Computing Semi-Stable Semantics of AF by 0-1 Integer Programming

Mauricio Osorio, Juan Díaz, and Alejandro Santoyo

Universidad de las Americas en Puebla,
Sta. Catarina Martir, Cholula, Puebla, 72820 Mexico
osoriomauri@gmail.com, juana.diaz@udlap.mx, alejandro1.santoyo@gmail.com
`http://www.udlap.mx`

Abstract. Dung's abstract argumentation frameworks has been object of intense study not just for its relationship with logical reasoning but also for its uses within artificial intelligence. One research branch in abstract argumentation has focused on finding new methods for computing its different semantics. We present a novel method, to the best of our knowledge, for computing semi-stable semantics using 0-1 integer programming, and also experimentally compare it with an answer set programming approach. Our results indicate that this new method performed well, and it has a great opportunity space for improving.

Keywords: Argumentation Frameworks, Binary Programming, Answer Set Programming, Semi Stable Semantics

1 Introduction

The main purpose of argumentation theory is to study the fundamental mechanism humans use in argumentation and to explore ways to implement this mechanism on computers. Currently formal argumentation research has been strongly influenced by abstract argumentation theory of Dung [6]. This approach is mainly orientated to manage the interaction of arguments by introducing a single structure called Argumentation Framework (AF). An AF basically is a pair of sets: a set of arguments and a set of disagreements between arguments called attacks. Indeed an AF can be regarded as a directed graph in which the arguments are represented by nodes and the attack relations are represented by arcs.

In [6], four argumentation semantics were introduced: grounded, preferred, stable, and complete semantics. The central notion of Dung's semantics is the acceptability of the arguments. Even though each of these argumentation semantics represents different patterns of selection of arguments, all of them are based on the basic concept of admissible set. Informally speaking, an admissible set presents a coherent and defendable point of view in a conflict between arguments.

One research branch in abstract argumentation has been to find new methods for computing its different semantics, *i.e.* the search for acceptable (*w.r.t.*

certain criteria) sets of arguments. Charwat *et al.* [10] surveys the approaches that has been used so far for computing AF semantics, and divided them into reduction and direct approaches. The direct approach consists in developing new algorithms for computing AF semantics, but our interest is in the reduction approach.

The reduction approach consists in using the software that was originally developed for other formalisms [10]. Thus, a given AF has to be formalized in the targeted formalism, like: constraint-satisfaction [5], propositional logic [1], or answer-set programming [3], [13].

To the best of our knowledge, there is just a previous work [11] where authors indirectly used 0-1 integer programming for computing preferred semantics, since their approach was based on a mapping from an argumentation framework AF into a logic program with negation as failure Π_{AF} to Clark's completion $Comp(\Pi_{AF})$ [12] to 0-1 integer program $lc(\Pi_{AF})$ [2], which then was solved by a mathematical programming solver.

In this work, we present a novel method, to the best of our knowledge, for directly computing semi-stable (SS) semantics using 0-1 integer programming with no mapping. Our results indicate that this new method performed well.

The paper is organized as follows. Section 2 gives some background on argumentation. Section 3 presents a procedure based on solving a series of 0-1 integer programming problems for computing SS semantics. Finally, Section 4 presents a brief description of results, some conclusions and future work.

2 Background

For space reasons, we assume that readers are familiar with the basic notions of 0-1 integer programming and ASP. We will use some concepts of Dung's argumentation approach. An AF captures the relationships between arguments.

Definition 1. *[6] An AF is a pair $AF := \langle AR, attacks \rangle$, where* AR *is a finite set of arguments, and* attacks *is a binary relation on* AR*,* i.e. attacks $\subseteq AR \times AR$*.*

We say that *a attacks b* (or b is attacked by a) if $attacks(a, b)$ holds. Similarly, we say that a set S of arguments attacks b (or b is attacked by S) if b is attacked by an argument in S.

Dung defined his argumentation semantics based on the basic concept of *admissible set*, which can be understood in terms of *defense* of arguments and in terms of *conflict-free* sets, as follows:

Definition 2. *[4] Let $AF := \langle AR, attacks \rangle$ be an argumentation framework, $A \in AR$ and $S \subseteq AR$, then:*

1. *A^+ as $\{B \in AR \mid A\ attacks\ B\}$ and*
2. *S^+ as $\{B \in AR \mid A\ attacks\ B\ for\ some\ A \in S\}$.*
3. *A^- as $\{B \in AR \mid B\ attacks\ A\}$ and*
4. *S^- as $\{B \in AR \mid B\ attacks\ A\ for\ some\ A \in S\}$.*
5. *S is conflict-free iff $S \cap S^+ = \emptyset$.*
6. *S defends an argument A iff $A^- \subseteq S^+$.*

7. $F : 2^{AR} \to 2^{AR}$ as $F(S) = \{A \in AR \mid A\ is\ defended\ by\ S\}$.

