Finding Partitions of Arguments with Dung's Properties via SCSPs

Stefano Bistarelli[1,2], Paola Campli[3], and Francesco Santini[1]

[1] Dipartimento di Matematica e Informatica, Università di Perugia, Italy
[bista,francesco.santini]@dmi.unipg.it
[2] Istituto di Informatica e Telematica (CNR), Pisa, Italy
[stefano.bistarelli]@iit.cnr.it
[3] Dipartimento di Scienze, Università G.d'Annunzio di Chieti-Pescara, Italy
campli@sci.unich.it

Abstract. Forming coalition structures allows agents to join their forces to achieve a common task. We suggest it would be interesting to look for homogeneous groups which follow distinct *lines of thought*. For this reason, we extend the Dung Argumentation Framework in order to deal with coalitions of arguments. The initial set of arguments is partitioned into subsets (or coalitions). Each coalition represents a different line of thought, but all the found coalitions show the same property inherited by Dung, e.g. all the coalitions in the partition are admissible (or conflict-free, complete, stable). Some problems in weighted argumentation are NP complete; we use (soft) constraints as a formal approach to reason about coalitions and to model all these problems in the same framework. Semiring algebraic structures can be used to model different optimization criteria for the obtained coalitions. To implement this mapping and practically find its solutions we use JaCoP, a Java constraint solver, and we test the code over a small-world network.

1 Introduction and Motivations

A coalition structure is a temporary alliance or partnering of groups in order to achieve a common purpose. Forming coalitions with other members of similar values, interests and goals, allow agents to combine their resources and become more powerful than when they each acted alone [12]. To form a successful coalition, the recognition of compatible interests and common *lines of thought* is needed, since the goal of different agents can be shared by multiple parties.

The abstract nature of Dung's seminal theory [9] of argumentation accounts for its widespread application for various species of non-monotonic reasoning. A Dung argumentation framework (see Sect. 2) is classically instantiated by arguments and a binary conflict based attack relation, defined by some underlying logical theory. The justified arguments under different extensional semantics (e.g. conflict-free ones) are then evaluated, and the claims of these arguments define the inferences of the underlying theory. The aim of this paper is to partition a set of arguments into coalition structures of arguments [8, 1, 6]. A classical scenario

could be represented by the need to aggregate a set of distinct arguments into different lines of thought. Suppose, for example, to have some statements belonging to candidates of different political parties; it would be interesting to check how consistent their ideas are. For example, "We do not want immigrants with the right to vote" is clearly closer to "Immigration must be stopped", than to "We need a multicultural and open society in order to enrich the life of everyone and boost our economy". In general, cooperating groups, referred to as coalition structures [16], have been thoroughly investigated in AI and Game Theory and have proved to be useful in both real-world economic scenarios and Multi-agent Systems [16, 19, 2]. The basic idea behind this work is to start from a single set of arguments and partition them to several agents, with the condition that each subset has to show the same properties defined by Dung, e.g. admissibility [9]. Some applications might be task allocation problem (let tasks be the agents), sensor network problems (agents must form groups), distributed winner determination in combinatorial auctions, agents grouping to handle work-flows (just-in-time incorporation) [16, 19, 2]. In order to model and solve the proposed extended problems we use *(Soft) Constraint Programming* (*(S)CP*) [18] (see Sect. 3), which is a powerful paradigm for solving combinatorial problems that draws on a wide range of techniques from AI, Databases, Programming Languages, and Operations Research [18]. The idea of the semiring-based constraint formalism presented in [4, 3] was to further extend the classical constraint notion by adding the concept of a structure representing the levels of satisfiability of the constraints. Such a structure is similar to a semiring (see Sec. 3). Problems defined according to the semiring-based framework are called *Soft Constraint Satisfaction Problems* (SCSPs) [4, 3, 18]. There already exist many efficient techniques, as constraint propagation [18], to solve such complex problems. The solution of the obtained SCSP represents the partition of the arguments (see Sec. 4) where each subset (i.e. coalition) of arguments has the same property originally defined by Dung in [9], e.g. each coalition in the partition is admissible. Semirings can be used to relax conflict-free partitions, by allowing a certain degree of conflicts inside the coalitions, by representing a weight (or preference) associated with each attack between arguments (see Sec. 5 - 6). At last (in Sec. 7), we show an implementation of a crisp CSP (equivalent to use a *Boolean* semiring in SCSPs) with the *Java Constraint Programming* solver (*JaCoP*) [15] and we test it over a small-world network randomly generated with the *Java Universal Network/Graph Framework* (*JUNG*) [17].

2 Dung Argumentation

Dung proposed an abstract framework for argumentation in which he focuses on the definition of the status (*attacked* / *defended*) of arguments [9]. It can be assumed that a set of arguments and the different conflicts among them are given.

Definition 1 ([9]). *An Argumentation Framework (AF) is a pair* $\langle \mathcal{A}, R \rangle$ *of a set* $\mathcal{A}$ *of arguments and a binary relation* R *on* $\mathcal{A}$ *called the attack relation.*

$\forall a_i, a_j \in \mathcal{A}$, $a_i \, R \, a_j$ *means that* a_i *attacks* a_j. *An AF may be represented by a directed graph (the interaction graph) whose nodes are arguments and edges represent the attack relation. A set of arguments* $\mathcal{B}$ *attacks an argument* a *if* a *is attacked by an argument of* $\mathcal{B}$. *A set of arguments* $\mathcal{B}$ *attacks a set of arguments* $\mathcal{C}$ *if there is an argument* $b \in \mathcal{B}$ *which attacks an argument* $c \in \mathcal{C}$.

