Computing Preferred Labellings by Exploiting SCCs and Most Sceptically Rejected Arguments

Beishui Liao, Liyun Lei, and Jianhua Dai

Center for the Study of Language and Cognition,
Zhejiang University, Hangzhou, 310028, P.R. China

Abstract. The computation of preferred labellings of an abstract argumentation framework (or briefly, AF) is generally intractable. The existing decomposition-based approach by exploiting strongly connected components (SCCs) of a general AF is promising to cope with this problem. However, the efficiency of this approach is highly limited by the maximal SCC of an AF. This paper presents a further solution by exploiting the most sceptically rejected arguments of an AF. Given an AF, its grounded labelling is first generated. Then, the attacks between the undecided arguments and the rejected arguments are removed. It turns out that the modified AF has the same preferred labellings as the original AF, but the maximal SCC in it could be much smaller than that of the original AF. Empirical results show that this new method dramatically reduce the computation time for some sparse AFs (when the ratio of the number of edges to the number of nodes of an AF is between 1:1 and 1.8:1).

Keywords: Argumentation, Semantics, Labellings, Computational Complexity

1 Introduction

Argumentation is an increasingly active research area in AI. One of the most important problems of this area is that many natural problems are computationally intractable [1]. While the worst-case computational complexity of argumentation has been well formulated, how to efficiently compute the argumentation semantics is still an open problem. To the best of our knowledge, the existing work related to this problem mainly consists of the following three lines. The first line of work is on identifying tractable classes of argumentation frameworks (AFs) with special structures [1], and developing efficient algorithms for some classes of AFs with fixed parameters, such as bounded tree-width [2] and bounded clique-width [3], etc. And, in [4], Dvořák et al proposed a generic approach for solving hard problems in the area of argumentation in a "complexity-sensitive" way. The corresponding empirical results showed that their approach significantly outperforms existing systems developed for hard argumentation problems (i.e., problems under the preferred, semi-stable, or stage semantics. The second line of work is on developing more efficient algorithms by means of some specific mechanisms. For instance, in [5], the authors proposed a more efficient algorithm for enumerating all preferred extensions, by utilizing further labels to improve labels' transitions. The third line of work is on decomposition-based computation. In [6], the authors proposed an SCC-recursive scheme for argumentation semantics, based on decomposition along the SCCs of an AF. In [7], the authors developed splitting-based algorithms for the computation of extensions. Their experimental results showed an average improvement by 50% and by 54% for preferred and stable semantics respectively, compared to Modgil and Caminada's algorithms [8]. In [9] and [10], the authors proposed methods to efficiently compute, respectively, the dynamic semantics and the partial semantics of argumentation, by means of decomposition and semantics combination. In [11], we formulated an approach to compute the extensions of an AF by exploiting its SCCs and acyclic fragments.

While the existing approaches have made some progress on developing tractable algorithms for some AFs with special topologies or more efficient algorithms for a general AF, there are no (approximately) tractable algorithms for a general AF in which the ratio of the number of attacks to the number of arguments is no less than 1:1. According to the theory formulated in [11], one possible way to cope with this problem is to decompose an AF into a set of SCCs, and compute the status of arguments in each SCC separately. However, the efficiency of this approach is highly limited by the size of the *maximal* SCC.

In this paper, we introduce a further solution by exploiting *most sceptically rejected arguments* (or briefly, MSR arguments) and SCCs of an AF. The feasibility of this approach lies in a new discovery that after removing some attacks related to MSR arguments from an AF, the status of arguments in the AF are unaffected, while the maximal SCC of the modified AF is often much smaller than that of the original AF. Since preferred semantics is a typical semantics of argumentation, and its computation is one of the most difficult ones, in this paper, for simplicity and without loss of generality, we only consider the computation under this semantics. Furthermore, since the labelling-based approach is one of the two mainstream approaches for formulating argumentation semantics, and it is closer to algorithms, we only study the computation of the preferred semantics that is formulated by the labelling-based approach.

The remaining contents of this paper are organised as follows. In the next section, we briefly introduce some basic notions of argumentation and some typical algorithms for computing argumentation semantics. In Section 3, we introduce an approach for computing preferred labellings by exploiting the SCCs of an AF. In Sections 4 and 5, we first propose a further solution by exploiting both the SCCs and MSR arguments of an AF, and then conduct an empirical investigation. Finally, in Section 6, we conclude the paper.

2 Preliminaries

2.1 Semantics of argumentation framewroks

In this paper, we only deal with (abstract) argumentation frameworks [12]. An argumentation framework (or briefly, AF) is defined as a tuple (A, R), in which

A is a set of arguments and $R \subseteq A \times A$ is a set of attacks. For all $\alpha, \beta \in A$, we often use $(\alpha, \beta) \in R$ to denote that α attacks β. It is obvious that an AF is in fact a directed graph (often called a *defeat graph*), where the nodes represent arguments and edges represent attacks. Figure 1 illustrates an AF (A_1, R_1).

$$\begin{tikzcd} 1 \arrow[r, bend left] & 2 \arrow[l, bend left] \arrow[r] & 3 \end{tikzcd}$$

Fig. 1. (A_1, R_1)

Given an AF, a fundamental problem is to determine which arguments can be regarded as (collectively) acceptable. There are two mainstream approaches to resolve this problem: extension-based approach and labelling-based approach [13]. The former defines various criteria (called *argumentation semantics*) under which a set (sets) of arguments are regarded as acceptable, while the latter is to assign a "reasonable" label to each argument, according to some criteria.

