Computing Argumentation with Matrices

Evgenios Hadjisoteriou

Department of Computer Science, University of Cyprus
75 Kallipoleos Str., 1678 Nicosia, Cyprus
csp7he2@cs.ucy.ac.cy

Abstract

Abstract argumentation frameworks with finitely many arguments can be presented in matrix form. For this reason, the strengths and weaknesses of matrix operations are migrated from a mathematical representation to a computer science interpretation. We present matrix operation algorithms that can answer whether a given set of arguments is part of an argumentation extension.

1998 ACM Subject Classification I.2.3 Answer/reason extraction

Keywords and phrases Argumentation, Semantics, Extension, Algorithm, Matrix

Digital Object Identifier 10.4230/OASIcs.ICCSW.2015.29

1 Introduction and Motivation

Argumentation theory tries to mimic the process of human reasoning. It is often used by agents to reason under specific knowledge, often incomplete, with the alternative to choose. Agents can construct arguments for and against a specific goal, in order to reach a conclusion. The construction of these arguments often follows a semantics, that is given in an abstract argumentation framework either by extensions [5] or by labellings [2, 14]. For what follows we use the former. Many semantics have been established, such as grounded (yields exactly one unique extension), complete, preferred, stable and all of them come with interesting properties [5, 1, 6]. Nowadays, agents perform tasks under incomplete information and dynamic environments, thus decisions must be precise and easily computable. Agents need a tool that is able to produce extensions under the working environment to help them decide what their next move should be.

Finding extensions can be a complex procedure when done without any computational help, when the argumentation framework contains several arguments and attacks. We have developed an algorithm that answers whether a given set of arguments is an extension. Part of this algorithm has been implemented in a system and presented at the ICCMA'15 [1] competition. Our solver, called $\mathcal{ASSA}$ [2] finds the stable extension(s) of an argumentation framework.

In Section 2 we give some basic notions and in Section 3 we present the algorithm tests we perform. A comparison to related work is done in Section 4, and Section 5 concludes with future work.

[1] http://argumentationcompetition.org/index.html
[2] The name was inspired from left and right matrix multiplication: $\mathcal{AS}$ and $\mathcal{SA}$

2015 Imperial College Computing Student Workshop (ICCSW 2015).
Editors: Claudia Schulz and Daniel Liew; pp. 29–36
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
ICCSW 15

2 Matrices & Argumentation Frameworks

In this section, we present the basic information on matrices and argumentation extensions. We assume that the reader is familiar with basic matrix tools and the fundamentals on argumentation frameworks (see e.g. [11, 5, 1, 6] for more details). A link between the fields of graph theory and logic programs has been presented in [4, 13], where it is shown that stable extensions corresponds to the kernels of the adjacency matrix.

2.1 Matrices

A **matrix** is a structure in rows and columns, where each one of its elements contains information. When the number of rows and columns are equal, the matrix is called **square matrix**. Square matrices have diagonals. A row vector is a $1 \times n$ matrix $\begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}$ and a column vector is a $n \times 1$ matrix $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$, often written as the transpose matrix $\begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}^T$.

Computers can perform matrix operations in a relatively fast way. The computational complexity for multiplying two matrices with n digit numbers is $\mathcal{O}(n^3)$ [12]. There are methods that can optimize this result [12, 10]. Our algorithm for computing argumentation extensions and verifying if something is an extension is based on matrix multiplication, thus it is of logarithmic space complexity [8]. In this initial paper, we will not be concerned with complexity issues. It is based on matrices because (a) matrices can be easily represented and handled by a computer, (b) illustrated information is compact, (c) through different operation tools matrices can be easily manipulated and (d) matrices can capture all the information of an abstract argumentation framework.

2.2 Argumentation Frameworks

We review some of the notions introduced by Dung [5] such as acceptable, conflict-free, complete extensions.

▶ **Definition 1** (argumentation framework)**.** An argumentation framework is a pair $\langle \mathcal{A}, \mathcal{R} \rangle$, where $\mathcal{A}$ is a set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is a binary relation on $\mathcal{A}$ called attack relation.