It is possible to define the semantics in terms of admissible sets as follows:

Definition 3. *[4] Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S \subseteq AR$ be a conflict-free set of arguments, then:*

1. *S is admissible iff $S \subseteq F(S)$.*
2. *S is a complete extension iff $S = F(S)$.*
3. *S is a preferred extension iff S is a maximal (w.r.t. set inclusion) complete extension.*

The SS semantics is similar to the preferred semantics [4], but instead of maximizing S it is required to maximize $S \cup S^+$, as states the following definition:

Definition 4. *[4] Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S \subseteq AR$ be a conflict-free set of arguments, then: S is called a SS extension iff S is a complete extension where $S \cup S^+$ is maximal*

The semi-stable semantics accepts an equivalente statements, as follows:

Proposition 1. *[4] Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and let $S \subseteq AR$. The following statements are equivalent: 1).- S is a complete extension such that $S \cup S^+$ is maximal (Definition No. 4), and 2).- S is an admissible set such that $S \cup S^+$ is maximal.*

3 Computing SS Semantics by 0-1 Integer Programming

In this section we show: the 0-1 integer programming formulation for the SS semantics problem, its encoding in FICO Xpress Mosel[1] language and also explain the objective function and constraints, as well as the iterative process for computing all the SS semantics' extensions.

3.1 Semi-Stable Semantics Problem Formulation

First of all, consider that it is required to have a mechanism to work with attacks more suitable than working with the adjacency matrix of a given AF, then we define:

Definition 5. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, then: $R_i^- = \{j \in AR : (j, i) \in attacks\}\ \forall i \in AR$, is the set of nodes attacking node i, and $R^- = \{R_i^- : i \in AR\}$.*

Considering Definitions 2, 3, and 5 we restate the admissible set definition in order to be able to derive the linear constraint that assures admissibility.

Definition 6. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, and set $S \subseteq AR$, then S is admissible iff $\forall i \in S, \forall j \in R_i^-, \exists k \in S,\ ((k, j) \in attacks))$.*

[1] `http://www.fico.com/en/wp-content/secure_upload/Xpress-Mosel-User-Guide.pdf`

Now consider that decision variables in a mathematical program are a set of quantities that need to be determined by the solver in order to solve the problem, *i.e.* when the best values of the decision variables have been identified in order to maximize or minimize (optimal solution) the objective function. Therefore, the first step is to define the required binary decision variables.

Considering that Definition 4 and Proposition 1 state a SS extension in terms of an *admissible* set S such that $S \cup S^+$ is maximal, then it is required a decision variable for S and other for S^+, as follows:

$$S_i = \begin{cases} 0 & \text{if } i \notin S \\ 1 & \text{if } i \in S \end{cases} \qquad \forall\, i \in AR \tag{1}$$

Definition 7. *The set $M = \{i \in AR : S_i = 1$ in the optimal solution$\}$, and $C = \{i \in AR : S_i = 0$ in the optimal solution$\}$ is M's complement.*

Accordingly, we define the decision variable for S^+, as follows:

$$S_i^+ = \begin{cases} 0 & \text{if } i \notin S^+ \\ 1 & \text{if } i \in S^+ \end{cases} \qquad \forall\, i \in AR \tag{2}$$

This way, the optimal solution to the 0-1 integer problem formulation for the SS semantics can be stated in terms of maximizing $S \cup S^+$.

Definition 8. *The set $M^+ = \{i \in AR : S_i^+ = 1$ in the optimal solution$\}$, and $C^+ = \{i \in AR : S_i^+ = 0$ in the optimal solution$\}$ is M^+' complement.*

It is required to have a decision variable for the union of this sets, as follows:

$$U_i = \begin{cases} 0 & \text{if } i \notin S \cup S^+ \\ 1 & \text{if } i \in S \cup S^+ \end{cases} \qquad \forall i \in AR \tag{3}$$

Definition 9. *The set $Mu = \{i \in AR : U_i = 1$ in the optimal solution$\}$, and $Cu = \{i \in AR : U_i = 0$ in the optimal solution$\}$ is Mu's complement.*

If the 0-1 program formulation with a given AF has an optimal solution, then we can think of the SS semantics in terms of a serie of solutions that the model finds in each iteration, as follows:

$$\{Mu^1, Mu^2, ..., Mu^q\} : |Mu^1| \geq |Mu^2| \geq ... \geq |Mu^q|$$

which states that the Mu^1 has the largest possible cardinality, and that $|Mu^1| \geq |Mu^2| \geq ... \geq |Mu^q|$ such that $|Mu^q|$ has the smallest possible cardinality.

