On the Application of Answer Set Programming to the Conference Paper Assignment Problem

Giovanni Amendola, Carmine Dodaro*, Nicola Leone, Francesco Ricca

Department of Mathematics and Computer Science
University of Calabria, Rende, Italy
{amendola,dodaro,leone,ricca}@mat.unical.it

Abstract. Among the tasks to be carried out by conference organizers is the one of assigning reviewers to papers. That problem is known in the literature as the Conference Paper Assignment Problem (CPAP). In this paper we approach the solution of a reasonably rich variant of the CPAP by means of Answer Set Programming (ASP). ASP is an established logic-based programming paradigm which has been successfully applied for solving complex problems arising in Artificial Intelligence. We show how the CPAP can be elegantly encoded by means of an ASP program, and we analyze the results of an experiment, conducted on real-world data, that outlines the viability of our solution.

Keywords: Answer Set Programming, Conference Paper Assignment Problem, Applications

1 Introduction

Among the tasks to be carried out by conference organizers is the one of assigning reviewers to papers. That problem is known in the literature as the Conference Paper Assignment Problem (CPAP). The CPAP has quickly attracted the interest of researchers, and several formulations of the problem as well as a range of different solutions have been proposed [1, 2]. Actually, there is no recognized canonical form of the CPAP, and there is debate around the optimality criterion to be used for computing "fair" or "desiderable" assignments of papers to reviewers [1, 2]. In this paper we focus on a reasonably rich formulation of the problem where: (i) each paper has to be assigned to a given number of reviewers, (ii) each reviewer receives at most a given number of papers, and (iii) assignments are not done in case of (declared) conflict of interest. Moreover additional preference criteria have to be satisfied. In particular, the reviewer preferences (expressed by means of a numeric score) are maximized and the number of papers assigned to each reviewer is balanced. Note that this formulation of the CPAP complies (in terms of input data and parameters) with the information usually available to conference organizers in well-known conference paper management system such as Easychair (`http://www.easychair.org`). Moreover, it contemplates a set of requirements that are common to the majority of CPAP formulations in the literature [2]. It is

* Corresponding author

worth noting that the CPAP variant we consider in this paper is a computationally hard problem, indeed it can be proved to be NP-hard [3].

Complex combinatorial optimization problems, such as the CPAP, are usually the target for the application of formalisms developed in the area of Artificial Intelligence. Among these, Answer Set Programming (ASP) [4], a well-known declarative programming paradigm which has been proposed in the area of logic programming and nonmonotonic reasoning, is an ideal candidate. Indeed, ASP combines a comparatively high knowledge-modeling power [4] with a robust solving technology [5–12]. For these reasons ASP has become an established logic-based programming paradigm with successful applications to complex problems in Artificial Intelligence [13, 14], Bioinformatics [15–17], Databases [18, 19], Game Theory [20]; more recently ASP has been applied to solve industrial applications [21, 22].

Despite ASP can be used –in principle– for solving the CPAP, no specific investigation has been done [1, 2] (to the best of our knowledge) about the suitability of the ASP framework for solving real-world instances of the CPAP. The goal of this paper is to provide an assessment of the applicability of ASP to the CPAP. To this end, we consider a variant of the CPAP including constraints and optimization criteria commonly considered in the literature (see Section 3), and we show that it can be compactly encoded by means of an ASP program (see Section 4). Moreover, we analyze and discuss on the results of an experiment, conducted on real-world data, that outline the viability of an ASP-based solution (see Section 5). This work paves the way for the development of a more comprehensive ASP-based system for the CPAP.

2 Answer Set Programming

Answer Set Programming (ASP) [4] is a programming paradigm developed in the field of nonmonotonic reasoning and logic programming. In this section we overview the language of ASP, and we recall a methodology for solving complex problems with ASP. The reader is referred to [23] for a more detailed introduction.

Syntax. The syntax of ASP is similar to the one of Prolog. Variables are strings starting with uppercase letter and constants are non-negative integers or strings starting with lowercase letters. A *term* is either a variable or a constant. A *standard atom* is an expression $p(t_1,\ldots,t_n)$, where p is a *predicate* of arity n and $t_1,\ldots,t_n$ are terms. An atom $p(t_1,\ldots,t_n)$ is ground if $t_1,\ldots,t_n$ are constants. A *ground set* is a set of pairs of the form $\langle consts : conj\rangle$, where $consts$ is a list of constants and $conj$ is a conjunction of ground standard atoms. A *symbolic set* is a set specified syntactically as $\{Terms_1 : Conj_1;\cdots;Terms_t : Conj_t\}$, where $t > 0$, and for all $i \in [1,t]$, each $Terms_i$ is a list of terms such that $|Terms_i| = k > 0$, and each $Conj_i$ is a conjunction of standard atoms. A *set term* is either a symbolic set or a ground set. Intuitively, a set term $\{X : a(X,c), p(X); Y : b(Y,m)\}$ stands for the union of two sets: The first one contains the X-values making the conjunction $a(X,c), p(X)$ true, and the second one contains the Y-values making the conjunction $b(Y,m)$ true. An *aggregate function* [24] is of the form $f(S)$, where S is a set term, and f is an *aggregate function symbol*. Basically, aggregate functions map multisets of constants to a constant. The most common functions

implemented in ASP systems are the following: `#min`, minimal term, undefined for the empty set; `#max`, maximal term, undefined for the empty set; `#count`, number of terms; `#sum`, sum of integers. An *aggregate atom* is of the form $f(S) \prec T$, where $f(S)$ is an aggregate function, $\prec \in \{<, \leq, >, \geq, =, \neq\}$ is a comparison operator, and T is a term called guard. An aggregate atom $f(S) \prec T$ is ground if T is a constant and S is a ground set. An *atom* is either a standard atom or an aggregate atom. A *rule* r is of the form:

