

Automatic Reviews' Assignments through Answer Set Programming

Davide Di Pierro^{1,*}, Eleonora Bernasconi¹ and Stefano Ferilli¹

¹Università degli Studi di Bari Aldo Moro, Via Edoardo Orabona, 4, Bari, 70125, Italy

Abstract

The problem of automatically assigning reviews is a crucial task in conference management. Not only is it a time-consuming and challenging task, but it is also one of the most important peculiarities that contribute to the good/bad organisation of the conference, not to mention the degree of satisfaction of the people involved. During review assignments, many constraints come into play: the maximum number of papers per reviewer, the minimum number of reviews per paper, conflict handling and, last but not least, the similarity between papers' topics and reviews' interests. In this paper, we propose a strategy to map topics using the ACM Computing taxonomy, and a modelization of the problem in a Constraint Satisfaction setting relying on the Answer Set Programming (ASP) logical framework. This strategy, although not scalable, showed the capability of managing many real situations that have not been captured so far. Experiments demonstrated the goodness of performances for small/medium numbers of papers.

Keywords

Assignment of reviewers to papers, Constraint Satisfaction Problem, Answer Set Programming, Modelling

1. Introduction


The problem of assigning papers to reviewers is fundamental in organizing conferences, workshops or peer review processes. The challenge lies in the fact that many constraints come into play. An effective assignment should, in principle, guarantee that every paper is reviewed by (at least two) persons among the most competent in the field(s) in which the paper resides. Moreover, it is expected that reviewers are comfortable in the reviewing process by judging papers they feel are related to their line of research, offering the possibility to create a profitable interaction between the reviewed and the reviewer. A mapping satisfying these conditions does not seem tough in principle. Yet in reality, these ideal conditions clash with constraints that make the assignment cumbersome, time-consuming, and with a generally lower satisfaction of the researchers involved, possibly causing frustration and a lower quality of the event itself. More formally, the problem of paper assignments can be viewed as finding a *function* (assignment) mapping papers to (a set of) reviewers. Given a finite set of papers $\mathcal{P} = \{p_1, p_2, \dots, p_k\}$,

IRCDL 2025: 21st Conference on Information and Research Science Connecting to Digital and Library Science, February 20-21 2025, Udine, Italy

*Corresponding author.

✉ davide.dipierro@uniba.it (D. Di Pierro); eleonora.bernasconi@uniba.it (E. Bernasconi); stefano.ferilli@uniba.it (S. Ferilli)

ORCID [0000-0002-8081-3292](https://orcid.org/0000-0002-8081-3292) (D. Di Pierro); [0000-0003-3142-3084](https://orcid.org/0000-0003-3142-3084) (E. Bernasconi); [0000-0003-1118-0601](https://orcid.org/0000-0003-1118-0601) (S. Ferilli)

 © 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

a finite set of authors $\mathcal{A} = \{a_1, a_2, \dots, a_m\}$ with authorship function $\mathcal{T} : \mathcal{P} \rightarrow 2^{\mathcal{A}}$, and a set of reviewers $\mathcal{R} = \{r_1, r_2, \dots, r_n\}$, the assignment function $\mathcal{F} : \mathcal{P} \rightarrow 2^{\mathcal{R}}$ is a function that assigns to each paper a set of reviewers. The function is neither injective since multiple papers may be submitted to the exact set of reviewers (although rare) nor surjective given that it is not mandatory in all cases a person has to work on reviewing. Figure 1 shows the visual interpretation of an assignment function. The rest of the paper is organised as follows: Section 2 provides the theoretical background of the involved concepts, Section 3 provides an overview of the state-of-the-art for this task, Section 4 provides the modelization of the model, Section 5 lists the ASP implementation of the modelization, Section 6 shows the result of the experimentation and gives more insight about the process and, finally, Section 7 concludes the manuscript and gives rise to future extensions and integrations.

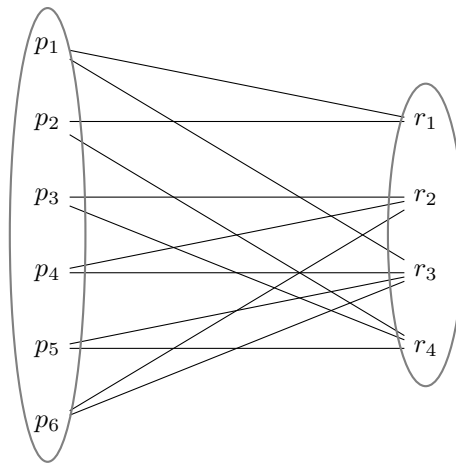


Figure 1: An example of papers' assignments.