Square matrices can capture the arguments and the attacking relations of an argumentation framework in a relatively easy way. We first need to label each argument with a distinct natural number and use the rows and columns of the matrix to represent the arguments and the attacks respectively. For example, the third row $a_{3,*}$ of a square matrix consists of the elements $\{a_{3,1}, a_{3,2}, \ldots, a_{3,n}\}$. If argument a_3 attacks argument a_4 then the element $a_{3,4}$ of the square matrix will be one (1), otherwise it will be zero (0). The value of the element $a_{3,3}$ will show if argument a_3 is self attacking.

▶ **Definition 2** (mapping an argumentation framework to a matrix)**.** Let $A = (a_{i,j})$ be the **adjacency matrix** of an argumentation framework $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ such that: $a_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{R} \\ 0 & \text{if } (i,j) \notin \mathcal{R} \end{cases}$

It is important to know who attacks who. The attacker is represented by the row of the adjacency matrix and each column represents the attacked argument. Therefore, $a_{3,4}$ represents the attack from argument a_3 to argument a_4 while $a_{4,3}$ represents the attack from a_4 to a_3.

(a) **(b)**

Figure 1 (a) and (b) show respectively the directed graph and matrix representation of Example 3.

▶ **Example 3.** Let $\{a, b, c\} = \mathcal{A}$ be three arguments such that $\{(a, b), (b, c)\} = \mathcal{R}$. Figure 1 depicts this example.

Matrix operations exist, i.e. multiplication, where when performed on the adjacency matrix of an argumentation framework, an interpretation exists connecting the operating result with the argumentation framework.

▶ **Definition 4** (representing a set of arguments as a column vector)**.** Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework with A its adjacency matrix and $S \subseteq \mathcal{A}$. Set S is represented by a column vector $\mathcal{S}_{n\times 1} = (s_{i,1})$, where $s_{i,1} = \begin{cases} 1 & \text{if } a_i \in S \\ 0 & \text{if } a_i \notin S \end{cases}$

▶ **Proposition 5.** *Let A be the adjacency matrix of an argumentation framework $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ and let $S \subseteq \mathcal{A}$ be a set of arguments with $\mathcal{S}$ its column (resp. $\mathcal{S}^T$ its row) vector representation. The product $A\mathcal{S}$ is a column (resp. $\mathcal{S}^T A$ is a row) vector where the entry $(A\mathcal{S})_{i,1}$ (resp. $(\mathcal{S}^T A)_{1,i}$) shows how many times argument $a_i \in \mathcal{A}$ attacks (resp. is attacked by) S.*

Proof. Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework with n arguments and $A = (a_{i,j})$ its adjacency matrix. Let $S \subseteq \mathcal{A}$ be a set of arguments with $\mathcal{S}$ its column vector representation. The product (of the two matrices A and $\mathcal{S}$) $A\mathcal{S}$ is defined as follows: $(A\mathcal{S})_{i,1} = \sum_{t=1}^{n}((i,t)_{th}$ element of $A \times (t,1)_{th}$ element of $\mathcal{S}) = \sum_{t=1}^{n} a_{i,t}s_{t,1}$. Based on Definition 4 and Definition 2, $(A\mathcal{S})_{i,1}$ is an addition of zeroes if at least one of the entries $a_{i,t}$ or $s_{t,1}$ is zero as $0 \times 1 = 1 \times 0 = 0 \times 0 = 0$ or an addition of ones if both entries $a_{i,t} = s_{t,1} = 1$ since $1 \times 1 = 1$. Intuitively, it is an addition of ones if and only if there exists an attack from a_i to a_t in AF and $a_t \in S$. Similar results hold for $\mathcal{S}^T A$. ◀

3 Algorithms

In this section, we give a computerized method under which given a set of arguments, we can answer whether this set of arguments is conflict-free, admissible, stable, or complete.

3.1 Conflict-free test

Given a set of arguments we can check if this set is conflict-free by running a conflict-free test as follows.

▶ **Definition 6** (conflict-free)**.** A set of arguments S is said to be conflict-free if there are no arguments $a, b \in S$ such that $(a, b) \in \mathcal{R}$.

