $t - s$ elements are then enumerated using an enumeration subroutine. This enumerates all elements of $\mathcal{F}_I$ with size s. Supposing the subroutine has run time $O^*(\alpha^k)$ (where k is the parameter), this step of Monotone Local Search has run time $O^*(|C_s| \cdot \alpha^{t-s})$.

Repeating this for all s, Monotone Local Search has running time $O^*(\max_{1 \leq s \leq n} |C_s| \cdot \alpha^{t-s})$. With the right choices of t and C_s, [17] shows the running time $O^*((2 - \frac{1}{\alpha})^{n+o(n)})$.

Algorithm 2 Structure of Improved Monotone Local Search algorithm.

Ensure: Returns a set containing all preferred extensions of G (and possibly some non-preferred extensions)
function ImprovedMLS(G)
  Let $\mathcal{F}_I = \emptyset$.
  for $b = 1$ to $|B|$ **do**
    Let $b' =$ determine_b$'(b)$.
    Let C_b be a $(|B|, b, b')$-set-containing-family.
    for $d = 1$ to $|D|$ **do**
      Let $d' =$ determine_d$'(d)$.
      Let C_d be a $(|D|, d, d')$-set-containing-family.
      for all pairs (S, T) with $S \in C_b, T \in C_d$ **do**
        Let $\mathcal{F}_I = \mathcal{F}_I \cup \text{MASE}(S \cup T, (b' - b) + (d' - d))$.
  return $\mathcal{F}_I$

4.2 Improving our Monotone Local Search

We start with some notation. For any digraph $G = (V, E)$, let B be the set of vertices in V in at least one 2-cycle and let $D := V \setminus B$. The key idea is we will sample vertices from B and D separately. The overall structure is described in Algorithm 2. Except for the separate sampling, it is identical to a standard application of Monotone Local Search.

First, we argue correctness, i.e: that every preferred extension is enumerated at least once. Fix a preferred extension, say U, and suppose U contains b vertices in B and d vertices in D. Then, by the definition of set-containing-families, there exists a $S \in C_b$ such that $U \cap B \subseteq S$ and a $T \in C_d$ such that $U \setminus B \subseteq T$. Now we note that $S \subseteq B, T \subseteq V \setminus B$ to get:

$$|(S \sqcup T) \setminus U| = |S \setminus (U \cap B)| + |T \setminus (U \setminus B)| = (b' - b) + (d' - d)$$

Hence U is enumerated in the call to MASE$(S \cup T, (b' - b) + (d' - d))$ as required.

Now we argue the runtime. For a fixed b, d the calls to MASE have total run time $O^*(|C_b| \cdot |C_d| \cdot 2^{\frac{(b'-b)+(d'-d)}{2} + \frac{b'}{4}})$ for some choice of b', d'. Our overall running time is

$$O^* \left(\max_{0 \le b \le |B|} \max_{0 \le d \le |D|} |C_b| \cdot |C_d| \cdot 2^{\frac{(b'-b)+(d'-d)}{2} + \frac{b'}{4}} \right)$$

We can split this into two terms to get a complexity of $O^* \left(\max_{0 \le d \le |D|} |C_d| \cdot 2^{\frac{(d'-d)}{2}} \right)$ multiplied by $O^* \left(\max_{0 \le b \le |B|} |C_b| \cdot 2^{\frac{(b'-b)}{2} + \frac{b'}{4}} \right)$.

We will now analyze the running time with the right choices of b' and d'.

The first of these two terms can be analyzed using the analysis in [17]. This term is the complexity one attains for Monotone Local Search with a $O^*(2^{\frac{k}{2}})$ subroutine. Hence the analysis in [17] gives a complexity of $O^*((2 - \frac{1}{\sqrt{2}})^{|D| + o(|D|)})$.

We need the extended analysis presented in [23] to analyze the second term. This term is the complexity one attains for Monotone Local Search with a $O^*(2^{\frac{k}{2} + \frac{n - |X|}{4}})$ subroutine (where k is the parameter, $n = |U|$ the size of the underlying set and X is the set we are extending). Hence the analysis in [23] gives a complexity of $O^*((1 + 2^{\frac{1}{4}} - \frac{1}{\sqrt{2}})^{|B| + o(|B|)})$.

The papers we have cited also give a corresponding combinatorial upper bound on the number of preferred extensions. We summarize the above results as follows.