$$a_1 \mid \ldots \mid a_n \leftarrow b_1, \ldots, b_k, \texttt{not}\ b_{k+1}, \ldots, \texttt{not}\ b_m.$$

where $a_1, \ldots, a_n$ are standard atoms, $b_1, \ldots, b_k$ are atoms, $b_{k+1}, \ldots, b_m$ are standard atoms, and $n, k, m \geq 0$. A literal is either a standard atom a or its negation `not` a. The disjunction $a_1 | \ldots | a_n$ is the *head* of r, while the conjunction $b_1, \ldots, b_k, \texttt{not}\ b_{k+1}, \ldots, \texttt{not}\ b_m$ is its *body*. A rule is a *fact* if its body is empty ($\leftarrow$ is omitted), whereas it is a *constraint* if its head is empty. A variable appearing uniquely in set terms of a rule r is said to be *local* in r, otherwise it is *global* in r. An ASP program is a set of *safe* rules. A rule r is *safe* if both the following conditions hold: *(i)* for each global variable X of r there is a positive standard atom ℓ in the body of r such that X appears in ℓ; *(ii)* each local variable of r appearing in a symbolic set $\{Terms : Conj\}$ also appears in $Conj$.

A *weak constraint* [25] ω is of the form:

$$\leftsquigarrow b_1, \ldots, b_k, \texttt{not}\ b_{k+1}, \ldots, \texttt{not}\ b_m. \ [w@l]$$

where w and l are the weight and level of ω. (Intuitively, $[w@l]$ is read "as weight w at level l", where weight is the "cost" of violating the condition in the body of w, whereas levels can be specified for defining a priority among preference criteria). An ASP program with weak constraints is $\Pi = \langle P, W \rangle$, where P is a program and W is a set of weak constraints. A standard atom, a literal, a rule, a program or a weak constraint is *ground* if no variable appears in it.

Semantics. Let P be an ASP program. The *Herbrand universe* U_P and the *Herbrand base* B_P of P are defined as usual [23]. The ground program G_P is the set of all the ground instances of rules of P obtained by substituting variables with constants from U_P.

An *interpretation* I for P is a subset I of B_P. A ground atom a is true w.r.t. I if $a \in I$, and false otherwise. Literal `not` a is true in I if a is false in I, and true otherwise. An aggregate atom is true w.r.t. I if the evaluation of its aggregate function (i.e., the result of the application of f on the multiset S) w.r.t. I satisfies the guard; otherwise, it is false. A ground rule r is *satisfied* by I if at least one atom in the head is true w.r.t. I whenever all conjuncts of the body of r are true w.r.t. I. A model is an interpretation that satisfies all the rules of a program. Given a ground program G_P and an interpretation I, the *reduct* [26] of G_P w.r.t. I is the subset G_P^I of G_P obtained by deleting from G_P the rules in which a body literal is false w.r.t. I. An interpretation I for P is an *answer set* (or stable model [27]) for P if I is a minimal model (under subset inclusion) of G_P^I (i.e., I is a minimal model for G_P^I) [26]. A program having an answer set is called coherent, otherwise it is incoherent [28]. Given a program with weak constraints $\Pi = \langle P, W \rangle$, the semantics of Π extends from the basic case defined above. Thus, let $G_\Pi = \langle G_P, G_W \rangle$ be the instantiation of Π; a constraint $\omega \in G_W$ is violated by I if all the literals in ω are

true w.r.t. I. An *optimum answer set* O for Π is an answer set of G_P that minimizes the sum of the weights of the violated weak constraints in a prioritized way.

Problem Solving in ASP. ASP can be used to encode problems in a declarative way usually employing a *Guess&Check&Optimize* programming methodology [29]. This method requires that a database of facts is used to specify an instance of the problem; a set of rules, called "guessing part", is used to define the search space; admissible solutions are then identified by other rules, called the "checking part", which impose some admissibility constraints; finally weak constraints are used to single out solutions that are optimal with respect to some criteria, the "optimize part". As an example, consider the Traveling Salesman Problem (TSP). Given a weighted graph $G = \langle N, A \rangle$, where N is the set of nodes and A is the set of arcs with integer labels, the problem is to find a path of minimum length containing all the nodes of G. TSP can be encoded as follows:

$$
\begin{array}{ll}
r_1: & node(n). \qquad \forall\, \mathtt{n} \in N \\
r_2: & arc(i,j,w). \qquad \forall\, (\mathtt{i},\mathtt{j},\mathtt{w}) \in A \\
r_3: & inPath(X,Y) \mid outPath(X,Y) \leftarrow arc(X,Y,W). \\
r_4: & \leftarrow node(X),\ \#count\{I : inPath(I,X)\} \neq 1. \\
r_5: & \leftarrow node(X),\ \#count\{O : inPath(X,O)\} \neq 1. \\
r_6: & \leftarrow node(X),\ \mathtt{not}\ reached(X). \\
r_7: & reached(X) \leftarrow inPath(M,X),\ \#min\{N : node(N)\} = M. \\
r_8: & reached(X) \leftarrow reached(Y),\ inPath(Y,X). \\
r_9: & \leftsquigarrow inPath(X,Y),\ arc(X,Y,W). \quad [W@1]
\end{array}
$$

The first two rules introduce suitable facts, representing the input graph G. Then, rule r_3, which can be read as "each arc may or may not be part of the path", guesses a solution (a set of *inPath* atoms). Rules r_4–r_6 select admissible paths. In particular, rule r_4 (r_5) is satisfied if each node has exactly one incoming (resp. outgoing) arc in the solution. Moreover, rule r_6 ensures that the path traverses (say, reaches) all the nodes of G. Actually, this condition is obtained by checking that there exists a path reaching all the nodes of G and starting from the first node of N, say M. In particular, a node X is reached either if there is an arc connecting M to X (rule r_7), or if there is an arc connecting a reached node Y to X (rule r_8). Finally, solutions of minimal weight are selected by minimizing the cost W of arcs in the solution (rule r_9).