In the extension-based approach, a set of collectively acceptable arguments is called an *extension.* A core notion of this approach is *admissible sets.* Specifically, given an AF (A, R), a set of arguments is *admissible*, if and only if it is *conflict-free* and it can *defend* each argument within the set. A set $B \subseteq A$ is *conflict-free* if and only if there exist no arguments α and β in B such that $(\alpha, \beta) \in R$. Argument $\alpha \in A$ is *defended* by a set $B \subseteq A$ if and only if for all $\beta \in A$, if $(\beta, \alpha) \in R$, then there exists $\gamma \in B$ such that $(\gamma, \beta) \in R$. An admissible set is called a *complete extension*, if and only if it contains all arguments it can defend. Given an AF, there might exist several complete extensions, in which the maximal ones (w.r.t. set inclusion) are called *preferred extensions*, while the minimal one (w.r.t. set inclusion) is called the *grounded extension* (the grounded extension of an AF is unique).The AF (A_1, R_1) has three complete extensions $\{\}$, $\{1, 3\}$ and $\{2\}$, two preferred extensions $\{1, 3\}$ and $\{2\}$, and one grounded extension $\{\}$.

On the other hand, in the labelling-based approach, there are usually three different labels: IN, OUT and UNDEC. An argument is IN if all its attackers are OUT. An argument is OUT if it is attacked by an argument that is IN. An argument is UNDEC, if it is neither IN nor OUT [14]. Given (A, R) and the three labels, a *labelling* is a total function $\mathcal{L} : A \mapsto \{\text{IN, OUT, UNDEC}\}$. The definition of labelling-based semantics is based on the notion of *legal labelling.* More specifically, an argument is legally IN if and only if it is labelled IN and each attacker is labelled OUT; an argument is legally OUT if and only if it is labelled OUT and there exists an attacker that is labelled IN; an argument is legally UNDEC if and only if it is not the case that (1) each attacker is labelled OUT or (2) there exists an attacker that is labelled IN. Then, a labelling $\mathcal{L}$ is called an *admissible labelling*, if and only if each IN-labelled argument is legally IN, and each OUT-labelled argument is legally OUT; $\mathcal{L}$ is called a *complete labelling*, if and only if it is an admissible labelling and each UNDEC-labelled argument is legally UNDEC; $\mathcal{L}$ is called a *preferred labelling*, if and only if it is an admissible labelling and the set of IN-labelled arguments is maximal; $\mathcal{L}$ is called a *grounded labelling*, if and only if it is a complete labelling and the set of IN-labelled arguments is minimal.

Let $in(\mathcal{L}) = \{\alpha : \mathcal{L}(\alpha) = \text{IN}\}$, $out(\mathcal{L}) = \{\alpha : \mathcal{L}(\alpha) = \text{OUT}\}$ and $undec(\mathcal{L}) = \{\alpha : \mathcal{L}(\alpha) = \text{UNDEC}\}$. A labelling $\mathcal{L}$ is often represented as a triple of the form $(in(\mathcal{L}), out(\mathcal{L}), undec(\mathcal{L}))$. Accordingly, the AF (A_1, R_1) in Figure 1 has two preferred labellings: $\mathcal{L}_1 = (\{1, 3\}, \{2\}, \{\})$ and $\mathcal{L}_2 = (\{2\}, \{1, 3\}, \{\})$, as illustrated in Figure 2.

$$\begin{tikzcd}
\text{IN} & \text{OUT} & \text{IN} & \text{OUT} & \text{IN} & \text{OUT} \\
1 \arrow[r, bend left] & 2 \arrow[l, bend left] \arrow[r] & 3 & 1 \arrow[r, bend left] & 2 \arrow[l, bend left] \arrow[r] & 3
\end{tikzcd}$$

Fig. 2. Preferred labellings of (A_1, R_1)

As summarised in [13], there exists a bijective correspondence between complete (respectively, preferred and grounded) labelling(s) and complete (respectively, preferred and grounded) extension(s).

2.2 Algorithms for computing argument labellings

In existing literature, there are mainly two approaches for computing labellings (extensions): labelling-based algorithms and answer-set programming. It has been recognised that Modgil and Caminada's labelling-based algorithms (briefly, MC algorithms) [8] have received much attention and been compared with some newly proposed algorithms ([7], [5], etc).

According to MC algorithms, generating the grounded labelling of an AF is simple. It starts by assigning IN to all arguments that are not attacked, and then iteratively: OUT is assigned to any argument that is attacked by an argument that has just been made IN, and then IN to those arguments all of whose attackers are OUT. The iteration continues until no more new arguments are made IN or OUT. Any arguments that remain unlabelled are then assigned UNDEC.