▶ **Theorem 10.** *Let $G = (V, E)$ be a digraph. Let r be the proportion of vertices in V that are in at least one 2-cycle.*

Then there exists a $O^*(((1+2^{\frac{1}{4}}-\frac{1}{\sqrt{2}})^r(2-\frac{1}{\sqrt{2}})^{1-r})^{|V|+o(|V|)}) \approx O^*((1.4822^r 1.2929^{1-r})^{|V|})$ time algorithm that enumerates all preferred extensions of G; however it may also enumerate some non-maximal admissible extensions. Furthermore, there are at most $O^*(((1+2^{\frac{1}{4}}-\frac{1}{\sqrt{2}})^r(2-\frac{1}{\sqrt{2}})^{1-r})^{|V|}) \approx O^*((1.4822^r 1.2929^{1-r})^{|V|})$ preferred extensions in G.

5 Improved Enumeration Algorithm for Oriented Graphs

Finally, we outline a branching algorithm with a finer analysis for oriented graphs. As with MASE, we will follow a standard template for branching algorithms. A summary of the necessary concepts can be found in Section 2. Our algorithm creates a search tree with at most φ^n leaves where $\varphi \approx 1.2321$ is the branching number for branching vector $(8, 1)$.

In Section 6 we will use the Oriented Translation (Theorem 2) to obtain a general enumeration algorithm parameterized by the number of vertices in at least one 2-cycle.

5.1 Overview

The overall structure of our algorithm is described in Algorithm 3.

The state of the algorithm consists of the subset Und $\subseteq V(G)$ and a queue of vertices Def (Und for undecided and Def for deferred). Und is the set of vertices we have yet to make a decision on whether we should include them. Def is a queue of vertices which have no outgoing arcs to Und. These vertices will be handled in the base case. While branching we can essentially assume the vertices in Def do not exist.

We maintain the following invariants. We outline why they are invariant, it is straightforward to verify that each case of our branching algorithm maintains these invariants.

1. Und and Def are disjoint. This holds as vertices are only ever deleted from or moved between Und and Def, never copied.
2. G[Def] is a DAG where each vertex $v \in$ Def only attacks vertices that were added to Def before v. This is crucial for the base case since a DAG has 1 maximal admissible subset. This holds as only vertices with out-degree 0 in G[Und] are ever moved to Def.
3. There is no attack from any vertex in Def to any vertex in Und. This holds for the same reason as Invariant 2.

Our measure is $\mu = |\text{Und}|$. For any instance, our algorithm creates a search tree with at most φ^μ leaves. Hence, calling OrientedEnumeration$(V(G), [])$ will return all preferred extensions of G by traversing a search tree with at most $\varphi^{|V(G)|}$ leaves.

5.2 Extra Notation

We will say a vertex has degree $(a, -)$ if it has in-degree a, a vertex has degree $(-, b)$ if it has out-degree b and a vertex has degree (a, b) if it has in-degree a and out-degree b.

5.3 Simplification Rules

Both of these are applicable in polynomial time and decrease $\mu = |\text{Und}|$.

▶ **Simplification Rule 2** (Out-degree 0). *Let v be a vertex in G*[Und] *with out-degree* 0. *Move v from* Und *to the end of the queue* Def.

Algorithm 3 Structure of Oriented maximal admissible enumeration algorithm.

Require: $\text{Und} \cap \text{Def} = \emptyset$.
Require: $G[\text{Def}]$ is a DAG where each $v \in \text{Def}$ only attacks vertices added to Def before v.
Require: There is no attack from any vertex in Def to any vertex in Und.
Ensure: Returns all maximal admissible subsets of $\text{Und} \cup \text{Def}$.

```
function OrientedEnumeration(Und, Def)
    while Any simplification rule applies do
        Apply said simplification rule
    if Base Case applies then
        Solve the instance directly through the base case subroutine.
    else
        Let b_i be the first branching rule that applies.
        Let (U_1, D_1) ... (U_r, D_r) be the subinstances obtained from applying b_i.
        Let C_i = OrientedEnumeration(U_i, D_i), for all 1 ≤ i ≤ r.
        return Maximal_Subset_Collation((U_1 ∪ D_1, C_1), ..., (U_r ∪ D_r, C_r))
```

▶ **Simplification Rule 3** (In-degree 0). *Let v be any vertex in $G[\text{Und}]$ with in-degree 0. Then by Invariant 3, v has in-degree 0 in $G[\text{Und} \cup \text{Def}]$. Applying Simplification Rule **(Undefendable)** we can set $\text{Und} \leftarrow \text{Und} \setminus N(v)$. After that, v has out-degree 0 in $G[\text{Und}]$ and hence we move v from Und to Def.*

Our new instance $I' = (\text{Und}', \text{Def}')$ has:

- $\text{Und}' = \text{Und} \setminus N[v]$.
- $\text{Def}' = \text{Def} \cup \{v\}$.