3 The Conference Paper Assignment Problem

Let $P = \{p_1, ..., p_s\}$ be a set of s papers and let $R = \{r_1, ..., r_t\}$ be a set of t reviewers. Each paper must be revised by ρ reviewers ($\rho \leq t$), and each reviewer must revise at most π papers ($\pi \leq s$). Moreover, to identify qualified reviewers, it is required that a reviewer r cannot review a paper p if there is a *conflict of interest* with some author of p. To formalize this property, it is introduced a *conflict function*, $\chi : R \times P \to \{0,1\}$, which assigns to each pair (r,p) the value 1 in case of conflict of interest, and 0, otherwise. Let $\chi(R,p) = \{r \in R | \chi(r,p) = 1\}$ be the set of all reviewers with a conflict of interest with p. A tuple $\langle P, R, \rho, \pi, \chi \rangle$ is called a *Paper Revision System* (PRS).

Definition 1 (Allocation solution). *An allocation solution for a PRS* $\Sigma = \langle P, R, \rho, \pi, \chi \rangle$ *is a function* $\psi : P \to 2^R$ *such that,*

$$|\psi(p)| = \rho, \text{ for each } p \in P; \tag{1}$$

$$\bigcap_{j \in M} \psi(p_j) = \emptyset, \text{ for each } M \subset \{1, \ldots, s\}, \text{ with } |M| = \pi + 1; \tag{2}$$

$$\psi(p) \cap \chi(R, p) = \emptyset, \text{ for each } p \in P; \tag{3}$$

$$R = \bigcup_{p \in P} \psi(p). \tag{4}$$

A PRS admitting an allocation solution is called *consistent*. Intuitively, first condition claims that each paper is assigned to exactly ρ reviewers. Second one states that it is not possible that a reviewer $r \in R$ revises more than π papers. Indeed, more formally, in such a case there would exist at least $\pi + 1$ papers $p^1, ..., p^{\pi+1} \in P$, such that $r \in \psi(p^j)$, for each $j = 1, ..., \pi + 1$. Hence, $r \in \bigcap_{j \in \{1,...,\pi+1\}} \psi(p^j)$, and so $\bigcap_{j \in \{1,...,\pi+1\}} \psi(p^j) \neq \emptyset$. Note that the number of papers assigned to a reviewer $r \in R$ is given by

$$\nu(r) = |\{p | r \in \psi(p)\}|.$$

In particular, we proved the following result.

Proposition 1. *Let* ψ *be an allocation solution for a consistent PRS* $\Sigma = \langle P, R, \rho, \pi, \chi \rangle$. *Then* $\nu(r) \leq \pi$, *for each* $r \in R$.

Third condition claims that an allocation solution cannot admit conflictual assignments. In particular, if $\chi = 0$ is the zero constant function ($\chi(r, p) = 0$, for each $(r, p) \in R \times P$), then condition (3) is always satisfied, because $\chi(R, p) = \emptyset$, for each $p \in P$. A PRS $\Sigma = \langle P, R, \rho, \pi, \chi \rangle$, where $\chi = 0$, is called a *non-conflictual* PRS, and we denote it by $\Sigma_0 = \langle P, R, \rho, \pi \rangle$. Finally, fourth condition states that to each reviewer r at least a paper p is assigned, i.e., $r \in \psi(p)$, for some $p \in P$.

It is important to establish sufficient or necessary conditions to have a consistent PRS, avoiding useless computations.

Proposition 2. *If* $\Sigma = \langle P, R, \rho, \pi, \chi \rangle$ *is a consistent PRS, then* $|R \setminus \chi(R, p)| \geq \rho$, $\forall p \in P$.

We give the following characterization for consistent non-conflictual PRS.

Proposition 3. *A non-conflictual PRS* $\Sigma_0 = \langle P, R, \rho, \pi \rangle$ *is consistent iff* $|P| \cdot \rho \leq |R| \cdot \pi$.

Example 1. Consider a non-conflictual PRS $\Sigma_0 = \langle P, R, \rho, \pi \rangle$ such that $P = \{p_1, p_2, p_3\}$ is a set of 3 papers and $R = \{r_1, r_2, r_3, r_4, r_5\}$ is a set of 5 reviewer. Each paper must be revised by $\rho = 3$ reviewers, and a reviewer must revise at most $\pi = 2$ papers. Note that Σ is consistent, since $3 \cdot \rho = 9 < 5 \cdot \pi = 10$. An allocation solution is given by $\psi(p_1) = \{r_1, r_2, r_3\}$, $\psi(p_2) = \{r_1, r_4, r_5\}$, $\psi(p_3) = \{r_2, r_3, r_4\}$.

In general, in a conference paper assignment, it is preferable that each reviewer has "more or less" the same number of papers of each other reviewer. Now, we introduce a notion of distance from a desiderata number of papers to formalize this request.