Thus, It is also required a decision variable for a couple of either/or constraints [9] which we will add per each iteration (see section 3.3) in order to help to assure $S + S^+$ maximality *w.r.t.* set inclusion, *i.e.* they help us just to avoid that solution Mu^{t+1} be different than solution found in iteration $t, t-1, \ldots, 1$ or any subset of them, as follows:

$$Y_r = \begin{cases} 1 & \text{if } |Mu^t| > |Mu^{t+1} \\ 0 & \text{otherwise} \end{cases} \qquad r = 1, \ldots, t-1 \tag{4}$$

Taking into account that a mathematical solver searches in the solution space for one optimal solution for a given problem formulation, then if we want all the extensions of a given AF, it is required to solve a series of binary programming models, one for each extension. Since an AF semantics is made up of several sets, we use Mu^t to denote the solution of the binary subproblem in iteration t, and q to denote the amount of preferred extensions that a given AF has, such that $q \geq 1$ and $q \in \mathbb{N}$. The 0-1 integer problem formulation to compute the t^{th} semi-stable extension of a given AF is the following, including (1), (2), (3), (4):

$$max\ f(S) = \sum_{i \in AR} (S_i + S_i^+) \tag{5}$$

Subject to:

$$
\begin{aligned}
& S_i + S_j \leq 1 && \forall i \in AR, \forall j \in R_i^- && (6)\\
& \sum_{k \in R_j^-} S_k \geq S_i && \forall i \in AR, \forall j \in R_i^- && (7)\\
& S_i^+ \geq S_j && \forall i \in AR, \forall j \in R_i^- && (8)\\
& S_i^+ \leq \sum_{j \in R_i} S_j && \forall i \in AR && (9)\\
& U_i = S_i + S_i^+ && \forall i \in AR && (10)\\
& \sum_{i \in C^r} S_i \geq 1 && \forall r = 1, \ldots, t-1 && (11)\\
& -(\sum_{i \in Mu^r} U_i) + |Mu^r| \leq |AR| * Y_r && \forall r = 1, \ldots, t-1 && (12)\\
& -(\sum_{i \in Cu^r} U_i) + 1 \leq |AR| * (1 - Y_r) && \forall r = 1, \ldots, t-1 && (13)\\
& S_i \in \{0,1\} && i = 1, ..., |AR| && (14)\\
& S_i^+ \in \{0,1\} && i = 1, ..., |AR| && (15)\\
& U_i \in \{0,1\} && i = 1, ..., |AR| && (16)\\
& Y_i \in \{0,1\} && i = 1, ..., t-1 && (17)
\end{aligned}
$$

In (1), (2), (3) and (4) we have the decision variables definition and constraints (14), (15), (16), (17) define the problem's domain. The following paragraphs explain each constraint of the SS semantics problem formulation:
Maximality with regard to set inclusion. The model's objective function (OF)(5) guarantees us that we will find a *maximum cardinality set*, which will be the solution Mu^t. This set is made up of $S \cup S^+$. Constraints (11), (12), and (13) along with the objective function will avoid that M^{t+1} and Mu^{t+1} be any subset of M^t and Mu^t respectively, thus they guarantee us *maximality with*

regard to set inclusion. Subsection 3.3 explains how constraints (12) and (13) were defined and why they are added after the computation of each additional extension in order to compute the whole semantics.
Conflict-Freeness Note that the definition of a conflict-free set in Definition 2 item 5 is not stated in terms of attacks's directions but just in terms of attacks between arguments, without considering the directions of them. In this way, such a definition is considering an arc just as an edge, and therefore the whole AF can be regarded as an undirected graph, at least with regard to the conflict-free set problem.