Fig. 1: A classical AF using weather forecast; e.g. b attacks c and viceversa.

In Fig. 1 we show an example of AF represented as an *interaction graph*. Dung [9] gave several semantics of "acceptability", which produce none, one or several acceptable sets of arguments, called extensions. The stable semantics is only defined via the notion of attacks:

Definition 2 ([9]). *A set* $\mathcal{B} \subseteq \mathcal{A}$ *is conflict-free iff for no two arguments* a *and* b *in* $\mathcal{B}$, a *attacks* b. *A conflict-free set* $\mathcal{B} \subseteq \mathcal{A}$ *is a stable extension iff each argument not in* $\mathcal{B}$ *is attacked by an argument in* $\mathcal{B}$.

The other semantics for "acceptability" rely upon the concept of defense. An admissible set of arguments according to Dung must be a conflict-free set which defends all its elements. Formally:

Definition 3 ([9]). *An argument* b *is defended by a set* $\mathcal{B} \subseteq \mathcal{A}$ *(or* $\mathcal{B}$ *defends* b*) iff for any argument* $a \in \mathcal{A}$, *if* a *attacks* b *then* $\mathcal{B}$ *attacks* a. *A conflict-free set* $\mathcal{B} \subseteq \mathcal{A}$ *is admissible iff each argument in* $\mathcal{B}$ *is defended by* $\mathcal{B}$

Besides the stable semantics, one semantics refining admissibility has been introduced by Dung [9].

Definition 4 ([9]). *An admissible* $\mathcal{B} \subseteq \mathcal{A}$ *is a complete extension iff each argument which is defended by* $\mathcal{B}$ *is in* $\mathcal{B}$.

In Fig. 2 we show an example of a stable (A), admissible (B) but not complete (due to x_6) and complete (C) extension.

3 Semirings and Soft Constraints

A semiring [4, 3] S is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ where A is a set with two special elements $\mathbf{0}, \mathbf{1} \in A$ (respectively the bottom and top elements of A) and with two operations $+$ and $\times$ that satisfy certain properties: $+$ is defined over (possibly infinite) sets of elements of A and is commutative, associative and idempotent;

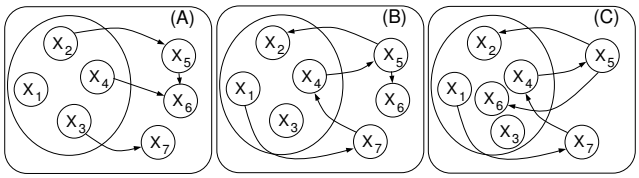

Fig. 2: A stable (A), an admissible (B) and a complete (C) extension (clearly also conflict-free).

it is closed, **0** is its unit element and **1** is its absorbing element; $\times$ is closed, associative, commutative and distributes over $+$, **1** is its unit element and **0** is its absorbing element (for the exhaustive definition, please refer to [4]). The $+$ operation defines a partial order $\leq_S$ over A such that $a \leq_S b$ iff $a+b=b$; we say that $a \leq_S b$ if b represents a value *better* than a. Moreover, $+$ and $\times$ are monotone on $\leq_S$, **0** is its min and **1** its max, $\langle A, \leq_S \rangle$ is a complete lattice and $+$ is its lub. A *soft constraint* [4, 3] may be seen as a constraint where each instantiation of its variables has an associated preference. Given $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and an ordered set of variables V over a finite domain D, a soft constraint is a function which, given an assignment $\eta : V \to D$ of the variables, returns a value of the semiring. Using this notation $\mathcal{C} = \eta \to A$ is the set of all possible constraints that can be built starting from S, D and V. Any function in $\mathcal{C}$ depends on the assignment of only a finite subset of V. For instance, a binary constraint $c_{x,y}$ over variables x and y, is a function $c_{x,y} : V \to D \to A$, but it depends only on the assignment of variables $\{x, y\} \subseteq V$ (the *support*, or *scope*, of the constraint). Note that $c\eta[v := d_1]$ means $c\eta'$ where η' is η modified with the assignment $v := d_1$. Notice that $c\eta$ is the application of a constraint function $c : V \to D \to A$ to a function $\eta : V \to D$; what we obtain is a semiring value $c\eta = a$. $\bar{a}$ represents the constraint functions associating a to all assignments of domain values. Given the set $\mathcal{C}$, the combination function $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is defined as $(c_1 \otimes c_2)\eta = c_1\eta \times c_2\eta$ [4, 3]. The $\otimes$ builds a new constraint which associates with each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associated by the original constraints to the appropriate sub-tuples. Given a constraint $c \in \mathcal{C}$ and a variable $v \in V$, the *projection* [4, 3] of c over $V - \{v\}$, written $c \Downarrow_{(V \setminus \{v\})}$ is the constraint c' such that $c'\eta = \sum_{d \in D} c\eta[v := d]$. Informally, projecting means eliminating some variables from the support.

An SCSP [3] is defined as $P = \langle C \rangle$ where C is the set of constraints. The *best level of consistency* notion defined as $blevel(P) = Sol(P) \Downarrow_{\emptyset}$, where $Sol(P) = \bigotimes C$ [3]. A problem P is α-consistent if $blevel(P) = \alpha$ [3]; P is instead simply "consistent" iff there exists $\alpha >_S \mathbf{0}$ such that P is α-consistent [3]. P is inconsistent if it is not consistent.