Definition 2 (Distance). *Given a consistent PRS $\Sigma = \langle P, R, \rho, \pi, \chi \rangle$, an allocation solution ψ, a reviewer $r \in R$, and a desiderata number of papers D, we define the distance of r from D as $\delta_D(r) = |D - \nu(r)|$, and the distance of R from D as $\delta_D(R) = \sum_{r \in R} \delta_D(r)$.*

Example 2. Consider the PRS Σ_0 and the allocation solution ψ of Example 1, and a desiderata number of papers $D = 1$. Therefore $\delta_1(r_i) = |1 - \nu(r_i)| = 1$, for $i = 1, 2, 3, 4$, and $\delta_1(r_5) = |1 - \nu(r_5)| = 0$. Hence, the distance of R from D is $\delta_1(R) = 4$.

Definition 3 (Minimal Allocation Solution). *Let $\Sigma = \langle P, R, \rho, \pi, \chi \rangle$ be a PRS, and let D be a desiderata number of papers for each reviewer. An allocation solution ψ for Σ is called minimal, if the distance of R from D is minimized.*

Another main feature of conference paper assignment is the possibility given to each reviewer of bidding some papers from the most desirable to the least desired. To this end, a *preference function* ϕ_r from P to a finite set $N = \{0, 1, ..., n\}$, assigning a preference value to each paper, is associated to each reviewer $r \in R$.

Definition 4 (Satisfaction degree). *Given an allocation solution ψ for a consistent PRS $\Sigma = \langle P, R, \rho, \pi, \chi \rangle$, and a preference function ϕ_r, for each $r \in R$, we define the satisfaction degree of ψ for Σ as the number*

$$d(\psi, \Sigma) = \sum_{p \in P} \sum_{r \in \psi(p)} \phi_r(p).$$

Example 3. Consider again the PRS Σ_0 and the allocation solution ψ of Example 1. Let $N = \{0, 1\}$ be a boolean set of preferences. Hence, a reviewer can just specify if a paper is desired (value 1) or not (value 0). Suppose that reviewers r_1 and r_2 desire paper p_3; reviewer r_3 desires papers p_1 and p_2; reviewer r_4 desires paper p_2; and reviewer r_5 desires paper p_1. Therefore, we have the following preference functions $\phi_{r_1}(p_3) = 1$, $\phi_{r_2}(p_3) = 1$, $\phi_{r_3}(p_1) = \phi_{r_3}(p_2) = 1$, $\phi_{r_4}(p_2) = 1$, $\phi_{r_5}(p_1) = 1$, and in all other cases the value is zero. The satisfaction degree of ψ for Σ is $d(\psi, \Sigma) = \phi_{r_1}(p_1) + \phi_{r_2}(p_1) + \phi_{r_3}(p_1) + \phi_{r_1}(p_2) + \phi_{r_4}(p_2) + \phi_{r_5}(p_2) + \phi_{r_2}(p_3) + \phi_{r_3}(p_3) + \phi_{r_4}(p_3) = 0 + 0 + 1 + 0 + 1 + 0 + 1 + 0 + 0 = 3$. Note that there exist others allocations solutions whose satisfaction degree is greater that this. Moreover, for this PRS, it is even possible to obtain the maximum satisfaction degree, that is 6, considering, for instance, $\psi'(p_1) = \{r_1, r_3, r_5\}$, $\psi'(p_2) = \{r_2, r_3, r_4\}$, $\psi'(p_3) = \{r_1, r_2, r_5\}$.

Definition 5 (Maximal Satisfying Allocation Solution). *Let $\Sigma = \langle P, R, \rho, \pi, \chi \rangle$ be a PRS, and let ϕ_r be a preference function, for each $r \in R$. An allocation solution ψ for Σ is called maximal satisfying, if the satisfaction degree of ψ for Σ is maximized.*

In the next, we consider the following formulations of CPAP:

1. Given a PRS Σ, a desiderata number of papers for each reviewer, and a preference function for each reviewer, finding among the minimal allocation solutions for Σ the one that is maximal satisfying.
2. Given a PRS Σ, a desiderata number of papers for each reviewer, and a preference function for each reviewer, finding among the maximal satisfying allocation solutions for Σ the one that is minimal.

4 Encoding CPAP in ASP

This section illustrates the ASP program which solves the Conference Paper Assignment problem specified in the previous section. First, the input data is described (Section 4.1), then the ASP encoding is presented (Section 4.2).

4.1 Data Model

The input is specified by means of factual instances of the following predicates:

- Instances of the predicate *paper(id)* represent the information about papers, where *id* represents a numerical identifier of a specific paper.
- Instances of the predicate *reviewer(id)* represent the information about reviewers, where *id* represents a numerical identifier of a specific reviewer.
- Instances of the predicate *score(id_reviewer, id_paper, score)* represent the information about the preference of reviewers for papers, where *id_reviewer* is the identifier of the reviewer, *id_paper* is the identifier of the paper, and *score* represents a numerical preference ($0 \leq score \leq 4$) assigned by the reviewer to the paper, where a lower score is associated with a higher confidence.
- Instances of the predicate *conflict(id_reviewer, id_paper)* represent a conflict of the reviewer with the paper.
- The only instance of the predicate *reviewersToPaper(ρ)* represents the number of reviewers that must be assigned to each paper.
- The only instance of the predicate *maxPaperPerReviewer(π)* represents the maximum number of papers that can be assigned to each reviewer.
- The only instance of the predicate *desiderata(d)* represents the number of papers that organizers are willing to assign to each reviewer.

4.2 ASP encoding

In this section we describe the ASP rules used for solving the conference paper assignment problem. We follow the *Guess&Check&Optimize* programming methodology [29]. In particular, the following rule guesses the reviewers to assign to each paper:

$$assign(R,P) \mid nassign(R,P) \leftarrow paper(P), reviewer(R), \texttt{not}\ conflict(R,P). \quad (5)$$

The guess is limited to the reviewers that are not in conflict with the specific paper.