2. Background

2.1. Ontologies

In order to understand how to relate topics of interest of reviewers with those related to the papers, a large, standard and comprehensive conceptualization needs to be considered. We refer to these conceptualizations as *ontology*. This concept has origins in philosophy. The Computer Science community understood the need to catalogue and classify a domain since the early beginning of the discipline. The word *ontology* is used with different meanings in different communities. In the early days, ontology designers sought to provide a complete classification of every concept available in the “universe” of knowledge. Not too far did this approach fail due to the overwhelming amount of objects to be classified and the variety of possible ways in which objects can be categorized according to the specific use, context, period and many other environmental factors. Last but not least, the objective of an *ontology* formalization is to provide a formalization that is “stable”, as much as possible. For this reason, it should not be too tight to a specific context.

In the AI field, the aim is to create a model of the reality of interest. From a formal point of view, an ontology is a triple $(\mathcal{C}, \mathcal{R}, \mathcal{A})$ in which:

- \mathcal{C} is a set of concepts.
- \mathcal{R} is a set of relationships between concepts.
- \mathcal{A} is a set of axioms on which the universe we want to describe is based [1].

\mathcal{R} is therefore a subset of the Cartesian product $\mathcal{C} \times \mathcal{C}$. An ontology can also be devoid of axioms, in which case it is called non-axiomatized. An ontology can be basically seen as a set of concepts connected through relationships: hence, graphs are often used to describe them. A *graph* \mathcal{G} is a pair $(\mathcal{V}, \mathcal{E})$ where \mathcal{V} represents the set of vertices, in our case the concepts, while \mathcal{E} is the set of arcs joining two vertices. In this case, \mathcal{V} is the set of concepts \mathcal{C} , and \mathcal{E} is the set of relationships \mathcal{R} . Given this analogy, networks are the most common visual tool to represent ontologies. They represent an oriented graph in which each arc is labelled, and the label is, in a very intuitive way, the relationship between the two concepts. We report a small example in Figure 2 where $\mathcal{C} = \{\text{Entity, Object, Person, Engine, Car, Mechanic}\}$, $\mathcal{R} = \{\text{is_a, has_part, repairs}\}$ showing some general concepts put into a hierarchy by the well-known “is_a” relationship.

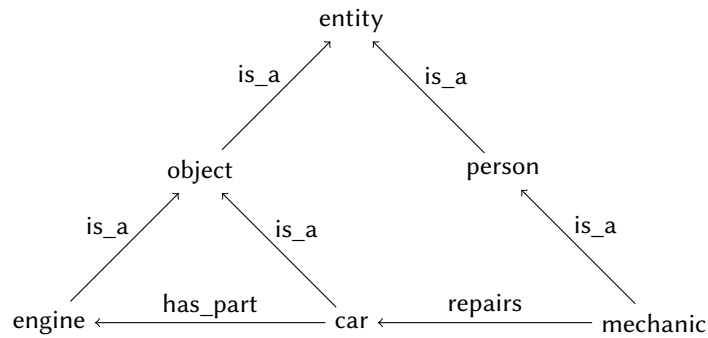


Figure 2: An example of ontology.

2.2. Constraint Satisfaction Problems

The problem of defining the correct function from papers to reviewers has to consider general constraints regarding (for instance) the minimum or maximum number of reviews per paper, conflicts between authors, the policy of the conference or workshop, but also some specific needs of some authors (e.g. availability of a reviewer for few papers). This setting comes under the Constraint Satisfaction Problem domain.

The classic definition of a Constraint Satisfaction Problem (CSP) is as follows. A CSP \mathcal{P} is a triple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ where \mathcal{X} is an n -tuple of variables $\mathcal{X} = \langle x_1, x_2, \dots, x_n \rangle$, \mathcal{D} is a corresponding n -tuple of domains $\mathcal{D} = \langle \mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_n \rangle$ such that $x_i \in \mathcal{D}_i$, \mathcal{C} is a t -tuple of constraints $\mathcal{C} = \langle \mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_t \rangle$. A constraint \mathcal{C}_j is a pair $\langle \mathcal{R}_{S_j}, \mathcal{S}_j \rangle$ where \mathcal{R}_{S_j} is a relation on

the variables in $\mathcal{S}_j = \text{scope}(\mathcal{C}_j)$. In other words, \mathcal{R}_i is a subset of the Cartesian product of the domains of the variables in \mathcal{S}_i . A solution to the CSP \mathcal{P} is an n -tuple $\mathcal{A} = \langle a_1, a_2, \dots, a_n \rangle$ where $a_i \in \mathcal{D}_i$ and each \mathcal{C}_j is satisfied in that $\mathcal{R}_{\mathcal{S}_j}$ holds on the projection of \mathcal{A} onto the scope \mathcal{S}_j . In a given task one may be required to find the set of all solutions to determine if that set is non-empty or just to find any solution, if one exists. If the set of solutions is empty the CSP is unsatisfiable [2].