An SCSP Example. Figure 3 shows a weighted SCSP as a graph: the *Weighted* semiring is used, i.e. $\langle \mathbb{R}^+ \cup \infty, \min, \hat{+}, \infty, 0 \rangle$ ($\hat{+}$ is the arithmetic plus operation). Variables and constraints are represented respectively by nodes and arcs (unary for c_1 and c_3, and binary for c_2), and semiring values are written to the right of each tuple, $D = \{a, b\}$. The solution of the CSP in Fig. 3 associates

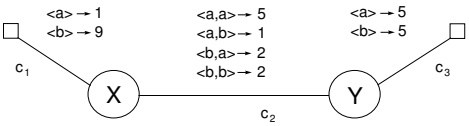

Fig. 3: An SCSP based on a Weighted semiring.

a semiring element to every domain value of variables X and Y by combining all the constraints together, i.e. $Sol(P) = \bigotimes C$. For instance, for the tuple $\langle a, a\rangle$ (that is, $X = Y = a$), we have to compute the sum of 1 (which is the value assigned to $X = a$ in constraint c_1), 5 (which is the value assigned to $\langle X = a, Y = a\rangle$ in c_2) and 5 (which is the value for $Y = a$ in c_3). Hence, the resulting value for this tuple is 11. For the other tuples, $\langle a, b\rangle \rightarrow 7$, $\langle b, a\rangle \rightarrow 16$ and $\langle b, b\rangle \rightarrow 16$. The *blevel* for the example in Fig. 3 is 7, related to the solution $X = a$, $Y = b$.

4 Extending Dung Argumentation to Coalitions

Given the set of arguments $\mathcal{A}$, the problem of coalition formation consists in selecting an appropriate partition of $\mathcal{A}$, $\mathcal{G} = \{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$ ($|\mathcal{G}| = |\mathcal{A}|$ if each argument forms a coalition on its own), such that $\bigcup_{B_i \in G} B_i = \mathcal{A}$ and $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$, if $i \neq j$; clearly, $\forall i.\mathcal{B}_i \neq \emptyset$. In this section we extend Dung's semantics (see Sec. 2) in order to deal with a partition of arguments, that is, we cluster the arguments into different subsets representing *distinct lines of thought*. An example representing the difference between the original framework [9] and our extension is illustrated in Fig. 4: Fig. 4 (A) represents a conflict-free extension as described in Def. 3, while Fig. 4 (B) represents a conflict-free partition of coalitions, since each coalition is conflict-free (see Def. 5). Thus, while in Dung it is sufficient to find only one set with the conflict-free property, we want to find a set of conflict-free sets that represents a partition of the given arguments; we can compute partitions by considering the other properties as well, i.e. admissible, complete and stable semantics. Notice that, in general, we can have a combinatorial number of partitions for a given set of arguments [7, 16]. For example, instead of $P_1 = \{\{x_1, x_2, x_3\}, \{x_4, x_5, \}, \{x_6, x_7, x_8, x_9\}\}$ we can have $P_2 = \{\{x_1, x_2, x_3, x_4\}, \{x_5\}, \{x_6, x_7, x_8, x_9\}\}$. We can have 21147 different partitions for the 9 elements in Fig. 4 (B): this number is called *Bell Number* and is recursively computed as $B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$ [7] (with $B_0 = B_1 = 1$). Clearly, not all of these partition are (e.g.) conflict-free.

In the following, we extend the definitions given in Sec. 2 to deal with coalitions.

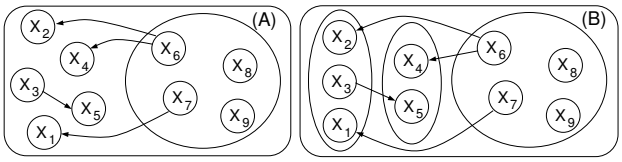

Fig. 4: Differences between classical Dung AF (A) and the extended partitioned framework (B).

Definition 5. *A partition of coalitions* $\mathcal{G} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ *is* ***conflict-free*** *iff for each* $\mathcal{B}_i \in \mathcal{G}$, $\mathcal{B}_i$ *is conflict-free, i.e.* $\forall a, b \in \mathcal{B}_i.(a, b) \notin R$*: no attacking arguments inside the same coalition.*

From the argumentation theory point of view, finding a conflict-free partition of coalitions corresponds to partitioning the arguments into coherent subsets, in order to find feasible lines of thought which do not internally attack themselves. Now we revise the concept of attack/defence among coalitions and arguments and the notion of stable partitions of coalitions:

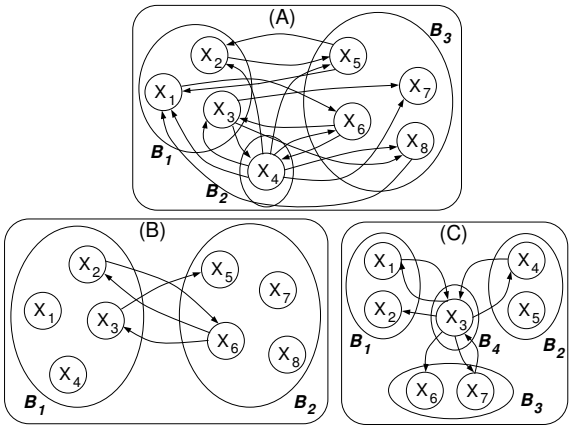

Fig. 5: A stable (A), an admissible and complete (B) and an admissible but not complete (C) partitions of coalitions.