▶ **Proposition 7** (conflict-free test)**.** *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and A its adjacency matrix. Let $S \subseteq \mathcal{A}$ be a given set of arguments with $\mathcal{S}$ its column vector representation. Let $\Gamma = \mathcal{S}^T A$. $\mathcal{S}$ passes the conflict-free test if and only if whenever $\gamma_i \neq 0 \in \Gamma$ then $s_i = 0 \in \mathcal{S}$.*

Proof. Based on Proposition 5, $\gamma_i \in \Gamma$ shows how many times argument $a_i \in \mathcal{A}$ is attacked by $\mathcal{S}$. Therefore, when $\gamma_i \neq 0 \in \Gamma$ means that argument a_i is attacked in γ_i ways by the arguments in S. This does not conform with Definition 6. To pass the test arguments that are attacked should not be part of $\mathcal{S}$, thus $s_i = 0 \in \mathcal{S}$. ◀

By constructing a matrix multiplication we can answer if a given set of arguments S is conflict-free. When a row matrix passes (resp. fails) the test we conclude that S is (resp. is not) conflict-free. Note that the empty set always passes the conflict-free test as the generated matrix Γ has zeroes everywhere.

▶ **Example 8.** (a) Consider Example 3 illustrated in Figure 1. Let $S_1 = \{a_1\}$ with $\mathcal{S} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ and $\mathcal{S}^T A = \Gamma = (\,0\;1\;0\,)$. $\gamma_i \neq 0$ when $i = 2$ and $s_2 = 0$. For this reason it passes the conflict-free test. Therefore, the set $S_1 = \{a_1\}$ is conflict-free, i.e. not self attacking. Let us now consider the set $S_2 = \{a_1, a_2\}$ with $\mathcal{S} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ and $\mathcal{S}^T A = \Gamma = (\,0\;1\;1\,)$. $\gamma_2 = s_2 = 1$, thus the set S_2 fails the conflict-free test.

3.2 Admissibility test

To check the admissibility of a given set of arguments S we perform two tests: (a) the conflict-free test and (b) the defending test. For S to be admissible it has to pass both tests.

▶ **Definition 9** (defendable)**.** An argument a is defendable with respect to a set S if and only if each argument attacking a is attacked by an argument in S.

▶ **Proposition 10** (defending test)**.** *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework with adjacency matrix A and let $S \subseteq \mathcal{A}$ be a set of arguments. Let $\mathcal{S}$ be the column vector representation of S and $\Gamma = (\gamma_i) = A\mathcal{S}$. For every i that $\gamma_i \neq 0$ create a column vector $\Delta^{(i)} = (\delta_j^{(i)})$, such that: $\delta_j^{(i)} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$. Set S passes the defending test if and only if at least one of the following holds: (1) Γ is a zero matrix. (2) $E^{(i)} = A\Delta^{(i)}$ and $\forall i, \exists e_k^{(i)} \in E^{(i)}$ such that $e_k^{(i)} \neq 0$ and $0 \neq s_k \in \mathcal{S}$.*

Proof. Based on Definition 9. Matrix $\Gamma = (\gamma_i) = A\mathcal{S}$ shows if S is under attack. When (a) $\Gamma = \mathbf{0}$, S passes the defending test as no attackers exist. When (b) $\Gamma \neq \mathbf{0}$, i.e. attackers exist, we check if S counter attacks them. $\gamma_i \neq 0$ shows the number of attacks but the information who attacks who is lost through the algebraic operations of matrix multiplication. To retrieve this critical information we create $\Delta^{(i)}, \forall i$ that $\gamma_i \neq 0$ and evaluate $E^{(i)} = A\Delta^{(i)}$. $E^{(i)}$ shows if any arguments in A attack $\Delta^{(i)}$. To ensure that when counter attacks exist, they come from arguments in S, we add the restriction $\forall i, \exists e_k^{(i)} \in E^{(i)}$ when $e_k^{(i)} \neq 0$, $s_k \neq 0$, i.e. for every attack there exists at least one argument counter attacking it and this argument belongs to S. ◀