Due to these rules, henceforth we may assume each vertex in $G[\text{Und}]$ has in-degree ≥ 1, out-degree ≥ 1 and (total) degree ≥ 2.

5.4 Branching Rules

Our algorithm will apply the first rule applicable to the instance:

Case	Requirement to apply	Worst case branching number
Base	$\text{Und} = \emptyset$.	Solves in $O^*(1)$, returns 1 set.
1	$\exists v \in G[\text{Und}]$ with total degree ≥ 7.	Branching vector $(8, 1)$, branching number $\varphi \approx 1.2321$.
2	$\exists v \in G[\text{Und}]$ with degree $(1, -)$.	Branching vector $(4, 3)$, branching number ≈ 1.221.
3	$\exists v \in G[\text{Und}]$ with in-degree $\neq$ out-degree.	Branching vector $(6, 5, 5)$, branching number ≈ 1.2298.
4	$G[\text{Und}]$ has a weakly connected component where every vertex has degree $(2, 2)$.	Branching vector $(6, 5, 5)$, branching number ≈ 1.2298.
5	$G[\text{Und}]$ has a weakly connected component where every vertex has degree $(3, 3)$.	Branching vector $(7, 7, 7, 7)$, branching number ≈ 1.219.
6	There is a weakly connected component in $G[\text{Und}]$ where every vertex has in-degree = out-degree.	Branching vector $(7, 5, 5)$, branching number ≈ 1.218.

We note the base case and cases 3 and 6 cover all possible inputs. Hence there will always be at least one applicable rule.

Our primary strategy is to pick a specific vertex v and do a 2-way branch, separately enumerating the maximal admissible subsets that include v and the ones that exclude v.

- In the branch where we include v we must exclude v's neighbors. Hence we create a new instance $I' = (\text{Und}', \text{Def})$ where $\text{Und}' = \text{Und} \setminus N(v)$.
  Now v is isolated in $G[\text{Und}']$ so by Simplification Rule 2 we may move v from Und' to Def'. Hence in our new instance we finally have:
  - $\text{Und}' = \text{Und} \setminus N[v]$.
  - $\text{Def}' = \text{Def} \cup \{v\}$.

  and hence $\mu(I') = \mu(I) - |N[v]| = \mu(I) - \deg(v) - 1$.
  As a technical note, not every subset enumerated in this branch contains v, however, every maximal admissible subset that contains v will be enumerated in this branch.
- In the branch where we exclude v our new instance $I' = (\text{Und}', \text{Def}')$ is:
  - $\text{Und}' = \text{Und} \setminus \{v\}$.
  - $\text{Def}' = \text{Def}$.

  and hence $\mu(I') = \mu(I) - 1$.

Hence our branching vector is $(\deg(v) + 1, 1)$. However, often our choice of v allows us to immediately apply our simplification rules to improve the branch where v is excluded.

Full proofs of each case can be found in the full version. A few short remarks:

In the base case, by Invariant 2, $G[\text{Und} \cup \text{Def}]$ is a DAG. The base case then follows trivially from Lemma 7.

Case 2 is a good example of the strategy of picking a specific v to apply a 2-way branch on. We will pick a v that allows us to apply our simplification rules in the branch where v is excluded. However, the choice of v will depend on some case analysis on degrees of vertices.

Case 3 is similar to Case 2, a specific v is chosen on which we apply a 2-way branch.

Case 4 is done through a graph classification theorem. We classify the family of oriented graphs where each vertex has in-degree 2, out-degree 2 and both in-neighbors adjacent (if there is a vertex with independent in-neighbors then we instead apply a 3-way branch on that vertex and its in-neighbors).

Case 5 is just a 4-way branch on any vertex v and its 3 in-neighbors.

For Case 6, we first show there is a vertex with degree $(3, 3)$ attacking a vertex with degree $(2, 2)$, say b. Then we apply a 3-way branch on b and its 2 in-neighbors.

5.5 Summary of results

In our base case we enumerate 1 extension in polynomial time.

Otherwise, we apply a branching rule with branching number $\leq \varphi$.

As in MASE, the Maximal Subset Collation subroutine does not incur additional overhead as the search tree's depth is bounded by $|V(G)|$ (see Subsection 3.2 for the argument which we can apply verbatim).