By comparison, the MC algorithm for computing preferred labellings is more complex. It computes *admissible labellings* that maximise the number of arguments that are legally IN. Admissible labellings are generated by starting with a labelling that labels all arguments IN and then iteratively, selects arguments that are illegally IN (or *super-illegally* IN) and applies a *transition step* to obtain a new labelling, until a lablling is reached in which no argument is illegally IN. For the details of this algorithm, readers may refer to [8].

3 An approach by exploiting SCCs

In this section, based on [11], we introduce an approach for the computation of preferred labellings by exploiting the SCCs of an AF (called the *SCC-based* *approach*). The basic idea of this approach is as follows. Given an AF, it is first decomposed into a set of sub-frameworks according to the SCCs of the AF. Then, along the order of SCCs, the preferred labellings of the sub-frameworks are generated separately, and combined incrementally to form the labellings of the original AF.

Decomposing a general AF according to its SCCs Since an AF can be viewed as a directed graph and the set of SCCs of a directed graph can be obtained by a polynomial time algorithm [15], it is intuitively feasible to decompose an AF along its SCCs [6].

According to graph theory, an important property of SCCs is that every directed graph is a directed *acyclic* graph of its SCCs. Consider an AF (A_2, R_2) in Figure 3(a). The directed graph of its SCCs is shown in Figure 3(b).

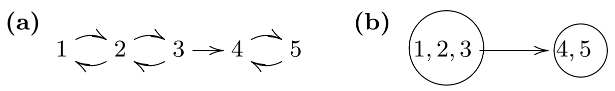


Fig. 3. (A_2, R_2) and the directed acyclic graph of its SCCs

Since there exists a partial order over all SCCs, it is possible to compute separately the preferred labellings of each *sub-framework* induced by an SCC. Now, let us define the notion of a sub-framework induced by an SCC.

Let $\{C_1, \ldots, C_n\}$ be the set of SCCs of an AF (A, R). It holds that $C_1, \ldots,$ and C_n are a partition of A. Let $R_{C_i} = R \cap (C_i \times C_i)$ be the set of attacks between the arguments in C_i, $C_i^- = \{\alpha \in A \backslash C_i : \exists \beta \in C_i$, such that $(\alpha, \beta) \in R\}$ be the set of arguments outside C_i that attack the arguments in C_i, and $I_{C_i} = R \cap (C_i^- \times C_i)$ be the set of interactions from the arguments in C_i^- to the arguments in C_i, in which $1 \leq i \leq n$. In terms of [9], C_i^- is called the set of *conditioning arguments*. A sub-framework of (A, R) induced by C_i is then defined as a tuple:

$$(C_i \cup C_i^-, R_{C_i} \cup I_{C_i}) \tag{1}$$

Computing the preferred labellings of each sub-framework In Formula (1) , when $C_i^- = \emptyset$ (and thus $I_{C_i} = \emptyset$), $(C_i \cup C_i^-, R_{C_i} \cup I_{C_i}) = (C_i, R_{C_i})$. In this case, the sub-framework is not related to any external arguments. Hence, its preferred labellings can be computed independently. On the contrary, when $C_i^- \neq \emptyset$, the labels of arguments in C_i^- are not assigned within $(C_i \cup C_i^-, R_{C_i} \cup I_{C_i})$, but assigned in an external sub-framework.

Consider the example in Figure 3. According to Formula (1), we get two sub-frameworks as shown in Figures 4(a) and 4(b). The sub-framework in Figure 4(a) has two preferred labellings: $\mathcal{L}_1 = (\{1, 3\}, \{2\}, \{\})$ (Figure 4(c)) and $\mathcal{L}_2 = (\{2\}, \{1, 3\}, \{\})$ (Figure 4(e)). With respect to $\mathcal{L}_k$ $(k \in \{1, 2\})$, we get a *partially labelled sub-framework* of $(\{3, 4, 5\}, \{(3, 4), (4, 5), (5, 4)\})$, denoted as

$(\{3,4,5\},\{(3,4),(4,5),(5,4)\})^{\mathcal{L}_k}$, in which the label of the conditioning argument 3 conforms to $\mathcal{L}_k$. These two partially labelled sub-frameworks are respectively illustrated in Figures 4(d) and 4(f).