The defending test is based on matrix multiplication. $\Gamma = A\mathcal{S}$ shows if S is under attack, i.e. if any arguments that exist in our argumentation framework can attack S. When $\Gamma = \mathbf{0}$, set S passes the defending test as no attackers exist and there are no arguments to defend against. When $\Gamma \neq \mathbf{0}$, set S is under attack and we have to check if S counter attacks them. Because we do not know who attacks who, instead we get an indication of the number of times each argument in the AF attacks S, we need to separate the arguments under attack and create the vectors $\Delta^{(i)}$. Based on another matrix multiplication, $E^{(i)} = A\Delta^{(i)}$, we find $E^{(i)}$ that shows any arguments in A that attack $\Delta^{(i)}$. To make sure that when these counter

attacks exist, they come from arguments in S, we compare $\mathcal{S}$ with $E^{(i)}, \forall i$. If the comparison shows that any attack is counter attacked by S, we conclude that set S has passed the defending test.

▶ **Example 11** (defending test examples). In Example 3 consider the sets $S^{'} = \{\}$ and $S = \{a, c\}$.

For $S^{'}$: all entries for its column vector representation are zeroes and since $S^{'}$ is represented by a zero matrix, the empty set passes the defending test.

For S: $\mathcal{S} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ and $A\mathcal{S} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \Gamma$. Using $\mathcal{S} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$, we are interested in which arguments can attack arguments a and c. The answer is $\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \Gamma$ which means that argument b attacks the argument set $S = \{a, c\}$. From Γ and its non zero entries we find $\Delta^{(1)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$. To check if all attackers (i.e. b) are attacked back by S we evaluate $E^{(1)} = A\Delta^{(1)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$. The result is $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ and since $e_1^{(1)} \neq 0 \in E^{(1)}$ and $s_1 \neq 0 \in \mathcal{S}$ we conclude that S has passed the defending test.

▶ **Definition 12** (admissible extension). Let S be a conflict-free set of arguments. S is admissible if and only if S is conflict-free and defends itself.

▶ **Proposition 13** (admissibility test). *Let S be a given set of arguments. For S to be admissible its matrix representation $\mathcal{S}$ has to pass both the conflict-free and defending test.*

Proof. The proof follows directly from Definition 12, Proposition 7 and Proposition 10. ◀

3.3 Stable extensions test

▶ **Definition 14** (stable extensions). Let S be a conflict-free set of arguments. S is called a stable extension if and only if every argument not in S is attacked by an argument in S.

Stable extensions are conflict-free and admissible. For any given set S to be stable it first has to pass the conflict-free test. We then want to see if S can attack all other arguments which do not belong to it. This is done with the stable extension test.

▶ **Proposition 15** (stable extensions test). *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework with adjacency matrix A. Let $S \subseteq \mathcal{A}$ be a given set of arguments and $\mathcal{S}$ the column vector of S and $\Gamma = \mathcal{S}^T A$. The set S passes the stable extensions test if and only if:*
1. *$\mathcal{S}$ passes the conflict-free test, and*
2. *$\forall i$ such that $s_i = 0, \gamma_i \neq 0$, ($s_i \in \mathcal{S}$, $\gamma_i \in \Gamma$).*

Proof. Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework, $S \subseteq \mathcal{A}$ be a set of arguments with $\mathcal{S}$ its column vector representation. Based on Definition 14, S should be conflict-free, i.e. it should pass the conflict-free test and every argument not in S should be attacked by an argument in S. $\Gamma = \mathcal{S}^T A$ is a row vector where its entry $(\mathcal{S}^T A)_{i,1}$ shows how many times argument $a_i \in \mathcal{A}$ is attacked by S. Fulfilling the constrain $\forall i$ such that $s_i = 0, \gamma_i \neq 0$ we make sure that every argument not in S should be attacked by an argument in S. ◀

Note that for the stable extension test we do not use the admissibility test. Intuitively, a set attacking anything that is "outside" of it means that it attacks all its potential attackers. This is true since passing the conflict-free test shows that there do not exist attacks coming "inside" of it thus any existing attacks should be from "outside" and it attack them back anyway.