Each paper must be assigned to exactly N reviewers, thus all assignments violating this requirement are filtered out by the following constraint:

$$\leftarrow\ paper(P),\ \#count\{R : assign(R,P)\} \neq N. \quad (6)$$

Then, assignments exceeding the maximum number of paper assigned to each reviewer are filtered out by the following constraint:

$$\leftarrow reviewer(R),\ maxPaperPerReviewer(M),\ \#count\{P : assign(R,P)\} > M. \quad (7)$$

Moreover, each reviewer must be assigned at least to one paper:

$$\begin{aligned} workload(R,N) \ &\leftarrow \#count\{P : assign(R,P)\} = N, reviewer(R), \\ &\qquad maxPaperPerReviewer(M), N \leq M. \\ &\leftarrow \ reviewer(R),\ workload(R,N),\ N < 1. \end{aligned} \tag{8}$$

The predicate *workload(reviewer, number)* stores the association between a reviewer and the number of papers assigned to him/her.

Theoretical improvements. In the following some constraints exploiting the theoretical results obtained in Section 3 are given. Since each paper must be assigned to exactly ρ reviewers, if the number of reviewers with no conflicts for a paper is less than ρ then a solution cannot exist (see Proposition 2). This is modeled by the following constraint:

$$\begin{aligned} \leftarrow \ & paper(P_1),\ reviewersToPaper(N),\ \#count\{R_1 : reviewer(R_1)\} = R, \\ & \#count\{R_1 : conflict(R_1,P_1)\} = C,\ R - C < N. \end{aligned} \tag{9}$$

Moreover, the results presented in Proposition 3 are exploited by adding:

$$\begin{aligned} \leftarrow \ & reviewersToPaper(N), maxPaperPerReviewer(M), \\ & \#count\{R_1 : reviewer(R_1)\} = R, \#count\{P_1 : paper(P_1)\} = P, \\ & P * N > R * M. \end{aligned} \tag{10}$$

Intuitively, if P (number of papers) times N (number of reviewers per paper) exceeds R (number of reviewers) times M (maximum number of papers assigned to a reviewer) a solution cannot exist.

Optimization requirements. The *satisfaction degree* and the *minimal allocation* requirements are obtained in our encoding by means of two weak constraints, where the numerical values ℓ_p and ℓ_w represent their levels; an order on the preferences can be later on specified by properly assigning a value to those levels.

Concerning the satisfaction degree of reviewers, the assignment of a reviewer to a paper is associated with a cost depending on the preference assigned from the reviewer to the paper. The maximum preference for a paper, i.e. a score equal to zero, is associated with no cost. Thus, the minimization of the cost (i.e. the maximization of satisfaction degree) is obtained by means of the following weak constraint:

$$\leftsquigarrow \ reviewer(R),\ assign(R,P),\ score(R,P,S).\ [S@\ell_p] \tag{11}$$

Finally, the minimization of the distance between the desiderata number of papers to be assigned to each reviewer and the number of papers assigned by the solution is obtained by means of the following weak constraint:

$$\leftsquigarrow \ reviewer(R),\ workload(R,N),\ desiderata(D),\ V = |D - N|.\ [V@\ell_w] \tag{12}$$

Intuitively, for each reviewer the distance is computed as the difference between the number of assigned papers to him/her and the desiderata number of papers. Then, a greater distance corresponds to a greater cost associated to the solution.

5 Experiments

In our experiments we considered a set of four real events held in the recent years, whose names are omitted for protecting our sources. For each event, we considered $\pi = 4$ and the desiderata number of papers to be assigned to each reviewer equal to 4. *Event 1* was composed of 31 papers, 46 reviewers, and $\rho = 4$; *event 2* was composed of 59 papers, 55 reviewers, and $\rho = 3$; *event 3* was composed of 16 papers, 31 reviewers, and $\rho = 4$; *event 4* was composed of 15 papers, 30 reviewers, $\rho = 4$. Concerning the preferences we considered two settings for the levels ℓ_p and ℓ_w of (11) and (12), i.e. $\ell_w > \ell_p > 0$ and $\ell_p > \ell_w > 0$ corresponding to formulations 1. and 2. of CPAP, respectively.

We executed the ASP solvers CLASP [8] and WASP [30]. The former has been configured with the model-guided algorithm called *bb* [8], which basically searches for an answer set so to initialize an upper bound of the optimum cost, and new answer sets of improved cost are iteratively searched until the optimum cost is found. WASP has been configured with the core-guided algorithm called *one* [5], which searches for an answer set satisfying all weak constraints. If there is no answer set of this kind, an unsatisfiable core is identified, i.e. a subset of the weak constraints that cannot be jointly satisfied, representing a lower bound of the optimum cost. In addition, WASP is able to produce upper bounds of the optimum cost during the search of an unsatisfiable core. The experiments were run on an Intel Xeon 2.4 GHz with 16 GB of RAM, and time and memory were limited to 60 minutes and 15 GB, respectively.

Formulation 1 ($\ell_w > \ell_p > 0$). An overview of the obtained results is given in Table 1. For each event the number of reviewers receiving one, two, three or four papers within different time limits is reported. Concerning the first event the 76% of reviewers received exactly the desiderata number of papers, i.e. 3. The remaining 17% and 7% received 2 and 1 paper, respectively. According to this solution no reviewer has to review more than 3 papers. Even better results are obtained for the second event, where 78% of reviewers received 3 papers, and the remaining 22% received 4 papers. None of the reviewers received less than 3 papers. Concerning events 3 and 4, solutions found by CLASP assign 3 papers only to few reviewers. Similar results are found by WASP where 29 out of 31 and 30 out of 30 reviewers are associated to exactly 2 papers, respectively. This might be explained by the few number of papers w.r.t. the number of reviewers, which makes it difficult to assign the desiderata number of papers to each reviewer.