Definition 6. *A coalition* $\mathcal{B}_i$ ***attacks*** *another coalition* $\mathcal{B}_j$ *if one of its elements attacks at least one element in* $\mathcal{B}_j$*, i.e.* $\exists a \in \mathcal{B}_i, b \in \mathcal{B}_j$ *s.t.* $a\,R\,b$*.* $\mathcal{B}_i$ ***defends*** *an attacked argument* a*, e.g.* $b\,R\,a$*, if* $\exists c \in \mathcal{B}_i$ *s.t.* $c\,R\,b$*.*

Definition 7. *A* conflict-free *partition* $\mathcal{G} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ *is* ***stable*** *iff for each coalition* $\mathcal{B}_i \in \mathcal{G}$*, all its elements* $a \in \mathcal{B}_i$ *are attacked by all the other coalitions* $\mathcal{B}_j$ *with* $j \neq i$*, i.e.* $\forall a \in \mathcal{B}_i, \exists b \in \mathcal{B}_j.b\,R\,a$ $(\forall j \neq i)$*.*

Fig. 5 (A) represents a stable partition: each argument in $\mathcal{B}_2$ (i.e. x_4) is attacked by at least one argument in $\mathcal{B}_1$ (i.e. x_3) and one argument in $\mathcal{B}_3$ (i.e. x_6), and the same also holds for the arguments in $\mathcal{B}_2$ and $\mathcal{B}_3$. To have a stable partition means that each of the arguments cannot be moved from one coalition to another without inducing a conflict in the new coalition. In the next two definitions we respectively extend the concept of admissible and complete extensions.

However different definitions for stable partitions can also be defined, for instance one could require that each argument has to be attacked by some (rather than all of the) other coalitions.

Definition 8. *A* conflict-free *partition* $\mathcal{G} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ *of coalitions is* ***admissible*** *iff for each argument* $a \in \mathcal{B}_i$ *attacked by* $b \in \mathcal{B}_j$ *(i.e.* $b\,R\,a$*),* $\exists c \in \mathcal{B}_i$ *that attacks* $b \in \mathcal{B}_j$ *(i.e.* $c\,R\,b$*), that is each* $\mathcal{B}_i$ *defends all its arguments.*

According to Dung's definition of admissible extension, *"the set of all arguments accepted by a rational agent is a set of arguments which can defend itself against all attacks on it"* [9]. Notice that if only one argument a in the interaction graph has no grandparents, it is not possible to obtain even one admissible partition: no argument in $\mathcal{A}$ is able to defend a. In Def. 8, we have naturally extended the definition of admissible extension [9] to coalitions: since each coalition represents the line of thought of an agent, each rational agent is able to defend its line of thought because it counter-attacks all its attacking lines.

Fig. 5 (B) represents an admissible partition as it is conflict-free and both $\mathcal{B}_1$ and $\mathcal{B}_2$ defend themselves: x_5 is defended by x_6 and for the attack performed by $x_6 \in \mathcal{B}_2$, x_2 and x_3 are defended by x_2.

Definition 9. *An admissible partition* $\mathcal{G} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ *is a* ***complete*** *partition of coalitions iff each argument* a *which is defended by* $\mathcal{B}_i$ *is in* $\mathcal{B}_i$ *(i.e.* $a \in \mathcal{B}_i$*).*

Fig. 5 (B) is a complete partition because all the elements defended by $\mathcal{B}_2$ (i.e. x_5, x_8) belong to $\mathcal{B}_2$ and all elements defended by $\mathcal{B}_1$ (x_2, x_3) belong to $\mathcal{B}_1$. Figure 5 (C) represents an admissible but not complete partition because x_6 is defended also by coalitions $\mathcal{B}_1$ (via x_1) and $\mathcal{B}_2$ (via x_4) but belongs to $\mathcal{B}_3$ (defending it via x_7). Intuitively, the notion of complete partition captures the rational agents who believe in every argument they can defend [9].

In Th. 1 we prove that each of the coalitions in every possible conflict-free partition is a conflict-free extension as defined by Dung [9]. Respectively, we can prove the same property for admissible, complete and stable partitions.

Theorem 1. *Given an AF* $\langle \mathcal{A}, R \rangle$ *as in Def. 1 and*

- *given the set of all* CFE *conflict-free extensions which can be obtained over an interaction graph by using Dung's semantics [9] (see also Sec. 2), each* CFP *conflict-free partition as defined in Def. 5 is a subset of them, i.e.* $CFP \subseteq CFE$*.*
- *given the set of all* AE *admissible extensions [9], each* AP *admissible partition as defined in Def. 8 is a subset of them, i.e.* $AP \subseteq AE$*.*

– *given the set of all CE complete extensions [9], each CP complete partition as defined in Def. 9 is a subset of them, i.e. $CP \subseteq CE$.*
– *given the set of all SE stable extensions [9], each SP stable partition as defined in Def. 7 is a subset of them, i.e. $SP \subseteq SE$.*

We can now define the hierarchy of the set inclusions among the proposed partitions like Dung has shown for set inclusions among classical extensions [9]:

Theorem 2. *Given the $CFPS$, APS, CPS and SPS respectively the set of all conflict-free, admissible, complete and stable partitions, we have that $SPS \subseteq CPS \subseteq AS \subseteq CFPS$.*

These two theorems can be proved by reasoning on the sets of classical extensions defined in [9]: the partitions, as defined in this paper, directly inherit their properties. Notice that since our aim is to find partitions and not classical extensions, it is possible that, given the same set of arguments, a stable (for example) extension exists, but a stable partition may not be possible. Let us consider the following example: $A = \{a, b, c, d, e\}$ and $R = \{(b, c), (c, d), (d, e), (e, b)\}$. According to Dung's stable semantics, this framework has two stable extensions: $\{a, b, d\}$ and $\{a, c, e\}$; however, it has no stable partition since the argument a is not attacked and it cannot be in two sets. Even if the situation in which more agents agree about an argument might be possible in several scenarios, we want an argument to be held by exactly one agent, that is, the one who first declared it. This is not a limitation, because our goal is to simultaneously form distinct stable extensions within the same set of arguments, which represent different lines of thought to be assigned to different agents. An application in the real world corresponds to the partitioning of arguments to find the difference among political parties. Indeed, even if an argument might be put forward by several political parties, it is necessary that this argument belongs only to one coalition.