▶ **Example 16** (stable extension test). For Example 3 illustrated in Figure 1, we check if set $S = \{a, c\}$, $\mathcal{S} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ passes the stable extension test. $\mathcal{S}^T A = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} = \Gamma$. First of all it passes the conflict-free test as $\gamma_2 \neq 0$ and $s_2 = 0$. Using $\mathcal{S}^T = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$ we ask which arguments in A are attacked by the set $S = \{a, c\}$. The answer is $\Gamma = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$ which means that it only attacks argument b. Based on the stable extension test, we compare matrices $\Gamma = \mathcal{S}^T A = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$ and $(\mathcal{S})^T = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$. $s_2 = 0$ and $\gamma_2 = 1$ and therefore it is stable.

3.4 Complete extensions test

▶ **Definition 17** (complete extension). An admissible set S of arguments is a complete extension if and only if S contains all arguments it defends.

Let S be a given set. For S to be complete it has to be admissible and for this reason its matrix representation has to first pass the admissibility test. We then have to check if it contains all arguments it defends. Note that if an argument has no attackers, it is trivially defended by any set. This check can be done through two matrix multiplications. The first multiplication $B = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}_{1 \times n} A_{n \times n}$ that finds all arguments that are not under attack and these arguments should be in any complete extension. The second multiplication is to check if the set S contains all arguments it defends. $\Gamma = (\mathcal{S}^T A)A = \mathcal{S}^T A^2$, where $Z = \mathcal{S}^T A$ will show (when mapped to the argumentation framework) all arguments that are under attack by the set S and $\Gamma = ZA$, will show all arguments that set S can defend. To find all complete extensions, we have to run the admissibility test with an extended set S that contains all possible combinations of the arguments in $\mathcal{A}$. This technic is time consuming and its computational complexity exponentially growths as the number of arguments not in S become bigger. To consider all possible combinations with n-many arguments, a matrix with 2^n number of columns has to be created.

▶ **Proposition 18** (complete extensions test). *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework with adjacency matrix $A_{n \times n}$. Let $S \subseteq \mathcal{A}$ with $\mathcal{S}$ its column vector. S passes the complete extension test if and only if:*

1. *$\mathcal{S}$ passes the admissibility test*
2. *Compute $B = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}_{1 \times n} A$ and $\Gamma = \mathcal{S}A^2$. For each entry $b_i = 0$ then $s_i \subseteq S$, and for each entry $\gamma_j \neq 0$ then $s_j \subseteq S$.*

Proof. The proof follows directly from Proposition 13, Definition 9 and Definition 17. ◀

▶ **Example 19.** Consider Example 3, its set $S = \{a, c\}$ and its matrix representation $\mathcal{S}^T = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$. Condition 1 is satisfied as $\mathcal{S}$ passes the admissibility test (see Example 11) and therefore S is admissible. To check if S is complete condition 2 should also be satisfied. Evaluate $B = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} A = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}$ and $\Gamma = \mathcal{S}^T A^2 = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$. Note that $b_1 = 0$ and $s_1 \subseteq \mathcal{S}$. Additionally, $\gamma_3 \neq 0$ and $s_3 \subseteq \mathcal{S}$. Thus condition 2 is satisfied and $S = \{a, c\}$ is complete.

4 Related work

We have introduced a matrix-based mathematical approach for answering questions of the form: "Is set A an extension?". Similar to this approach, is the work presented in [15]. Their approach is structured as follows: Consider the adjacency matrix of an argumentation framework and then define several parts of the adjacency matrix, which they call sub-blocks. Finding all sub-blocks that have zeroes everywhere, it is like finding the conflict-free sets

of the argumentation framework. Map each one of these sets into its matrix representation norm form, with respect to a specific semantics (stable, admissible, complete). Based on matrix criteria they find among the conflict-free sets which qualify as stable, admissible or complete extensions. Finally, select all extensions passing a given criteria based on matrix operations.

Our approach differs in the fact that we do not use sub-blocks but we also create sets of arguments in a matrix representation to define tests for the different argumentation semantics. Our work when considering all possible sets of arguments, comes close to their work as it answers similar questions but with a different algorithm. Our approach focuses on a given set of arguments as we do not try to find all arguments passing a given criteria.