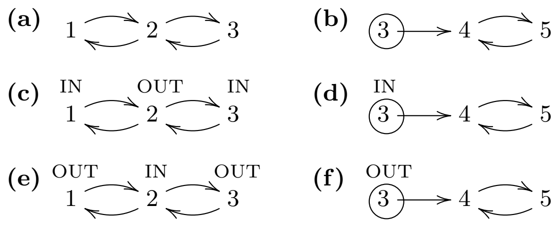


Fig. 4. Sub-frameworks and partially labelled sub-frameworks

Let $(P \cup P^-, R_P \cup I_P)^{\mathcal{L}}$ be a partially labelled sub-framework of (A,R), in which the labels of arguments in P^- conform to $\mathcal{L}$ that is a preferred labelling of an external sub-framework. Let $\mathcal{L}'$ be a labelling of $(P \cup P^-, R_P \cup I_P)^{\mathcal{L}}$, such that the labels for arguments in P^- conform to $\mathcal{L}$, while each argument in P is assigned with a new label. Then, $\mathcal{L}'$ is called an *admissible labelling*, if and only if each argument in P that is labeled IN is legally IN, and each argument in P that is labeled OUT is legally OUT. $\mathcal{L}'$ is called a *preferred labelling*, if and only if it is an admissible labelling and the set of arguments in P that are labelled IN is maximal. Let us return to the above example. It holds that $\mathcal{L}_3 = (\{3,5\},\{4\},\{\})$ is a preferred labelling of $(\{3,4,5\},\{(3,4),(4,5),(5,4)\})^{\mathcal{L}_1}$ (Figure 4(d)), in that argument 5 is legally IN, argument 4 is legally OUT, and $\{5\}$ is the maximal set of arguments in $\{4,5\}$ that are legally IN. Similarly, $\mathcal{L}_4 = (\{4\},\{3,5\},\{\})$ and $\mathcal{L}_5 = (\{5\},\{3,4\},\{\})$ are preferred labellings of $(\{3,4,5\},\{(3,4),(4,5),(5,4)\})^{\mathcal{L}_2}$ (Figure 4(f)).

Labelling combination When an AF has only two SCCs, in which one is restricted by another, the labelling combination is simple. Formally, let (A,R) be an AF, and P and Q be a partition of A, such that $P^- \subseteq Q$ and $Q^- = \{\}$. For every preferred labelling $\mathcal{L}$ of (Q,R_Q), $(P \cup P^-, R_P \cup I_P)^{\mathcal{L}}$ is a partially labelled sub-framework. Then, for every preferred labelling $\mathcal{L}'$ of $(P \cup P^-, R_P \cup I_P)^{\mathcal{L}}$, the combination of $\mathcal{L}$ and $\mathcal{L}'$ is defined as $\mathcal{L} + \mathcal{L}' = (in(\mathcal{L}) \cup in(\mathcal{L}'), out(\mathcal{L}) \cup out(\mathcal{L}'), undec(\mathcal{L}) \cup undec(\mathcal{L}'))$, which is a combined labelling of (A,R). For instance, the preferred labellings of (A_2,R_2) in Figure 3(a) can be obtained by the following way. We combine $\mathcal{L}_1$ with $\mathcal{L}_3$, $\mathcal{L}_2$ with $\mathcal{L}_4$, and $\mathcal{L}_2$ with $\mathcal{L}_5$, obtaining three combined labellings as follows: $(\{1,3,5\},\{2,4\},\{\})$, $(\{2,4\},\{1,3,5\},\{\})$ and $(\{2,5\},\{1,3,4\},\{\})$. In [9], we have proved the soundness and completeness of this kind of semantic combination under the context of extension-based approach. Since there is a one-to-one correspondence between sets of preferred labellings and sets of preferred extensions, the above labelling combination is correct.