5 Weighted Partitions

Weighted AFs extend Dung's AFs by adding weight values to every edge in the attack graph, intuitively corresponding to the strength of the attack, or equivalently, how reluctant we would be to disregard it [5, 10]. In this section we define a quantitative framework where attacks have an associated preference/weight and, consequently, also the computation of the coalitions as presented in this paper has an associated weight representing the level of inconsistency we tolerate in the solution: more specifically, "how much conflict" we tolerate inside a conflict-free partition, which can now include attacking arguments in the same coalition. Modeling this kind of problems as SCSPs (see Sec. 3) leads to a partition that optimizes the criteria defined by the chosen semiring, which is used to mathematically represent the attack weights.

Fig. 6 represents weighted attack relationships among arguments; in this example $\mathcal{A} = \{a, b, c\}$, $a\,R\,b$ and $c\,R\,b$, moreover, each of these two attack relationships is associated with a fuzzy weight (in $[0, 1]$) representing the strength

of the attack: a attacks b with more strength (i.e. 0.5) than c attacks b (i.e. 0.9). In this case 0 represents the strongest possible attack and 1 the weakest one.

Many other classical weighted AFs in literature can be modeled with semirings [5]. An argument can be seen as a chain of possible events that makes the hypothesis true. The credibility of a hypothesis can then be measured by the total probability that it is supported by arguments. The proper semiring to solve this problem consists in the *Probabilistic* semiring [3]: $\langle [0..1], max, \hat{\times}, 0, 1\rangle$, where the arithmetic multiplication (i.e. $\hat{\times}$) is used to compose the probability values together (assuming that the probabilities being composed are independent). The Fuzzy Argumentation [5] approach enriches the expressive power of the classical argumentation model by allowing to represent the relative strength of the attack relationships between arguments, as well as the degree to which arguments are accepted. In this case, the *Fuzzy* semiring $\langle [0..1], max, min, 0, 1\rangle$ can be used (e.g. in Fig. 6). In addition, the *Weighted* semiring $\langle \mathbb{R}^+ \cup \infty, min, \hat{+}, \infty, 0\rangle$, where $\hat{+}$ is the arithmetic plus ($\mathbf{0} = \infty$ and $\mathbf{1} = 0$), can model the (e.g. money) cost of the attack: for example, the number of votes in support of the attack [10]. By using the *Boolean* semiring $\langle \{true, false\}, \vee, \wedge, \ false, true\rangle$ we can cast the classic AF originally defined by Dung [9] in the same semiring-based framework ($\mathbf{0} = false, \mathbf{1} = true$). The implementation in Sec. 7 models the use of a *Boolean* semiring, since it adopts crisp constraints. Definition 10 rephrases the notion of AF given by Dung (see Sec. 2) into *semiring-based AF*, i.e. an AF_S:

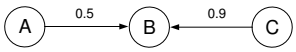

Fig. 6: A fuzzy Argumentation Framework with fuzzy scores modeling the attack strength.

Definition 10 ([5]). *A semiring-based Argumentation Framework (AF_S) is a quadruple $\langle \mathcal{A}, R, W, S\rangle$, where S is a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$, $\mathcal{A}$ is a set of arguments, R the attack binary relation on $\mathcal{A}$, and $W : \mathcal{A} \times \mathcal{A} \longrightarrow A$ a binary function called the* weight *function. Given $a, b \in \mathcal{A}$, $\forall (a, b) \in R$, $W(a, b) = s$ means that a attacks b with a strength level $s \in A$.*

In Def. 11 we define the notion of α-conflict-free partition: conflicts inside the same coalition can be now part of the solution until a cost threshold α is met, and not worse:

Definition 11. *Given a semiring-based AF_S, a partition of coalitions $\mathcal{G} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ is α-conflict-free for AF_S iff* $\prod_{\forall \mathcal{B}_i \in \mathcal{G}. b,c \in \mathcal{B}_i} W(b, c) \geq_S \alpha$ *(the $\prod$ uses the $\times$ of the semiring).*

In Fig. 7 there is an example of a 0.5-conflict-free partition using a *Fuzzy* semiring, i.e. the $\times$ used to compose the weights corresponds to min. Notice that only the attacks within the same coalition are considered: $min(0.6, 0.7, 0.5) = 0.5$.

Proposition 1. *If a partition is α_1-conflict-free, then the same partition is also α_2-conflict-free if $\alpha_1 <_S \alpha_2$.*

For instance, in *Weighted* semirings a 3-conflict-free partitions is also 4-conflict-free. In Def. 12 we extend with weights also the other kinds of partitions.

Definition 12. *Given an AF_S, a partition of coalitions $\mathcal{G} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ can be defined as α-stable (or α-admissible or α-complete) by only replacing conflict-free partitions with α-conflict-free partitions in Def. 7 (or Def. 8 or Def. 9).*

In Prop. 2 we relate the weighted partitions with those not weighted presented in Sec. 4.