Applying the Combine Analysis Lemma and Combine Analysis Lemma for Enumeration we obtain:

▶ **Theorem 11.** *Let $G = (V, E)$ be an oriented graph. Then there is an algorithm that enumerates all preferred extensions of G with running time $O^*(\varphi^{|V|})$ where φ is the unique positive root of $1 - x^{-1} - x^{-8} = 0$, $\varphi \approx 1.23205 < 1.2321$.*

Furthermore, G has at most $\varphi^{|V|}$ preferred extensions.

6 Bounds on number of preferred extensions

In this section, we collect our results bounding the number of preferred extensions.

6.1 Bounds on general directed graphs

A tight upper bound is $O(3^{\frac{|V|}{3}})$ [16]. This bound is easily attainable since preferred extensions coincide with maximal independent sets(MIS) in graphs where every edge is in a 2-cycle.

We can also enumerate them in $O^*(3^{\frac{|V|}{3}})$. We note each MIS has a single maximal admissible subset. We then take a branching algorithm for MIS[22], allowing us to apply the Maximal Subset Collation subroutine to remove the non preferred extensions without additional overhead. Hence, all preferred extensions can be enumerated in $O^*(3^{\frac{|V|}{3}})$.

6.2 Parameterizing by Resolution Order

We will give bounds based on r, the proportion of vertices that are in at least one 2-cycle.

6.2.1 Lower Bound

An undirected 3-cycle has 3 preferred extensions. Hence, applying the Oriented Translation to it, we obtain an oriented structure with 6 vertices and 3 preferred extensions.

Our construction for lower bounding will be to include as many undirected 3-cycles as possible and then include as many oriented translations of 3-cycles as possible. We can include $\frac{r|V|}{3}$ undirected 3-cycles and $\frac{(1-r)|V|}{6}$ oriented translations, obtaining an AF with $\Omega((3^{\frac{r}{3}}3^{\frac{1-r}{6}})^{|V|}) \approx ((1.44^r 1.2^{1-r})^{|V|})$ preferred extensions.

6.2.2 Upper Bound

There are 3 different upper bounds that are all optimal in a different range. See also Figure 1.

- $(\varphi^{2r}\varphi^{1-r})^{|V|} \approx O((1.5180^r 1.2321^{1-r})^{|V|})$ where φ is the unique positive root of $1-x^{-1}-x^{-8}$. This bound is obtained from using the Oriented Translation along with Theorem 11. This is best for r up to around 0.6684.
- $O^*(((1+2^{\frac{1}{4}}-\frac{1}{\sqrt{2}})^r(2-\frac{1}{\sqrt{2}})^{1-r})^{|V|}) \approx O^*((1.4822^r 1.2929^{1-r})^{|V|})$, the bound from our 2-phase Monotone Local Search. This is best for a small range where $0.6685 \leq r \leq 0.8004$.
- $3^{\frac{|V|}{3}}$. This is best for $r \geq 0.8005$.

7 Conclusion

We again note that the concept of an admissible (resp. preferred) extension has also been studied as a *semikernel*[27] (resp. maximal *semikernel*) in graph theory. Hence our result may be interpreted as a combinatorial upper bound on the number of maximal *semikernels*, parameterized by the proportion of vertices in at least one 2-cycle.

References

1 Leila Amgoud and Claudette Cayrol. A Reasoning Model Based on the Production of Acceptable Arguments. *Annals of Mathematics and Artificial Intelligence*, 34(1-3):197–215, 2002. doi:10.1023/A:1014490210693.

2 Cyril Banderier, Jean-Marie Le Bars, and Vlady Ravelomanana. Generating functions for kernels of digraphs (Enumeration & asymptotics for a constraint from game theory). In *Proceedings of the 16th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2004)*, pages 91–105, 2004.

3 Ringo Baumann and Hannes Strass. Open Problems in Abstract Argumentation. In *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, volume 9060 of *Lecture Notes in Computer Science*, pages 325–339. Springer, 2015.

4 Trevor J. M. Bench-Capon. Value Based Argumentation Frameworks. In *Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR 2002)*, volume cs.AI/0207059, pages 443–454, 2002. URL: http://arxiv.org/abs/cs.AI/0207059.

5 Raymond Bisdorff. On enumerating the kernels in a bipolar-valued outranking digraph. Technical Report 6, Annales du Lamsade, 2006. hal-00118995.