Note that the expression $S_i + S_j \leq 1$, in constraint (6), will be just fulfilled when $S_i = 1$ or $S_j = 1$ but not both and when $S_i = 0$ and $S_j = 0$, therefore at most one arguments will be selected which guarantees us that solution will be a *conflict-free set.*
Admissibility. The intuition of Definitions 1, 2 and 3 is that an *admissible set* S should defend each of its arguments, and Definition 6 just restates it in terms of Definition 5. Note that in Definition 6 the existential quantifier suggests that constraint (7) should be $\sum_{k \in R_j^-} S_k \geq 1$, but we used $\sum_{k \in R_j^-} S_k \geq S_i$ since the constraint must be fulfilled $\forall i \in AR$ [2]. This way, the translation from this definition to constraint (7) is a straightforward task, which guarantees us that the set M^t is admissible.
Creation of S^+***.*** So far, we have a conflict-free and admissible set, and it is still required to build S_i^+ as in Definition 2 item 2 $\forall i \in M^+$. It should be done by decision variables (2) such that the objective function can be executed. This way, S_i^+ should take on value 1 if argument i is attacked by some argument $j \in R_i^-$ such that $S_j = 1$. This is the same that $d = d_i \vee d_2 \ldots \vee d_n$ as a logical expression which can be linearized as follows [9]:

$$d \geq d_i \qquad i = 1, \ldots, n \tag{18}$$

$$d \leq \sum_i d_i \qquad i = 1, \ldots, n \tag{19}$$

$$d \leq 1 \tag{20}$$

Note that (18) and (19) become (8) and (9) respectively while (20) is redundant due to (15). Thus, constraints (8), (9) and (15) will determine the values of decision variables (2) from values on decision variables (1).

Thus, M^1 is a conflict-free and admissible set, considering that (5) guarantees a *maximum cardinality set*, and according to Definition 4 and Preposition 1, M^1 is a SS extension.
Construction of $U = S + S^+$***.*** In order to avoid that Mu^{t+1} be a subset of Mu^t, it is required to have as decision variables the union of (1) and (2), as defined in (3). To this end, and in order to ease this process, consider that it is not possible that $S_i = 1$ and $S_i^+ = 1$ at the same time, since it would mean

[2] There are two special cases: $S_i = 0$ and $S_i = 1$. In the first one, it does not matter the total of $\sum_{k \in R_j^-} S_k$ because $S_i \notin Solution$, thus in the second case we will have $\sum_{k \in R_j^-} S_k \geq 1$ which means that there will be at least one argument defending argument i since $S_i \in Solution$.

that M is not a conflict-free set. Thus, the value that U_i will take on should be $S_i + S_i^+$. Considering Definition 9, constraint (10) guarantees that Mu will have the whole solution.

3.2 Semi Stable Extensions Program

Notice that the problem formulation made up of (1)-(10), and (14)-(17) already can be used for computing the first SS extension of a given AF. To this end, this program should be coded using a mathematical programming language like *mosel* [3], which is a straightforward task, since the mathematical language was developed for expressing mathematical formulas, and even though we can not show the complete code for space reasons, the following code stands for the whole mathematical model:

```
z:= sum(i in arguments) (S(i) + Sp(i))
forall(i in arguments, j in R(i)) S(i) + S(j) <= 1
forall(i in arguments, j in R(i)) sum(k in R(j)) S(k) >= S(i)
forall(i in arguments, j in R(i)) Sp(i) >= S(j)
forall(i in arguments) Sp(i) <= sum(j in R(i)) S(j)
forall(i in arguments) U(i) = S(i) + Sp(i)
forall(i in arguments) S(i)  is_binary
forall(i in arguments) Sp(i) is_binary
forall(i in arguments) U(i)  is_binary
forall(i in t-1)   Y(i)  is_binary
maximize(z)
```

We will denote this program as BIP in order to make reference to it.

3.3 Semi Stable Semantics

Note that once the model is implemented in a mathematical programming language, the program just compute one SS extension. In order to compute an additional extension it is required to solve the model again, but adding additional constraints to avoid getting previous solutions, which will force to get another different extension. In this setting, it is required to iterate to find all the extensions of a given AF until there is no a feasible solution. Thus, we have to take care of getting no subsets of M^t (Case No. 1), and no proper subsets of Mu^t (Case No. 2).