Proposition 2. *Iff a partition is* **1***-conflict-free (or* **1***-stable,* **1***-admissible,* **1***-complete), then the same partition is also conflict-free (or stable, admissible, complete) as shown in Sec. 4.*

As a proof sketch, no attacks are present in the same coalition since **1** means "no attack", being the top element of the semiring.

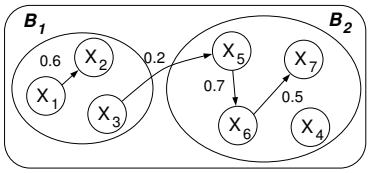

Fig. 7: A 0.5-conflict-free partition by using the *Fuzzy* semiring, i.e. $min(0.6, 0.7, 0.5) = 0.5$. The attack between x_3 and x_5 is not considered since they belong to different coalitions.

6 Mapping Partition Problems to SCSPs

In this section we show a mapping from the AF_S extended to coalitions (see Sec. 5) to SCSPs (see Sec. 3), i.e. $\mathcal{M} : AF_S \rightarrow SCSP$. $\mathcal{M}$ is described as follows: given an AF_S as described in Sec. 5, we define a variable for each argument $a_i \in \mathcal{A}$, i.e. $V = \{a_1, a_2, \ldots, a_n\}$. The value of a variable represents the coalition to which argument a_i belongs: i.e. each variable domain is $D = \{1, n\}$. For example if $a_1 = 2$ it means that the first argument belongs to the second coalition. We can have a maximum of n coalitions, that is all singletons.

In the following explanation, "b attacks a" means that b is a parent of a in the corresponding interaction graph, and "c attacks b attacks a" means that c is a grandparent of a. For the following constraint classes we consider a $AF_S = \langle \mathcal{A}, R, W, S \rangle$ where $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and $s \in A$:

1. **Conflict-free constraints.** Since we want to find an α-conflict-free partition, if $a_i \, R \, a_j$ and $W(a_i, a_j) = s$ we need to assign a s preference to the solution that includes both a_i and a_j in the same coalition of the partition: $c_{a_i,a_j}(a_i = k, a_j = k) = s$. Otherwise $c_{a_i,a_j}(a_i = k, a_j = l) = \mathbf{1}$ (with $l \neq k$).

2. **Admissible constraints.** For the admissibility of a partition, if a_i has several grandparents $a_{g1}, a_{g2}, \ldots, a_{gk}$ the parent a_f, we need to add a $k+1$-ary constraint $c_{a_i,a_{g1},\ldots,a_{gk}}(a_i = h, a_{g1} = j_1, \ldots, a_{gk} = j_k) = \mathbf{0}$ if $\forall j_i, j_i \neq h$ ($\mathbf{1}$ otherwise). This is because at least a grandparent must be taken in the same coalition, in order to defend a_i from his parent a_f. Notice that, if an argument is not attacked (i.e. has no parents), it can be taken or not in any admissible set. Moreover, if a_i has a parent but no grandparents, it is not possible to find any admissible partition, that is the SCSP is inconsistent (see Sec. 3).
3. **Complete constraints.** If we have an argument a_i with multiple grandchildren $a_{s1}, a_{s2}, \ldots, a_{sk}$, we need to add the constraint $c_{a_i,a_{s1},\ldots,a_{sk}}(a_i = j, a_{s1} = j, \ldots, a_{sk} = j) = \mathbf{1}$ ($\mathbf{0}$ otherwise). In words, if a_i is taken in a coalition j, all of its grandchildren must be included in the same coalition because j has to include all the defended arguments.
4. **Stable constraints.** They can be represented with a constraint such that for each pair of arguments a_i, a_j belonging to two different coalitions, respectively k and z, at least one of the attacks to a_j has to come from an argument in coalition k: if $b_1, b_2, \ldots b_n$ are all the arguments that attack a_j, $c_{a_i=k,a_j\neq k,b_1,b_2,\ldots,b_n}((b_1 = k) \vee (b_2 = k) \vee \cdots \vee (b_n = k)) = \mathbf{1}$ ($\mathbf{0}$ otherwise). Therefore, we model stable constraints with disjunctive constraints, which are difficult to solve.

Notice that in $\mathcal{M}$ only conflict-free constraints are soft in the strict sense, while the other constraints are associated with $\mathbf{0}$ (not admitted) or $\mathbf{1}$ (admitted) values of the semiring set.

Theorem 3 (Solution equivalence). *Given an $AF_S = \langle \mathcal{A}, R, W, S \rangle$, the solutions of the related SCSP obtained with the mapping $\mathcal{M}$ correspond to:*
- *all the α-conflict-free partitions of coalitions by using conflict-free constraints;*
- *all the α-stable partitions by using stable and conflict-free constraints;*
- *all the α-admissible partitions by using admissible, and conflict-free constraints;*
- *all the α-complete partitions by using complete and conflict-free constraints.*

Conflict-free, stable, admissible and complete partitions can be found by searching for $\mathbf{1}$-consistent solutions in the respective problems defined in Th. 3, as defined in Prop. 2. Notice that finding $\mathbf{1}$-conflict-free partitions is equivalent to well-known graph coloring problems which have been deeply studied also in constraint programming [18], where no two adjacent vertices share the same color:

Proposition 3. *The problem of finding a conflict-free partition of coalitions corresponds to finding a vertex-coloring partition of a graph [18], where each node of the same color belongs to the same coalition in a $\mathbf{1}$-conflict-free partition. The minimum number of colors needed to solve the problem corresponds to the minimum number of coalitions in a possible partition.*

In Fig. 8 we can see an example of classical (i.e. the attacks are not weighted) interaction graph. Only for this example we have 15 conflict-free partitions reported in Tab. 1. Among these conflict-free partitions, P_1, P_2, P_3, P_4, P_5 are also admissible partitions and P_1 is also the only one complete and stable partition; these partitions have been obtained with the implementation in Sec. 7.