When an AF has more than two SCCs, its sub-frameworks are organised into several layers conforming to the partial order of the SCCs of the AF. Then, the labellings of the AF are computed and combined incrementally, from the lowest layer in which each sub-framework is not restricted by other sub-frameworks, to the highest layer in which each sub-framework is most restricted by the sub-frameworks located in the lower layers. The following example illustrates the process of incremental combination of preferred labellings. Compared to (A_2, R_2), (A'_2, R'_2) in Figure 5(a) has two more sub-frameworks $(\{1,6\}, \{(1,6)\})$ (in which 1 is a conditioning argument) and $(\{3,4,7\}, \{(3,7),(4,7)\})$ (in which 3 and 4 are conditioning arguments), located in the second and the third layer, respectively. With respect to $\mathcal{L}_1$ and $\mathcal{L}_2$ mentioned above (Figures 4(c) and 4(e)), there are two partially labelled sub-frameworks of $(\{1,6\}, \{(1,6)\})$, i.e., $(\{1,6\}, \{(1,6)\})^{\mathcal{L}_1}$ and $(\{1,6\}, \{(1,6)\})^{\mathcal{L}_2}$. The former has a preferred labelling $\mathcal{L}_6 = (\{1\}, \{6\}, \{\})$, while the later has a preferred labelling $\mathcal{L}_7 = (\{6\}, \{1\}, \{\})$. Before the labellings of the second layer are combined with those of the first layer, the labellings of the sub-frameworks in the second layer are first combined. After combination, $\mathcal{L}_3+\mathcal{L}_6$ is a preferred labelling of $(\{1,3,4,5,6\}, \{(3,4),(4,5),(5,4), (1,6)\})^{\mathcal{L}_1}$, and $\mathcal{L}_4 + \mathcal{L}_7$ and $\mathcal{L}_5 + \mathcal{L}_7$ are preferred labellings of $(\{1,3,4,5,6\}, \{(3,4),(4,5),(5,4),(1,6)\})^{\mathcal{L}_2}$. Then, the labellings of the first and the second layer are combined, resulting $\mathcal{L}_1+\mathcal{L}_3+\mathcal{L}_6 = (\{1,3,5\}, \{2,4,6\}, \{\})$, $\mathcal{L}_2+\mathcal{L}_4+\mathcal{L}_7 = (\{2,4,6\}, \{1,3,5\}, \{\})$ and $\mathcal{L}_2 + \mathcal{L}_5 + \mathcal{L}_7 = (\{2,5,6\}, \{1,3,4\}, \{\})$, which are prefered labellings of the sub-framework $(\{1,2,3,4,5,6\}, \{(1,2),(2,1),(2,3),(3,2), (3,4),(4,5),(5,4),(1,6)\})$. And then, let $\mathcal{L}_8 = \mathcal{L}_1+\mathcal{L}_3+\mathcal{L}_6$, $\mathcal{L}_9 = \mathcal{L}_2+\mathcal{L}_4+\mathcal{L}_7$ and $\mathcal{L}_{10} = \mathcal{L}_2+\mathcal{L}_5+\mathcal{L}_7$. With respect to $\mathcal{L}_8$, $\mathcal{L}_9$ and $\mathcal{L}_{10}$, in the third layer, there are three partially labelled sub-frameworks of $(\{3,4,7\}, \{(3,7),(4,7)\})$, i.e., $(\{3,4,7\}, \{(3,7),(4,7)\})^{\mathcal{L}_8}$, $(\{3, 4,7\}, \{(3,7),(4,7)\})^{\mathcal{L}_9}$ and $(\{3,4,7\}, \{(3,7),(4,7)\})^{\mathcal{L}_{10}}$. Sets of preferred labellings of them are respectively $\{\mathcal{L}_{11}\}$, $\{\mathcal{L}_{12}\}$ and $\{\mathcal{L}_{13}\}$, in which $\mathcal{L}_{11} = (\{3\}, \{4,7\}, \{\})$, $\mathcal{L}_{12} = (\{4\}, \{3,7\}, \{\})$ and $\mathcal{L}_{13} = (\{7\}, \{3,4\}, \{\})$. Finally, the preferred labellings of the third layer and the labellings of the previous two layers are combined, resulting $\mathcal{L}_8 + \mathcal{L}_{11} = (\{1,3,5\}, \{2,4,6,7\}, \{\})$, $\mathcal{L}_9 + \mathcal{L}_{12} = (\{2,4,6\}, \{1,3, 5,7\}, \{\})$ and $\mathcal{L}_{10} + \mathcal{L}_{13} = (\{2,5,6,7\}, \{1,3,4\}, \{\})$, which are the preferred labellings of (A'_2, R'_2).

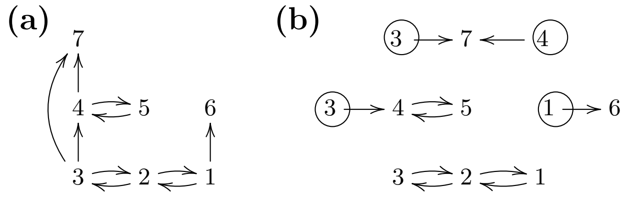


Fig. 5. (A'_2, R'_2) and a layered decomposition of it

4 A further solution by exploiting both SCCs and most sceptically rejected arguments

As mentioned in Section 1, the efficiency of the above SCC-based approach is highly limited by the size of the maximal SCC. Let us consider the AF (A_3, R_3) as shown in Figure 6. It has only two SCCs: $\{1, \dots, 6\}$ and $\{7\}$. The size of the maximal SCC is six. Hence, in this case, little execution time could be saved by using the SCC-based approach. In order to make the SCC-based approach more efficient, a natural idea is *to modify an AF such that the status of arguments in the original AF remains unchanged, while the size of the maximal SCC of the modified AF becomes smaller.*

$$\begin{tikzcd}
1 \arrow[r, bend left] & 2 \arrow[l, bend left] \arrow[r, bend left] & 3 \arrow[l, bend left] \arrow[r] & 4 \arrow[r, bend left] & 5 \arrow[l, bend left] \arrow[r] & 6 \arrow[lll, bend right] & 7 \arrow[l]
\end{tikzcd}$$

Fig. 6. (A_3, R_3)

In order to realise this idea, we resort to the *most sceptically rejected arguments* (briefly, MSR arguments) of an AF. We say an argument is most sceptically rejected, if it is labelled OUT in the grounded labelling of an AF.