Case No. 1. Then, in order to find the constraint that we have to add to avoid that $M^{t+1} \subseteq M^t$, consider Definitions 7, and let P be the solution in iteration $t+1$, and M the solution in iteration t, thus:

$$P \subseteq M \leftrightarrow \forall x (x \in P \rightarrow x \in M) \leftrightarrow \forall x (x \notin P \vee x \in M) \tag{21}$$

$$P \not\subseteq M \leftrightarrow \exists x (x \in P \wedge x \notin M) \leftrightarrow \exists x (x \in P \wedge x \in C) \tag{22}$$

[3] http://www.fico.com/en/products/fico-xpress-optimization-suite/

The intuition of this result is that it is required that solution in iteration $t+1$ has at least one element from solution's complement in iteration t. Constraint (11) is defined from this intuition. Note that the mosel code must take care of the special case where $|M| = |AR|$.

Case No. 2. Additionally, we have to add other constraint to avoid that $Mu^{t+1} \subset Mu^t$. In order to find such a constraint(s), consider Definition 9 and let P be the solution in iteration $t+1$, and Mu the solution in iteration t, thus:

$$\begin{aligned} P \subset Mu &\leftrightarrow \forall x(x \in P \rightarrow x \in Mu) \wedge \exists x(x \notin P \wedge x \in Mu) && (23)\\ &\leftrightarrow \forall x(x \notin P \vee x \in Mu) \wedge \exists x(x \notin P \wedge x \in Mu) && (24)\\ P \not\subset Mu &\leftrightarrow \exists x(x \in P \wedge x \notin Mu) \vee \forall x(x \in P \vee x \notin Mu) && (25)\\ &\leftrightarrow \exists x(x \in P \wedge x \in Cu) \vee \forall x(x \in P \vee x \notin Mu) && (26) \end{aligned}$$

Note that the intuition of the expression in (22) should be the same as that of the first part of the disjunction in (26). Consider also that we wanted to find an expression that led us to a linearized constraint to avoid getting proper subsets of previous solutions. Thus, we can have a proper subset when $|P| < |Mu|$ holds, and therefore apply the expression $\exists x(x \in P \wedge x \in Cu)$, otherwise apply the expression $\forall x(x \in P \vee x \notin Mu)$, whose intuition is that the new solution P can have any element, which means that it requires no constraint. Thus, we have just to work with the first part of the disjunction. Therefore, we have to linearize the expression [9]:

$$\begin{aligned} if\ |P| < |Mu|\ then \sum_{i \in Cu} U_i \geq 1 &\leftrightarrow not\ (|P| < |Mu|)\ \vee \sum_{i \in Cu} U_i \geq 1\\ &\leftrightarrow |P| \geq |Mu| \vee \sum_{i \in Cu} U_i \geq 1\\ &\leftrightarrow \sum_{i \in Mu^r} U_i \geq |Mu^r| \vee \sum_{i \in Cu^r} U_i \geq 1 \end{aligned}$$

which means that the model should apply either $\sum_{i \in Mu^r} U_i \geq |Mu^r|$ or $\sum_{i \in Cu} U_i \geq 1$, but to satisfy the simultaneousness assumption of binary integer programming, they must be transformed considering the following general format [9]:

$$f(x_1, x_2, \ldots, x_n) \leq By \qquad g(x_1, x_2, \ldots, x_n) \leq B(1-y)$$

where B is a big number, in our case $B = |AR|$, and y is the binary variables defined in (17). This transformation becomes constraints (12) and (13).

Now, let SSE a set of all the SS extensions of a given AF, and MC a set of additional (11), (12), and (13) constraints, then the algorithm for computing the q extensions of a given AF is the following:

```
1  Set $SSE = \emptyset$, $MC = \emptyset$;
2  Solve $BIP \cup MC$;
3  while optimal solution found do
4      Let $M, M^+, and\ Mu$ the optimal solution, and $C, C^+, and\ Cu$ its
       complements respectively.;
5      Add M to SSE;
6      Add $\sum_{i \in C} S_i \geq 1$ to $MC$;
7      Add $-(\sum_{i \in Mu^r} U_i) + |Mu^r| \leq |AR| * Y(r) \forall r = 1, \ldots, t-1$ to $MC$;
8      Add $-(\sum_{i \in Cu^r} U_i) + 1 \leq |AR| * (1 - Y(r)) \forall r = 1, \ldots, t-1$ to $MC$;
9      Solve $BIP \cup MC$;
10 end
```

Algorithm 1: For Computing all Semi-Stable Extensions of a Given AF

Now it is possible to state the following theorem:

Theorem 1. *Let AF be an argumentation framework, R^- as defined in 5, $M, M^+, and\ Mu$ is a solution of BIP, and SSE is computed as described in Algorithm No. 1, then SSE is the set of all SS extensions of AF.*