5 Conclusion & Future work

Given a set S of arguments, we can present it as a matrix and perform different kinds of tests. Based on the results, we can answer whether a set S has a property, i.e. belongs to an argumentation extension of some kind.

In the work of Modgil et all [9] several questions are answered: 1) Does an extension exist? 2) Give an extension. 3) Give all extensions. Let A be a set of arguments. We can answer the question: "Is A an extension?". To find several extensions or all extensions we have to create such a matrix $\mathcal{S}^{'}$ where each one of its columns will represent all possible combination sets that the argumentation framework can define. Then by running tests under $\mathcal{S}^{'}$ we can tackle similar questions by comparing each column of the resulting matrix.

We have implemented a system, called $\mathcal{ASSA}$ that is able to answer different questions for stable semantics for any given abstract argumentation framework. This program creates all possible instances of selected set S into a vector form $\mathcal{S}$. Instead of handling each vector as an individual, it handles them all together when all vectors are combined into a massive matrix $\mathcal{S}^{'}$. By handling $\mathcal{S}^{'}$ in a similar way as $\mathcal{S}$ we manage to find all stable extensions that exist in an abstract argumentation framework. $\mathcal{ASSA}$ at this time is ineffective as it needs to create a massive matrix with 2^n number of columns. We plan to study ways to make this more effective. We also plan to extend our algorithms to cover other extensions (e.g. grounded, ideal) and to extend the $\mathcal{ASSA}$ system to handle these semantics as well.

There is a lot of interest in computing extensions at a point where a competition exist (see ICCMA'15). Other approaches that tackle similar results exist, such as the ASPARTIX (Answer Set Programming Argumentation Reasoning Tool) and DIAMOND [7, 8, 3] but all of them do not use matrices to compute extensions. As a future work, a comparison to these methods in relation to speed and complexity can be studied.

As matrix representation of argumentation frameworks has not received much attentions so far, and our approach constitutes an interesting new research direction, we hope many researches will be inspired and find the content stimulating and thought provoking. We are optimistic that our technique can be used to query graph related problems. Known properties of directed graph can improve our understanding of argumentation extensions since directed graphs allow us to approach the formalizations in a way that ignores the logical meaning and concentrates on their structural properties.

References

1 Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Eng. Review*, 26(4):365–410, 2011.

2 Martin W. A. Caminada and Dov M. Gabbay. A logical account of formal argumentation. *Studia Logica*, 93(2-3):109–145, 2009.

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

4 Yannis Dimopoulos and Alberto Torres. Graph theoretical structures in logic programs and default theories. *Theor. Comput. Sci.*, 170(1-2):209–244, 1996.

5 Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

6 Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Computing ideal sceptical argumentation. *Artif. Intell.*, 171(10-15):642–674, 2007.

7 Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. ASPARTIX: implementing argumentation frameworks using answer-set programming. In *Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings*, pages 734–738, 2008.

8 Stefan Ellmauthaler and Hannes Strass. The DIAMOND system for computing with abstract dialectical frameworks. In *Computational Models of Argument – Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014*, pages 233–240, 2014.

9 S. Modgil and Martin W.A. Caminada. Proof theories and algorithms for abstract argumentation frameworks. In I. Rahwan and G. Simari, editors, *Argumentation in Artif. Intell.*, pages 105–129. Springer Publishing Company, Incorporated, 2009.

10 Christos H Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

11 Gilbert W Stewart. *Introduction to matrix computations*. Academic Press, 1973.

12 Andrew James Stothers. *On the complexity of matrix multiplication*. PhD thesis, The University of Edinburgh, 2010.

13 A. Torres. Negation as failure to support. In L. M. Pereira and A. Nerode, editors, *Logic Programming and Non-Monotonic Reasoning: Proc. of the Second International Workshop. Cambridge*, pages 223–243. MIT Press, MA, 1993.

14 Y. Wu and M.W.A. Caminada. A labelling-based justification status of arguments. *Studies in Logic*, 3(4):12–29, 2010.

15 Yuming Xu. A matrix approach for computing extensions of argumentation frameworks. *CoRR*, abs/1209.1899, 2012.