Proposition 1. *Let $\mathcal{L}_g = (in(\mathcal{L}_g),\ out(\mathcal{L}_g), undec(\mathcal{L}_g))$ be the grounded labelling of an AF (A, R). The interactions between $out(\mathcal{L}_g)$ and $undec(\mathcal{L}_g)$ do not influence the preferred labellings of (A, R).*

Proof. Let $\mathcal{L}_p = (in(\mathcal{L}_p),\ out(\mathcal{L}_p), undec(\mathcal{L}_p))$ be a preferred labelling of (A, R). Since the grounded extension is contained in every preferred extension [13], the arguments labelled OUT in $\mathcal{L}_g$ are also labelled OUT in $\mathcal{L}_p$. Let (α, β) be an interaction from $out(\mathcal{L}_g)$ to $undec(\mathcal{L}_g)$. It follows that β is attacked by an argument α that is itself OUT in $\mathcal{L}_p$. Hence, whether β belongs to $in(\mathcal{L}_p)$, $out(\mathcal{L}_p)$ or $undec(\mathcal{L}_p)$, (α, β) does not influence $\mathcal{L}_p$. On the other hand, let (α, β) be an interaction from $undec(\mathcal{L}_g)$ to $out(\mathcal{L}_g)$. Since $\beta \in out(\mathcal{L}_g)$, it is attacked by a third argument $\gamma \in in(\mathcal{L}_g) \subseteq in(\mathcal{L}_p)$. Since attacking an argument that is already OUT has no effect, (α, β) does not affect $\mathcal{L}_p$.

Let us consider (A_3, R_3) again. Since argument 6 is an MSR argument, after we remove the attacks $(5, 6)$ and $(6, 3)$ from the framework, we get a modified framework (A_3, R_3') (Figure 7), which has the same preferred labellings as the original one. Now, the size of the maximal SCC $\{1, 2, 3\}$ is three.

Let (A, R') be the remaining part of (A, R) after removing the interactions between $out(\mathcal{L}_g)$ and $undec(\mathcal{L}_g)$. According to Proposition 1, (A, R') and (A, R) have the same preferred labellings.

Since the computation of the grounded labelling of (A, R) is polynomial time tractable, (A, R') can be obtained easily. The preferred labellings of (A, R') are then computed by the SCC-based approach described above.

$$\begin{tikzcd}
1 \arrow[r, bend left] & 2 \arrow[l, bend left] \arrow[r, bend left] & 3 \arrow[l, bend left] \arrow[r] & 4 \arrow[r, bend left] & 5 \arrow[l, bend left] & 6 & 7 \arrow[l]
\end{tikzcd}$$

Fig. 7. (A_3, R'_3)

5 Empirical investigation

In previous sections, we have introduced three approaches for computing the preferred labellings of a general AF, including the MC algorithm, the SCC-based approach and the approach by exploiting both SCCs and MSR arguments (called *SCC-MSR approach*). In the SCC-based approach, the algorithm for generating preferred labellings of each sub-framework is based on MC algorithm with a slight modification such that the preferred labellings of a partially labelled sub-framework can be generated. Meanwhile, the SCC-MSR approach is in turn directly established on top of the SCC-based approach.

The above approaches were implemented in Java, and tested on a machine with an Intel CPU running at 1.86 GHz and 1.98 GB RAM.

First, we tested the average sizes of the maximal SCCs of AFs in the SCC-based approach and the SCC-MSR approach, respectively. Given an assignment of edge density (#edges/#nodes = 1, 1.2, ..., 4) and the size of AFs (#nodes =50, 500, 5000), the programs (in which the components for generating preferred labellings were disabled) of the two approaches were executed **100** times. In each time, an AF with the given edge density and size is generated at random, and the size of the maximal SCC produced by each approach was recorded. The average results are illustrated in Figure 8, where S[n] (SM[n]) (n = $50, 500$, or 5000) denotes that the results were produced by the SCC-based approach (respectively, the SCC-MSR approach) and the size of every AF is n. From this figure, we may observe that when the edge density of AFs is sparse (#edges/#nodes $\leq$2) , for a given AF, the percentage of arguments in the maximal SCC (denoted as "maxscc/#nodes" where "maxscc" represents the size of the maximal SCC) produced by the SCC-MSR approach is much smaller than the one produced by the SCC-based approach. For instance, when #edges/#nodes =1.8 and #nodes =500, the average number of arguments in the maximal SCCs produced by the SCC-based approach is 264, but only 2 by the SCC-MSR approach.

Second, we tested the performance of the SCC-MSR approach by comparing it with other two approaches. Given an assignment of edge density (#edges/ #nodes = 1, 1.1, ..., 2) and the size of AFs (#nodes =200,1000), the programs of the three approaches were executed **20** times. In each time, an AF with the given edge density and size is generated at random, and then its preferred labellings were generated by the three approaches respectively. The overall execution time of each approach was recorded. In the SCC-based approach, the overall execution time is mainly used for generating a set of SCCs, constructing a set of layered sub-frameworks, and generating and combining the preferred labellings of all sub-frameworks. In the SCC-SMR approach, the overall execution time is mainly used for generating the grounded labelling of a given AF, and computing the