Proof. Sketch: From the above discussion consider the following items:

1. *By OF* (5) *we know that $M \cup M^+$ is a maximum cardinality set.*
2. *By constraints*(6), (7) *we know that M is a conflict-free and admissible set.*
3. *By constraints* (8) *and* (9) *we know that M^+ is made up from M.*
4. *By constraint* (10) *we know that $Mu = M \cup M+$.*
5. *By Definition 4 and Proposition 1 we know that if a set M is a conflict-free and admissible set, and $M \cup M^+$ is maximal then M is a SS extension.*
6. *Now, notice that in each iteration, due to the constraint added to MC in steps 6, 7 and 8 in iteration t, the solution Mu (if exists), obtained in step 4 must not be a superset or subset of any previous solutions already in SSE, and Mu must be of maximum cardinality among the solutions that satisfy $BIP \cup MC$, therefore Mu is maximal w.r.t. set inclusion.*

In order to measure the performance of the 0-1 integer program, it was compared with an ASP approach: the ASPARTIX [4] [8] which we will call just ASP1, and we used the solver Clingo[5]. Additionally, the approach based on 0-1 integer programming will be called BIP, and we used the *ad-hoc* Xpress[6] solver. The instances that were used during all the experiments were taken from the ASPARTIX web page [7].

As a summary, BIP approach had a performance quit similar to the ASP approach for arbitrary instances[8], but had problems until 60-arguments instances for 4-grid and 8-grid instances. This result shows that the BIP approach performed well and it has a great opportunity space for improving.

[4] The program code is available at ASPARTIX's web page. It is worth mentioning that ASPARTIX is the *de facto* benchmark for argumentations systems

[5] `http://potassco.sourceforge.net`

[6] http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx

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

[8] There is a complete explanation of each kind of instance in [7]

4 Conclusions

We have presented a novel method for computing SS semantics using binary integer programming, the performance was good although the ASP approach outperformed it.

However, it is well known that binary integer programs can be improved, in order to compute more efficiently its objective function, by using mathematical programming techniques such as relaxation or adding strong valid inequalities. It means that it is possible to improve the performance of this novel approach for computing SS semantics.

This new approach constitutes an alternative for computing AF semantics using mathematical programming, and even though we used an state of the art mathematical programming solver, there exists several libraries for java, C++ and other general purpose languages.

References

1. Handbook of satisfiability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.
2. Colin Bell, Anil Nerode, Raymond T. Ng, and V. S. Subrahmanian. Mixed integer programming methods for computing nonmonotonic deductive databases. *Journal of the ACM*, 41(6):1178–1215, 1994.
3. Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.
4. Martin W. A. Caminada, Walter Alexandre Carnielli, and Paul E. Dunne. Semi-stable semantics. *J. Log. Comput.*, 22(5):1207–1254, 2012.
5. Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., 2003.
6. Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.
7. Wolfgang Dvorák, Sarah Alice Gaggl, Johannes Peter Wallner, and Stefan Woltran. Making use of advances in answer-set programming for abstract argumentation systems. *CoRR*, abs/1108.4942, 2011.
8. Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Aspartix: Implementing argumentation frameworks using answer-set programming. In M. Garcia de la Banda and E. Pontelli, editors, *International Conference of Logic Programming (ICLP)*, volume 5366 of *Lecture Notes of Computer Science*, pages 734–738. Springer, 2008.
9. FICO. *MIP Formulations and Linearizations*. Fair Isaac Corporation, 2009.
10. Sarah A. Gaggl Johannes P. Wallner Stefan Wolfran Gunter Charwat, Wolfgang Dvorak. Implementing abstract argumentation: A survey. Technical report, Institut Fur Information Systeme, 2013.
11. Alejandro Santoyo Mauricio Osorio. Preferred extensions as minimal models of clark's completion semantics, 2013.
12. Juan Carlos Nieves, Mauricio Osorio, and Ulises Cortés. Preferred Extensions as Stable Models. *Theory and Practice of Logic Programming*, 8(4):527–543, July 2008.
13. Francesca Toni and Marek Sergot. Argumentation and answer set programming. In Marcello Balduccini and Tran Cao Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, pages 164–180. Springer-Verlag, Berlin, Heidelberg, 2011.
14. Laurence A. Wolsey. *Integer Programming*. Discrte Mathematics and Optimization. John Wiley & Sons, Inc., 1998.