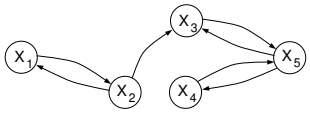

Fig. 8: An interaction graph.

$P_1 = \{\{x_1, x_3, x_4\}, \{x_2, x_5\}\}$	$P_2 = \{\{x_1, x_3, x_4\}, \{x_2\}, \{x_5\}\}$	$P_3 = \{\{x_1, x_3\}, \{x_2, x_4\}, \{x_5\}\}$
$P_4 = \{\{x_1, x_3\}, \{x_2, x_5\}, \{x_4\}\}$	$P_5 = \{\{x_1, x_3\}, \{x_2\}, \{x_4\}, \{x_5\}\}$	$P_6 = \{\{x_1, x_4\}, \{x_2, x_5\}, \{x_3\}\}$
$P_7 = \{\{x_1, x_4\}, \{x_2\}, \{x_3\}, \{x_5\}\}$	$P_8 = \{\{x_1, x_5\}, \{x_2, x_4\}, \{x_3\}\}$	$P_9 = \{\{x_1\}, \{x_2, x_4\}, \{x_3\}, \{x_5\}\}$
$P_{10} = \{\{x_1, x_5\}, \{x_2\}, \{x_3, x_4\}\}$	$P_{11} = \{\{x_1, \}, \{x_2, x_5\}, \{x_3, x_4\}\}$	$P_{12} = \{\{x_1\}, \{x_2\}, \{x_3, x_4\}, \{x_5\}\}$
$P_{13} = \{\{x_1, x_5\}, \{x_2\}, \{x_3\}, \{x_4\}\}$	$P_{14} = \{\{x_1\}, \{x_2, x_5\}, \{x_3\}, \{x_4\}\}$	$P_{15} = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\}$

Table 1: The list of all the conflict-free partitions of coalitions for the example in Fig. 8.

7 Implementation in JaCoP

The *Java Constraint Programming* solver [15] (JaCoP) is a Java library which provides a *Finite Domain Constraint Programming* paradigm [18].

To practically develop and test our model, we adopted the *Java Universal Network/Graph Framework* (*JUNG*) [17], a software library for the modeling, generation, analysis and visualization of graphs. Interaction graphs, where nodes are arguments and edges are attacks (see Sec. 2), clearly represent a kind of social network and consequently show the related properties [6]. Therefore, for the following tests we used the *KleinbergSmallWorldGenerator* class [17, 14] in JUNG, which randomly generates a $m \times n$ lattice with small-world properties [14]; each node has 4 local connections and 1 long range connection chosen randomly. An example of such random graphs with 25 nodes is shown in Fig. 9.

In this first implementation we decided to only implement **1**-conflict-free partitions, i.e. we do not consider weights on the attacks, and therefore we only need the crisp constraints of JaCoP. With this tool we can immediately check if a given partition is conflict-free, admissible, complete or stable. Moreover, we can exhaustively generate the partitions with such given properties: since the problem is $O(n^n)$ [16] (where n is the number of arguments) we limit the implementation to a partial search. In particular, we used the *Limited Discrepancy Search* (*LDS*), which is a kind of *Depth First Search* procedure adopting the method proposed in [11]. If a given number of different decisions along a search path is exhausted, then backtracking is initiated [15, 11]. Each time during the search, we select the variable which has most constraints assigned to it and we try the median from its current domain. Moreover, we set a timeout of 60 sec. to

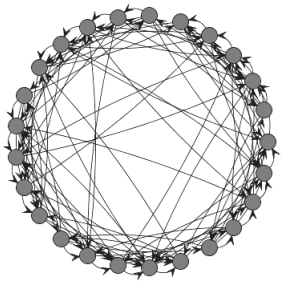

Fig. 9: A small-world network with 25 nodes generated with JUNG by using the *KleinbergSmallWorldGenerator* class [17, 14].

interrupt the search procedure and to report the number of solutions found only in that interval; we ran our experiments over 3 different random graphs with 9, 25 and 100 nodes. The results are shown in Tab. 2: it reports the number of found conflict-free and stable partitions (which limit the number of the other admissible and complete partitions as defined in Th. 2), the number of constraints used to represent the problem and the measured max depth of the search tree. Notice that, within the 60 sec. timeout, the proposed partial search is able to find only one stable partition for 100 nodes; also the reported number of conflict-free solutions in Tab. 2 is less for 100 than for 25 nodes. Therefore, further constraint solving techniques need to be used to improve these performance (left to future work in Sec. 9).

Nodes	Attacks	CFPS	SPS	#constr.	Max Depth
9	45	123	8	∼220	11
25	125	495984	119543	∼1440	61
100	500	92562	1	∼20600	218

Table 2: The test and the related statistics on three different small-world graphs : CFPS and SPS respectively are all the found conflict-free and stable partitions.

Notice that, in order to prevent symmetrically equivalent solutions we have also implemented symmetry breaking constraints for graph coloring as explained in [13] (see Prop. 3 for the analogies): any value permutation is a value symmetry in the coalition assignment of arguments.