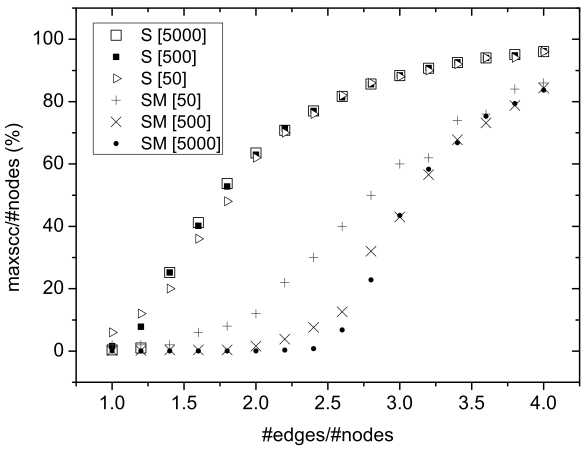
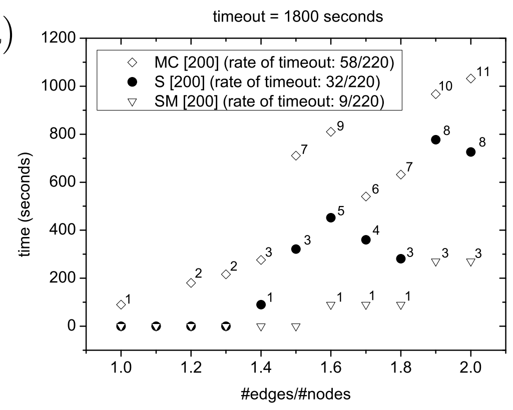


Fig. 8. Average results of the sizes of maximal SCCs generated by the SCC-based approach and the SCC-MSR approach

preferred labellings of the modified AF by using the SCC-based approach. Since in many cases, the overall execution time may last very long, to make the test easier, when the time for computing the preferred labellings of an AF is over **30** minutes, we stopped the execution by setting a break in the program. The average results of this test are illustrated in Figure 9, where MC[n] ($n = 200$, or 1000) denotes that the results were produced by the MC algorithm and the size of every AF is n, similar to the meanings of S[n] and SM[n] mentioned above. Each number near a symbol in the graph indicates the number of timeouts among the 20 times of execution (the overall rate of timeout is indicated in the legend of the plots). For instance, when #nodes = 200 and #edges/#nodes = 1.5 (in Figure 9(a)), there were 7 timeouts in the MC algorithm, 3 timeouts in the SCC-based approach and 0 timeout in the SCC-MSR approach. Table 1 shows the detailed records of this case.

When an execution is timeout, we use 30 minutes (1800 seconds) in computing the average execution time. From this table we found that the execution of the SCC-MSR approach under this configuration is very low (less than 0.016 seconds in all cases), while the execution time of other two approaches fluctuates from 0.015 seconds to more than 30 minutes. In addition, from Table 1, we also observed that in the SCC-based approach and the SCC-MSR approach, the time for generating SCCs, constructing sub-frameworks, combining preferred labellings and computing the grounded labelling is negligible when we compare it to the time for generating preferred labellings.

According to the results shown in Figure 9, when the ratio of the number of edges to the number of nodes of an AF is between 1:1 and 1.8:1, its preferred

(a)


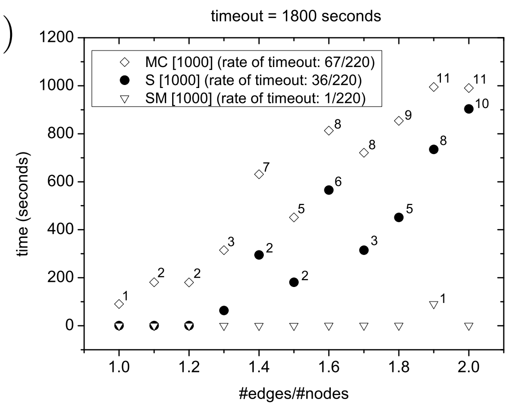
(b)


Fig. 9. Average results of the overall execution time of the three approaches.

labellings are approximately polynomial time tractable. In order to make this point more clear, we conducted a further test on the SCC-MSR approach. In this test, given an edge density (#edges/#nodes = 1.3, 1.5, 1.7) and the size of AFs (#nodes =100, 200, . . ., 1000), the program of the SCC-MSR approach was executed **200** times. Meanwhile, the timeout is set to **2** seconds. The results in Figure 10 show that when #edges/#nodes = 1.3 (respectively, 1.5 and 1.7), there were only 6 (respectively, 15 and 56) timeouts among the 2000 (= 200×10) times of execution.

No.	MC [200]<1.5> (seconds)	S [200]<1.5> (seconds)	SM [200]<1.5> (seconds)
01	0.031	0.016	0.016
02	0.016	0.015	0.016
03	timeout	0.031	0.015
04	timeout	0.015	0
05	0.391	0.062	0.015
06	0.016	0.015	0.015
07	1613.016	2.141	0.015
08	timeout	timeout	0
09	0.015	0.016	0.016
10	0.015	0.016	0.016
11	0.016	0.031	0.015
12	timeout	timeout	0
13	0.046	0.016	0
14	timeout	1030.188	0
15	0.047	0.015	0.016
16	0.031	0.016	0.015
17	0.031	0.016	0
18	timeout	0.032	0
19	0.016	0.015	0.016
20	timeout	timeout	0.015
Avg.	**710.684 (7)**	**321.633 (3)**	**0.010**

Table 1. The overall execution time of the three approaches when #nodes = 200 and #edges/#nodes = 1.5

6 Conclusions

In this paper, we have proposed an efficient method to compute the preferred labellings of a general AF by exploiting both its SCCs and most sceptically rejected arguments. The empirical results show that the computation time decreases *dramatically* when the defeat graphs are sparse. As illustrated in Figure 9, when the ratio between the number of edges (attacks) to the number of nodes (arguments) is less than 1.8:1, the computation of preferred labellings of various AFs is approximately polynomial time tractable. Meanwhile, when the edge density keeps the same, the average computation time tends to decrease when the number of nodes becomes bigger (as shown in Figure 9). The fundamental reason behind these phenomena is that after removing the most sceptically rejected arguments, the maximal SCC of the modified AF is smaller or much smaller than that of the original AF (as shown in Figure 8).