In our settings, we can map \mathcal{X} , \mathcal{D} and \mathcal{C} according to the automatic paper assignment process. In this case, \mathcal{X} is the set of papers, \mathcal{D} is the power set of the reviewers ($2^{\mathcal{R}}$), and a complete assignment is a function associating papers with a set of reviewers. The set \mathcal{C} is composed of a set of constraints that guide the assignment.

We are not only interested in finding an assignment satisfying the constraints but also in finding the assignment that maximises the similarity between paper topics and reviewers' interests. This setting lies in an extension of the Constraint Satisfaction Problem which is called the Constraint Optimization Problem. Basically, a Constraint Optimization Problem is defined in the same way as a CSP one but the (*function*) assignment needs to maximise (resp. minimise) a target numerical function. Constraint Satisfaction Problems and Constraint Optimization Problems can be managed by different solvers like Gecode [3] or Chuffed [4].

2.3. Answer Set Programming

Answer Set Programming (ASP), also known as Disjunctive Logic Programming (DLP) under stable model semantics, is a powerful framework for Knowledge Representation and Reasoning. Originating from the work of Gelfond, Lifschitz, and Minker in the 1980s, ASP has gained increasing interest within the scientific community. One key factor in its success is its highly expressive language: asp programs can precisely express any property of finite structures over a function-free first-order structure that is decidable in nondeterministic polynomial time with an NP oracle, meaning asp encompasses the complexity class $\Sigma_2^P = NP^{NP}$. As a result, asp enables encoding programs that cannot be converted to SAT in polynomial time. Notably, asp is entirely declarative (the sequence of literals and rules does not affect the outcome), and its encodings for a wide range of problems are concise, straightforward, and elegant. Unfortunately, the high expressiveness of asp comes at the price of a high computational cost in the worst case, making the implementation a tough task.

2.3.1. Syntax

Following a convention dating back to Prolog, strings starting with uppercase letters denote logical variables, while strings starting with lowercase letters denote constants. A term is either a variable or a constant. An atom is an expression $p(t_1, \dots, t_n)$, where p is a predicate of arity n and t_1, \dots, t_n are terms. A literal l is either an atom p (positive literal) or its negation $\neg p$ (negative literal). Two literals are said to be complementary if they are of the form p and $\neg p$ for some atom p . Given a literal l , $\neg l$ denotes its complementary literal. Accordingly, given a set \mathcal{L} of literals, $\neg\mathcal{L}$ denotes the set $\{\neg l \mid l \in \mathcal{L}\}$. A set \mathcal{L} of literals is consistent if its complementary literal is not contained in \mathcal{L} for every literal $l \in \mathcal{L}$. A disjunctive rule (rule, for short) r is a construct: $a_1 \vee \dots \vee a_n \Leftarrow b_1, \dots, b_k, \neg b_{k+1}, \dots, \neg b_m$ where $a_1, \dots, a_n, b_1, \dots, b_m$ are literals and

$n \geq 0, m \geq k \geq 0$. The disjunction $a_1 \vee \dots \vee a_n$ is called the head of r , while the conjunction $b_1, \dots, b_k, \neg b_{k+1}, \dots, \neg b_m$ is referred to as the body of r . A rule without head literals (i.e. $n = 0$) is usually referred to as an integrity constraint. A rule having precisely one head literal (i.e. $n = 1$) is called a normal rule. If the body is empty (i.e. $k = m = 0$), it is called a fact. If r is a rule of form (1), then $\mathcal{H}(r) = \{a_1, \dots, a_n\}$ is the set of literals in the head and $\mathcal{B}(r) = \mathcal{B}^+(r) \cup \mathcal{B}^-(r)$ is the set of the body literals, where $\mathcal{B}^+(r)$ (the positive body) is $\{b_1, \dots, b_k\}$ and $\mathcal{B}^-(r)$ (the negative body) is $\{b_{k+1}, \dots, b_m\}$. An asp program (also called Disjunctive Logic Program or DLP program) \mathcal{P} is a finite set of rules. A not-free program \mathcal{P} (i.e., such that $\forall r \in \mathcal{P} : \mathcal{B}^-(r) = \emptyset$) is called positive or Horn, and a v-free program \mathcal{P} (i.e., such that $\forall r \in \mathcal{P} : |\mathcal{H}(r)| \leq 1$) is called normal logic program. In asp, rules in programs are usually required to be safe. A rule r is safe if each variable in r also appears in at least one positive literal in the body of r . An asp program is safe if each of its rules is safe. A term (an atom, a rule, a program, etc.) is called ground if no variable appears in it [5].