Formulation 2 ($\ell_p > \ell_w > 0$). An overview of the obtained results is given in Table 2, where for each event lower and upper bounds found by CLASP and WASP are reported. The analysis of lower and upper bounds allows us to estimate the *error* of the best found solution, reported in the last column of the table and computed as follows:

$$\varepsilon(ub, lb) := \begin{cases} \frac{ub-lb}{lb} & \text{if } ub \neq \infty \text{ and } lb \neq 0; \\ \infty & \text{if } ub = \infty, \text{ or both } ub \neq 0 \text{ and } lb = 0; \\ 0 & \text{if } ub = lb = 0. \end{cases}$$

As first observation, WASP is able to find the solution maximizing the satisfaction of reviewers within 2 seconds for all the events but *event2*. Concerning *event2*, the best

result is obtained by CLASP that is able to provide a solution with an error equal to 1 within 60 seconds. Results are far better if we look at the solution found within 3600 seconds, where CLASP provides a solution with an error equal to 0.21. For the sake of completeness, we also mention that intermediate solutions were found within 600 and 900 seconds with errors equal to 0.36 and 0.22, respectively. Concerning the minimization of the distance, the solution produced by CLASP within 60 seconds has an error less than 0.5 for all the events but *event2*.

6 Related Work

The problem of conference paper assignment has been attracting the interest of researchers in the last two decades [1, 2]. Researchers from different areas have focused on different aspects of CPAP [31–35]. Data mining techniques have been applied for inferring preferences and desiderata of reviewers; operational research tools have been used to compute assignments; in Economy the CPAP has been related to the allocation of indivisible goods to a set of agents. The solving methods in the literature range from dedicated algorithms, to genetic algorithms, integer programming-based methods, and approximation algorithms. All these are different from our approach in terms of modeling language, whereas our formulation of the problem shares often the constraints on assignments and in some cases the optimization criteria with some of these works.

Table 1. Workload of reviewers computed by CLASP when $\ell_w > \ell_p > 0$.

	Time (s)	Rev. with 1 paper	Rev. with 2 papers	Rev. with 3 papers	Rev. with 4 papers
event1	≤ 1	7	4	31	4
	≤ 2	6	5	32	3
	≤ 10	3	8	35	0
	≤ 60	3	8	35	0
	≤ 3600	3	8	35	0
event2	≤ 1	8	1	17	29
	≤ 2	4	3	25	23
	≤ 10	0	0	43	12
	≤ 60	0	0	43	12
	≤ 3600	0	0	43	12
event3	≤ 1	13	3	15	0
	≤ 2	13	3	15	0
	≤ 10	9	11	11	0
	≤ 60	9	11	11	0
	≤ 3600	9	11	11	0
event4	≤ 1	11	8	11	0
	≤ 2	12	6	12	0
	≤ 10	12	6	12	0
	≤ 60	11	8	11	0
	≤ 3600	13	4	13	0

Since an exhaustive description of the state of the art can be found on existing survey papers [2, 1], in the following we locate our contribution by comparing some of the recent papers on CPAP.

In [36] a fuzzy mathematical model for the assignment of experts to project proposal is investigated, which is a problem similar to CPAP. The method imposes assignment constraints that are similar to the ones considered in this paper, but it does not consider conflicts and focuses on a matching criteria that is defined using linguistic variables denoting the expertise of experts with respect to proposals. The resultant fuzzy model was solved with the selected fuzzy ranking methods. The approach of [36] cannot be directly compared to ours, since the modeling itself would not fit the standard ASP framework that works on problems formulations where all the information is crisp.

In [37] authors considered the problem of determining a prediction of reviewer's preference, and provided some empirical evidence that an accurate identification of preferences can improve satisfaction of reviewers. Thus, the goal of [37] is to improve the modeling of reviewer preferences. The problem of determining reviewer relevance by automatically identifying reviewer profiles was also the subject of research [38]. These studies could be employed for designing better models of reviewer's preferences that could be used as input of a method that computes the optimal assignment as the one considered is in this paper.

Table 2. Lower and upper bounds computed by CLASP and WASP when $\ell_p > \ell_w > 0$.

	Time (s)	CLASP (ub)		WASP (ub)		WASP (lb)		ε		Optimum	
		ℓ_p	ℓ_w	ℓ_p	ℓ_w	ℓ_p	ℓ_w	ℓ_p	ℓ_w	ℓ_p	ℓ_w
event1	$\leq$ 1	56	28	∞	∞	18	0	2.11	∞	-	-
	$\leq$ 2	46	26	**25**	18	**25**	10	**0**	0.80	25	-
	$\leq$ 10	**25**	16	**25**	18	**25**	10	**0**	0.60	25	-
	$\leq$ 60	**25**	14	**25**	18	**25**	10	**0**	0.40	25	-
	$\leq$ 3600	**25**	14	**25**	18	**25**	10	**0**	0.40	25	-
event2	$\leq$ 1	54	48	∞	∞	8	0	5.75	∞	-	-
	$\leq$ 2	52	46	∞	∞	8	0	5.50	∞	-	-
	$\leq$ 10	45	30	∞	∞	8	0	4.62	∞	-	-
	$\leq$ 60	28	24	37	56	14	0	1	∞	-	-
	$\leq$ 3600	17	28	37	56	14	0	0.21	∞	-	-
event3	$\leq$ 1	25	31	**18**	29	**18**	18	**0**	0.61	18	-
	$\leq$ 2	20	29	**18**	29	**18**	20	**0**	0.45	18	-
	$\leq$ 10	**18**	29	**18**	29	**18**	20	**0**	0.45	18	-
	$\leq$ 60	**18**	29	**18**	29	**18**	20	**0**	0.45	18	-
	$\leq$ 3600	**18**	29	**18**	29	**18**	20	**0**	0.45	18	-
event4	$\leq$ 1	24	36	**18**	30	**18**	26	**0**	0.15	18	-
	$\leq$ 2	20	34	**18**	30	**18**	26	**0**	0.15	18	-
	$\leq$ 10	**18**	32	**18**	30	**18**	26	**0**	0.15	18	-
	$\leq$ 60	**18**	30	**18**	30	**18**	26	**0**	0.15	18	-
	$\leq$ 3600	**18**	30	**18**	30	**18**	27	**0**	0.11	18	-