To the best of our knowledge, although much work has been done in developing efficient algorithms for some classes of AFs with fixed parameters (e.g., [2] and [3]), there are still no research efforts on finding approximately tractable algorithms to compute the preferred labellings (extensions) of a general AF in

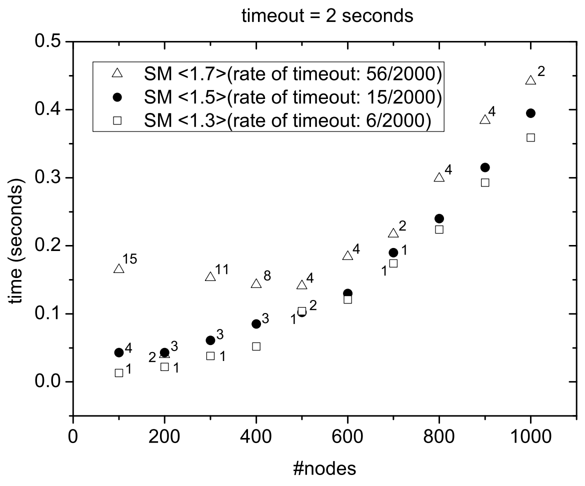


Fig. 10. Average results of the overall execution time of the SCC-MSR approach

which the ratio of the number of attacks to the number of arguments is no less than 1:1.

Although this paper only focused on the computation of preferred labellings, the computational mechanism of the SCC-MSR approach is not restricted to the preferred semantics. The application of this approach under other argumentation semantics will be our future work.

Funding

The research reported in this article was financially supported by the National Social Science Foundation Major Project of China under grant No.11&ZD088, the National Grand Fundamental Research 973 Program of China under grant No.2012CB316400 and the National Natural Science Foundation of China under grant No.61175058 and No.60773177.

References

1. Dunne, P.E.: Computational properties of argument systems satisfying graph-theoretic constraints. Artificial Intelligence **171** (2007) 701–729
2. Dvorák, W., Pichler, R., Woltran, S.: Towards fixed-parameter tractable algorithms for abstract argumentation. Artificial Intelligence **186** (2012) 1–37
3. Dvorák, W., Szeider, S., Woltran, S.: Reasoning in argumentation frameworks of bounded clique-width. In: Proceedings of the 3th International Conference on Computational Models of Argument, Desenzano del Garda, Italy, IOS Press (September 2010) 219–230

4. Dvořák, W., Jărvisalo, M., Wallner, J.P., Woltran, S.: Complexity-sensitive decision procedures for abstract argumentation. In: Proceedings of the KR2012, AAAI Press (2012) 54–64
5. Nofal, S., Dunne, P.E., Atkinson, K.: On preferred extension enumeration in abstract argumentation. In: Proceedings of the 4th International Conference on Computational Models of Argument, Vienna, Austria, IOS Press (September 2012) 205–216
6. Baroni, P., Giacomin, M., Guida, G.: Scc-recursiveness: a general schema for argumentation semantics. Artificial Intelligence **168** (2005) 162–210
7. Baumann, R., Brewka, G., Wong, R.: Splitting argumentation frameworks: An empirical evaluation. In: Proceedings of the 1st International Workshop on Theory and Applications of Formal Argumentation, Barcelona, Catalonia (Spain), Springer (July 2011) 17–31
8. Modgil, S., Caminada, M.: Proof theories and algorithms for abstract argumentation frameworks. In: Rahwan, G. R. Simari (eds.), Argumentation in Artificial Intelligence, Springer (2009) 105–129
9. Liao, B., Jin, L., Koons, R.C.: Dynamics of argumentation systems: A division-based method. Artificial Intelligence **175**(11) (2011) 1790–1814
10. Liao, B., Huang, H.: Partial semantics of argumentation: Basic properties and empirical results. Journal of Logic and Computation **doi:10.1093/logcom/exs047** (November 2012)
11. Liao, B., Huang, H.: Computing the extensions of an argumentation framework based on its strongly connected components. In: Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence, Athens, Greece (September 2012)
12. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence **77** (1995) 321–357
13. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. The Knowledge Engineering Review **26**(4) (2011) 365–410
14. Caminada, M.: On the issue of reinstatement in argumentation. In: Proceedings of the 10th European Conference on Logics in Artificial Intelligence, Liverpool, UK, Springer (September 2006) 111–123
15. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM Journal on Computing **1** (1972) 146–160