6 Stefano Bistarelli, Fabio Rossi, and Francesco Santini. A Comparative Test on the Enumeration of Extensions in Abstract Argumentation. *Fundamenta Informaticae*, 140(3-4):263–278, 2015.

7 Martin Caminada. An Algorithm for Computing Semi-stable Semantics. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2007)*, volume 4724 of *Lecture Notes in Computer Science*, pages 222–234. Springer, 2007.

8 Martin Caminada. An Algorithm for Computing Semi-stable Semantics. In *ECSQARU 2007: Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 4724 of *Lecture Notes in Computer Science*, pages 222–234. Springer, Berlin, Heidelberg, 2007.

9 Federico Cerutti, Paul E. Dunne, Massimiliano Giacomin, and Mauro Vallati. Computing Preferred Extensions in Abstract Argumentation: A SAT-Based Approach. In *Proceedings of the 2nd International Workshop on Theory and Applications of Formal Argumentation (TAFA 2013)*, volume 8306 of *Lecture Notes in Computer Science*, pages 176–193. Springer, 2013.

10 Federico Cerutti, Massimiliano Giacomin, and Mauro Vallati. Algorithm Selection for Preferred Extensions Enumeration. In *Proceedings of the 5th International Conference on Computational Models of Argument (COMMA 2014)*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 221–232. IOS Press, 2014.

11 Federico Cerutti, Mauro Vallati, and Massimiliano Giacomin. On the impact of configuration on abstract argumentation automated reasoning. *International Journal of Approximate Reasoning*, 92:120–138, 2018.

12 Günther Charwat, Wolfgang Dvorák, Sarah Alice Gaggl, Johannes Peter Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation - A survey. *Artificial Intelligence*, 220:28–63, 2015.

13 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

14 Sylvie Doutre and Jérôme Mengin. Preferred Extensions of Argumentation Frameworks: Query, Answering, and Computation. In *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Computer Science*, pages 272–288. Springer, 2001.

15 Phan Minh Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

16 Paul E. Dunne, Wolfgang Dvorák, Thomas Linsbichler, and Stefan Woltran. Characteristics of multiple viewpoints in abstract argumentation. *Artificial Intelligence*, 228:153–178, 2015.

17 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2016)*, pages 764–775. ACM, 2016.

18 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM*, 56(5):25:1–25:32, 2009.

19 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer, 2010.

20 Hortensia Galeana-Sánchez and Xueliang Li. Semikernels and (k, l)-Kernels in Digraphs. *SIAM Journal on Discrete Mathematics*, 11(2):340–346, 1998.

21 Hortensia Galeana-Sánchez and Victor Neumann-Lara. On kernels and semikernels of digraphs. *Discrete Mathematics*, 48(1):67–76, 1984.

22 Serge Gaspers. *Exponential time algorithms: Structures, measures, and bounds*. PhD thesis, University of Bergen, 2008.

23 Serge Gaspers and Edward J. Lee. Exact Algorithms via Multivariate Subroutines. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 69:1–69:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.69.

24 Sanjay Modgil. Hierarchical Argumentation. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA 2006)*, pages 319–332, 2006. doi:10.1007/11853886_27.

25 Sanjay Modgil and Martin Caminada. Proof Theories and Algorithms for Abstract Argumentation Frameworks. In *Argumentation in Artificial Intelligence*, pages 105–129. Springer, 2009.

26 Sanjay Modgil and Henry Prakken. Resolutions in Structured Argumentation. In *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 310–321. IOS Press, 2012.

27 Victor Neumann-Lara. Seminúcleos de una digráfica. Technical report, Anales del Instituto de Matemáticas II, Universidad Nacional Autónoma México, 1971.

28 Samer Nofal, Katie Atkinson, and Paul E. Dunne. Algorithms for decision problems in argument systems under preferred semantics. *Artificial Intelligence*, 207:23–51, 2014.

29 Jayme Luiz Szwarcfiter and Guy Chaty. Enumerating the Kernels of a Directed Graph with no Odd Circuits. *Information Processing Letters*, 51(3):149–153, 1994.

30 Mauro Vallati, Federico Cerutti, and Massimiliano Giacomin. Argumentation Extensions Enumeration as a Constraint Satisfaction Problem: a Performance Overview. In *Proceedings of the International Workshop on Defeasible and Ampliative Reasoning (DARe@ECAI 2014)*, volume 1212 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

31 Mauro Vallati, Federico Cerutti, and Massimiliano Giacomin. Argumentation Frameworks Features: an Initial Study. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 1117–1118. IOS Press, 2014.

32 John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.