2.3.2. Semantics

From [6] the semantics of asp programs can be defined. For every program \mathcal{P} , its answer sets are defined by using its ground instantiation $grnd(\mathcal{P})$ in two steps: first, the answer sets of positive disjunctive programs are defined, and then, the answer sets of general programs are defined by a reduction to positive disjunctive programs and a stability condition. Indicating with $\mathcal{B}_{\mathcal{P}}$ the set of positive literals of a program \mathcal{P} , an interpretation \mathcal{I} is a consistent set of ground literals $\mathcal{I} \subseteq \mathcal{B}_{\mathcal{P}}$ w.r.t. a program \mathcal{P} . A consistent interpretation $X \subseteq \mathcal{B}_{\mathcal{P}}$ is called closed under \mathcal{P} (where \mathcal{P} is a positive disjunctive datalog [7] program), if, for every $r \in grnd(\mathcal{P})$, $\mathcal{H}(r) \cap X \neq \emptyset$ whenever $\mathcal{B}(r) \subseteq X$. An interpretation which is closed under \mathcal{P} is also called a model of \mathcal{P} . An interpretation $\mathcal{X} \subseteq \mathcal{B}_{\mathcal{P}}$ is an answer set for a positive disjunctive program \mathcal{P} , if it is minimal (under set inclusion) among all (consistent) interpretations that are closed under \mathcal{P} .

3. Related Work

Many solutions to this problem rely on semi-automatic processes. Dumais et al. [8] introduced a strategy based on a first random assignment that reviewers can evaluate and then, taking into account the feedback, use the Latent Semantic Index (LSI) [9] to measure similarity between papers and reviewers. Historically, there exists many techniques to get the hidden semantics. More common strategies are Vector Space Models [10] or Latent Dirichlet Analysis [11]. In the Knowledge Graph field, they have been overcome by graph embedding techniques [12]. Many automatic strategies still require additional information from reviewers. For instance, they are asked to select keywords from a list of standardised concepts [13]. Similarity can also be computed by processing the history of authors (publications) as in [14]. By linguistic model, it is possible to create a dataset of similarities between the author's history and papers and train a classifier.

Nguyen et al. [15] introduced an Ordered Weighted Averaging (OWA) operator to average different parameters of reviewers and authors (publications, keywords, students, etc). Similarity among papers and reviewers sometimes has been mapped as the distance between authors and

reviewers. To this extent, many graph-based algorithms are useful. For instance, co-author relationship and then use the above-mentioned graph embeddings or a cosine similarity [16].

To the best of our knowledge, Kalmugov [17] developed the algorithm that performs best, while also distributing papers in a homogeneous way and prioritising assignments for which the reviewer is highly confident with the topic. The assignment is not done singularly but as a batch, and with a greedy algorithm it gets closer to local optimization.

All these works explore a wide range of methods to compute the distance between papers and reviewers. Yet there is still the need to model constraints in the assignment process, which gives us reasons to explore a new approach.

4. Problem Modellization

Having shown the ingredients we are going to use, we describe here how we model the problem. First of all, we need a taxonomy (ontology) of disciplines in the Computer Science field. This taxonomy allows us to measure the similarity between topics. Referring to Figure 2, we have only a set of classes and the `is_a` relationship. The structure we will have is a (rooted) tree, in which the root is named `Computer Science`, and from that, all the specializations will be given. With this structure, we have a *connected* graph, in which it is always possible to find a path connecting two classes by traversing the `is_a` arcs or their inverses (indicated as `is_a-`). Hence, the algorithm used is the Shortest Path Length [18]. In our case, the Shortest Path Length coincides with the minimum number of edges (`is_a` or `is_a-`) needed to connect the two nodes in the graph. More formally, we have a tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ where \mathcal{V} is the set of nodes and $\mathcal{E} : \mathcal{V} \times \mathcal{V}$ expressing the `is_a` function. We simplify the notation $(v_1, v_2) \in \mathcal{E}$ with $v_1 - \text{is_a} - v_2$ and we generalise with `is_a*` to represent `is_a` or `is_a-`. We define a path from v_0 to v_n as $v_0 - \text{is_a}^* - v_1 - \text{is_a}^* - \dots - \text{is_a}^* - v_n$. The minimum number of needed `is_a*` is the Shortest Path Length. Given the tree structure, it is necessary to assume that $v_i \neq v_j \forall i \neq j$ in order to find the minimum path. Given \mathcal{L} the length of the taxonomy, the distance between two nodes a, b (indicated as `distance(a, b)`) always exists and $0 \leq \text{distance}(a, b) \leq 2\mathcal{L}$. The two extreme cases (`distance = 0` or `distance = 2\mathcal{L}`) happen, respectively, when $a = b$ and when a, b have only the root of the tree as a common parent.