A formulation of reviewer preferences based on a combination of information about topics and direct preference of reviewers is considered in [39]. Here a matching criteria based on a matching degree function is proposed. The criterion of [39] can be modeled in ASP, and can be considered as one possible extension of our current model. Topic coverage of the paper reviewer assignment is the main optimization employed in [40], where also an algorithm for computing an approximation of the optimal solution is proposed in presence of conflicts of interest.

An alternative formulation of the CPAP has been proposed in [41], where a group-to-group reviewer assignment problem is defined. The idea is that manuscripts and reviewers are divided into groups, with groups of reviewers assigned to groups of manuscripts. A two-phase stochastic-biased greedy algorithm is then proposed to solve the problem. This variant of the problem is less similar to the traditional CPAP formulation that we consider in this paper, so a direct comparison is not feasible.

The approach that is most related ours is [42] where ASP has been also employed, but the CPAP is studied as an example of *reconfiguration* problem. Thus, the focus is on *updating a given solution* rather than in the computation of a new assignment.

7 Conclusion

The main goal of this paper was to provide an assessment of the applicability of ASP to the CPAP. We first provided a formal description of the problem, which combines the most common constraints and optimization criteria considered in the literature, and we outlined some theoretical properties of the problem. Then, we provided a disjunctive logic program modeling the CPAP. The ASP encoding is natural and intuitive, in the sense that it was obtained by applying the standard modeling methodology, and it is easy to understand. We also modeled in a natural way the theoretical conditions ensuring the absence of solutions, so that ASP solvers can easily recognize unsatisfiable instances. Finally, the performance of our ASP-based approach was studied in an experiment.

We conclude that ASP is suitable from the perspective of modeling, since we obtained a natural ASP encoding of the problem. Moreover, the results of an experiment outline that an ASP-based approach can perform well on real-world data.

Future work will focus on extending the framework with additional information (e.g., topics, coauthor information, etc.) and with additional preference criteria (e.g., different models of fairness, coauthors distance, etc.). Indeed, the flexibility of ASP as a modeling language should allow us to enrich current model or encode some of its variants. We also planned to extend the experimental analysis by considering more data and more computation methods. Moreover, we are investigating the application of ASP to other hard problems [43, 44].

Acknowledgments. This work was partially supported by MIUR under PON project "Ba2Know (Business Analytics to Know) Service Innovation - LAB", N. PON03PE_00 001_1, and by MISE under project "PIUCultura (Paradigmi Innovativi per l'Utilizzo della Cultura)", N. F/020016/01-02/X27.

References

1. Goldsmith, J., Sloan, R.H.: The AI conference paper assignment problem. In: Proc. AAAI Workshop on Preference Handling for Artificial Intelligence. (2007) 53–57
2. Wang, F., Chen, B., Miao, Z.: A Survey on Reviewer Assignment Problem. In: New Frontiers in Applied Artificial Intelligence, IEA/AIE. Volume 5027 of LNCS., Springer (2008) 718–727
3. Manlove, D., Irving, R.W., Iwama, K., Miyazaki, S., Morita, Y.: Hard variants of stable marriage. Theor. Comput. Sci. **276**(1-2) (2002) 261–279
4. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12) (2011) 92–103
5. Alviano, M., Dodaro, C., Ricca, F.: A MaxSAT Algorithm Using Cardinality Constraints of Bounded Size. In: IJCAI, AAAI Press (2015) 2677–2683
6. Alviano, M., Faber, W., Leone, N., Perri, S., Pfeifer, G., Terracina, G.: The disjunctive datalog system DLV. In de Moor, O., Gottlob, G., Furche, T., Sellers, A.J., eds.: Datalog. Volume 6702 of LNCS., Springer (2010) 282–301
7. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the Fifth Answer Set Programming Competition. Artif. Intell. **231** (2016) 151–181
8. Gebser, M., Kaminski, R., Kaufmann, B., Romero, J., Schaub, T.: Progress in clasp Series 3. In: LPNMR. Volume 9345 of LNCS., Springer (2015) 368–383
9. Giunchiglia, E., Leone, N., Maratea, M.: On the relation among answer set solvers. Ann. Math. Artif. Intell. **53**(1-4) (2008) 169–204
10. Giunchiglia, E., Maratea, M.: On the Relation Between Answer Set and SAT Procedures (or, Between cmodels and smodels). In: ICLP. Volume 3668 of LNCS., Springer (2005) 37–51
11. Maratea, M., Pulina, L., Ricca, F.: A multi-engine approach to answer-set programming. TPLP **14**(6) (2014) 841–868
12. Maratea, M., Ricca, F., Faber, W., Leone, N.: Look-back techniques and heuristics in DLV: Implementation, evaluation, and comparison to QBF solvers. J. Algorithms **63**(1-3) (2008) 70–89
13. Balduccini, M., Gelfond, M., Watson, R., Nogueira, M.: The USA-Advisor: A Case Study in Answer Set Planning. In: LPNMR, Springer (2001) 439–442
14. Gaggl, S.A., Manthey, N., Ronca, A., Wallner, J.P., Woltran, S.: Improved answer-set programming encodings for abstract argumentation. TPLP **15**(4-5) (2015) 434–448
15. Campeotto, F., Dovier, A., Pontelli, E.: A declarative concurrent system for protein structure prediction on GPU. J. Exp. Theor. Artif. Intell. **27**(5) (2015) 503–541
16. Erdem, E., Öztok, U.: Generating explanations for biomedical queries. TPLP **15**(1) (2015) 35–78
17. Fionda, V., Palopoli, L., Panni, S., Rombo, S.E.: A technique to search for functional similarities in protein-protein interaction networks. IJDMB **3**(4) (2009) 431–453
18. Manna, M., Ricca, F., Terracina, G.: Taming primary key violations to query large inconsistent data via ASP. TPLP **15**(4-5) (2015) 696–710
19. Marileo, M.C., Bertossi, L.E.: The consistency extractor system: Answer set programs for consistent query answering in databases. Data Knowl. Eng. **69**(6) (2010) 545–572
20. Amendola, G., Greco, G., Leone, N., Veltri, P.: Modeling and reasoning about NTU games via answer set programming. In: IJCAI, IJCAI/AAAI Press (2016) 38–45
21. Grasso, G., Leone, N., Manna, M., Ricca, F.: ASP at work: Spin-off and applications of the DLV system. In: Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning - Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday. Volume 6565 of LNCS., Springer (2011) 432–451