8 Related Work and Comparison

The framework of Dung for argumentation is extended by Amgoud in [1] with a preference relation between elements; more in detail, Amgoud [1] provides the semantics (conflict-free, stable and preferred ones) of a coalition structure and a proof theory for testing whether a coalition is in the set of acceptable coalitions. An application of the model is also provided for the problem of task allocation

among partitions of autonomous agents. With respect to the work in this paper, the view in [1] is not focused on generating partitions of arguments, but on directly checking the property of already given coalition structures. Furthermore, [1] has no implementation to practically find solutions, as we instead do in Sec. 7. Moreover, the method to compute the weights of coalitions is not quantitative (but it is only qualitative) and parametric, as we are alternatively able to represent with semirings. In [8] an extension of the *Alternating-time Temporal Logic* (*ATL*) for modeling coalitions through argumentation is presented: a merge between ATL and the coalitional framework is obtained in order to express that agents are able to form a coalition which can successfully achieve a given property; the notions of defence and conflict-free are defined in terms of defeat rather than attack and preferences of arguments are given in a qualitative way (instead of quantitative as in our paper); to compute the desired classes of coalitions a model checker can be used; however, with such techniques, exponential complexity can be hardly faced while constraint programming provides a lot of techniques to tackle combinatorial problems [18]. In [6], social viewpoints (a model for goal based reasoning) are used to argue about coalitions in argumentation theory. The attack relation is based on the goal that agents have to achieve, that is, a coalition attacks another coalition if they share the same goal; this work does not provide a computational framework and only qualitative preferences over arguments are considered. In [5] a common computational and quantitative framework is presented, where attacks (and consequently, also the computation of the classical Dung's semantics) have an associated weight to represent how much inconsistency we tolerate in the solution. Our work extends [5] by considering partitions of arguments and showing an implementation in JaCoP (no implementation is given in [5]) with related tests on small-world graphs. Partitions of arguments implies redefining the whole (argumentation) theory concepts w.r.t [5], e.g. stability.

9 Summary and Future Work

We extended classical argumentation frameworks of [9] to the problem of forming coalitions of arguments, partitioning all the arguments of a given starting set. We redefined the classical definitions of Dung's extensions (conflict-free, admissible, stable and complete ones) in order to consider a partition of all the arguments into multiple coalitions, and modeled the problem of finding such coalitions with SCSPs [4, 3, 18]: this semiring-based formalism can be used to relax the concept of conflict-free partitions in order to allow some inconsistency (i.e. attacks) within the same coalition. The proposed quantitative framework can be used also to solve classical (i.e. crisp) CSPs. We have also solved a problem example considering only **1**-solutions with JaCoP [15] and then we performed tests on a small-world network randomly generated with [17]. Starting from a single set of arguments, the goal has been to partition it into multiple coalitions with the same features (e.g. stability or admissibility) without discarding any argument. In the future we want to implement α-conflict-free, α-stable, α-admissible and

α-complete partitions in JaCoP, for $\alpha <_S \mathbf{1}$. Moreover, we want to improve the performance obtained in Sec. 7 by testing different solvers and constraint techniques (e.g. by taking the inspiration from [16]).

References

1. L. Amgoud. An argumentation-based model for reasoning about coalition structures. In *ArgMAS05*, volume 4049 of *LNCS*, pages 217–228. Springer, 2005.
2. K. R. Apt and A. Witzel. A generic approach to coalition formation. *CoRR*, abs/0709.0435, 2007.
3. S. Bistarelli. *Semirings for Soft Constraint Solving and Programming*, volume 2962 of *LNCS*. Springer, 2004.
4. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 44(2):201–236, March 1997.
5. S. Bistarelli and F. Santini. A common computational framework for semiring-based argumentation systems. In *ECAI'10*, volume 215, pages 131–136. IOS Press, 2010.
6. G. Boella, L. van der Torre, and S. Villata. Social viewpoints for arguing about coalitions. In *PRIMA*, volume 5357 of *LNCS*, pages 66–77. Springer, 2008.
7. K. P. Bogart. *Introductory Combinatorics*. Academic Press, Inc., Orlando, FL, USA, 2000.
8. N. Bulling, J. Dix, and C. I. Chesñevar. Modelling coalitions: Atl + argumentation. pages 681–688. IFAAMAS, 2008.
9. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–357, 1995.
10. P. E. Dunne, A. Hunter, P. McBurney, S. Parsons, and M. Wooldridge. Inconsistency tolerance in weighted argument systems. pages 851–858. IFAAMS, 2009.
11. W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *IJCAI (1)*, pages 607–615, 1995.
12. B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *Knowl. Eng. Rev.*, 19(4):281–316, 2004.
13. G. Katsirelos and T. Walsh. Dynamic symmetry breaking constraints. In *Workshop on Modeling and Solving Problems with Constraints (at ECAI08)*, pages 39–44. Informal Proc., 2008.
14. J. Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.
15. K. Kuchcinski and R. Szymanek. Jacop - java constraint programming solver, 2001. http://jacop.osolpro.com/.
16. N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo. Coalition structure generation utilizing compact characteristic function representations. In *CP*, volume 5732 of *LNCS*, pages 623–638. Springer, 2009.
17. J. O'Madadhain, D. Fisher, S. White, and Y. Boey. The JUNG (Java Universal Network/Graph) framework. Technical report, UC Irvine, 2003.
18. F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming*. Elsevier Science Inc., NY, USA, 2006.
19. O. Shehory and S. Kraus. Task allocation via coalition formation among autonomous agents. In *IJCAI (1)*, pages 655–661, 1995.