22. Dodaro, C., Gasteiger, P., Leone, N., Musitsch, B., Ricca, F., Shchekotykhin, K.: Combining answer set programming and domain heuristics for solving hard industrial problems. TPLP **16**(5-6) (2016) (to appear)
23. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
24. Alviano, M., Faber, W., Gebser, M.: Rewriting recursive aggregates in answer set programming: back to monotonicity. TPLP **15**(4-5) (2015) 559–573
25. Buccafurri, F., Leone, N., Rullo, P.: Enhancing Disjunctive Datalog by Constraints. IEEE Trans. Knowl. Data Eng. **12**(5) (2000) 845–860
26. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artif. Intell. **175**(1) (2011) 278–298
27. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Comput. **9**(3/4) (1991) 365–386
28. Amendola, G., Eiter, T., Fink, M., Leone, N., Moura, J.: Semi-equilibrium models for paracoherent answer set programs. Artif. Intell. **234** (2016) 219–271
29. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Trans. Comput. Log. **7**(3) (2006) 499–562
30. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: LPNMR. Volume 9345 of LNCS., Springer (2015) 40–54
31. Bogomolnaia, A., Moulin, H.: A new solution to the random assignment problem. Journal of Economic Theory **100**(2) (October 2001) 295–328
32. Demange, G., Alkan, A., Gale, D.: Fair Allocation of Indivisible Goods and Money and Criteria of Justice. Econometrica **59**(4) (1991) 1023–1039
33. Hartvigsen, D., Wei, J.C., Czuchlewski, R.: The Conference Paper-Reviewer Assignment Problem. Decision Sciences **30**(3) (1999) 865–876
34. Janak, S.L., Taylor, M.S., Floudas, C.A., Burka, M., Mountziaris, T.J.: Novel and Effective Integer Optimization Approach for the NSF Panel-Assignment Problem: A Multiresource and Preference-Constrained Generalized Assignment Problem. Industrial & Engineering Chemistry Research **45**(1) (2006) 258–265
35. Svensson, L.G.: Strategy-proof allocation of indivisible goods. Social Choice and Welfare **16**(4) (1999) 557–567
36. Das, G.S., Göçken, T.: A fuzzy approach for the reviewer assignment problem. Computers & Industrial Engineering **72** (2014) 50–57
37. Conry, D., Koren, Y., Ramakrishnan, N.: Recommender systems for the conference paper assignment problem. In: RecSys, ACM (2009) 357–360
38. Mimno, D.M., McCallum, A.: Expertise modeling for matching papers with reviewers. In: ACM SIGKDD, ACM (2007) 500–509
39. Li, X., Watanabe, T.: Automatic Paper-to-reviewer Assignment, based on the Matching Degree of the Reviewers. In: KES. Volume 22 of Procedia Computer Science., Elsevier (2013) 633–642
40. Long, C., Wong, R.C., Peng, Y., Ye, L.: On Good and Fair Paper-Reviewer Assignment. In: ICDM, IEEE Computer Society (2013) 1145–1150
41. Wang, F., Zhou, S., Shi, N.: Group-to-group reviewer assignment problem. Computers & OR **40**(5) (2013) 1351–1362
42. Ryabokon, A., Polleres, A., Friedrich, G., Falkner, A.A., Haselböck, A., Schreiner, H.: (Re)Configuration Using Web Data: A Case Study on the Reviewer Assignment Problem. In: RR. Volume 7497 of LNCS., Springer (2012) 258–261
43. Fionda, V., Greco, G.: The complexity of mixed multi-unit combinatorial auctions: Tractability under structural and qualitative restrictions. Artif. Intell. **196** (2013) 1–25
44. Fionda, V., Pirrò, G.: Querying graphs with preferences. In: CIKM, ACM (2013) 929–938