A shared conceptualization is useful for aligning papers' topics with reviewers' ones. To this extent, the keywords are used for the papers; for the reviewers, we use the labels available on Google Scholar. Obviously, paper keywords and researchers' labels are not syntactically aligned. Here the shared conceptualization comes into play. In order to align words syntactically, we use the Large Language Model (LLM) GPT-4o [19]. The query asked to select, for every keyword in the paper (also taking into account the others) the closest category in the ontology. Quite similarly, for every Google Scholar label of researchers, we asked for the closest category in the ontology. As it happens in general with LLMs, hallucinations are a problem in the results. To mitigate this problem, we do not take the result provided by GPT-4o as it is but we try to link it to the correct ACM label. We make use of the Levehnstein distance [20] to retrieve the string that has the lowest distance with the label provided by the LLM. Given the interpretation of this distance (i.e. the number of characters to be added, removed or changed in order for two strings to be equal), we decided that if the value is greater than 40% of the length of the closest

ACM category, then the label provided by the LLM is discarded since considered unreliable. Ultimately, we have a full syntactic mapping between papers and researchers. From now on, we use keyword both for paper keywords and researchers' labels since they have been mapped within a common vocabulary.

The Constraint Optimization Problem is provided with the following variables (and domains):

- $\mathcal{P} = \{p_1, \dots, \}$ papers,
- $\mathcal{W} = \{w_1, \dots, \}$ researchers,
- $\mathcal{R} \subseteq \mathcal{W}$ reviewers,
- $\mathcal{K} = \{k_1, \dots, \}$ keywords,
- $\mathcal{L} : (\mathcal{P} \cup \mathcal{W}) \rightarrow 2^{\mathcal{K}}$ labelling function on papers and authors,
- $\mathcal{A} : \mathcal{P} \rightarrow 2^{\mathcal{W}}$ authorship function on papers,
- $\mathcal{C} : \mathcal{W} \times 2^{\mathcal{W}}$ conflicts relationship between researchers.

Having defined the distance between two keywords, we can define the distance between a paper and a researcher. Both papers and researchers are represented by sets of keywords, and, intuitively, we look for the similarity between the two sets. For every keyword of the paper, we look for the keyword of the researcher minimizing the distance. The same keyword for the researcher may be used more than once. Formally, the distance between a paper and a researcher is computed as: $\text{distance}(p, r) = \sum_{k \in \mathcal{L}(p)} \min_{l \in \mathcal{L}(r)} \text{distance}(k, l)$.

Stepping back to Figure 2, let's suppose a paper p has keywords `mechanic`, `car` and `entity`, and the reviewer r has labels `engine`, `object`, and `entity`. For the keyword `mechanic`, the closest reviewer's label is one of the three indifferently since the minimum distance is 2, for `car` the closest are `engine` and `object` with distance 1, and for `entity` we have an overlap with the reviewer's label. Hence, in this case, the distance is 0. In the end, the sum is $2 + 1 + 0 = 3$.

The objective is to find the assignment $\theta : \mathcal{P} \rightarrow 2^{\mathcal{R}}$ that minimizes $\sum_{p \in \mathcal{P}} \sum_{r \in \theta(p)} \text{distance}(p, r)$. Many constraints can be considered for this task. Given $M, N \in \mathbb{N}$, the ones we consider are:

- i. $\forall r \in \mathcal{R}, |p \in \mathcal{P} : r \in \theta(p)| \leq M$,
- ii. $\forall p \in \mathcal{P}, |\theta(p)| \geq N$,
- iii. $\forall p \in \mathcal{P}, \theta(p) \cap \mathcal{A}(p) = \emptyset$,
- iv. $\forall p \in \mathcal{P}, \theta(p) \cap \{r \mid \exists a \in \mathcal{A}(p) : r \in \mathcal{C}(a)\} = \emptyset$,
- v. $|p \in \mathcal{P} : r_i \in \theta(p)| \leq n_i, n_i < M$.

Constraint (i) guarantees that at most M papers are assigned to each reviewer so as to not overload someone. Constraint (ii) guarantees that a paper is reviewed by at least N reviewers, to guarantee a certain reliability and heterogeneity in the judgment process. Constraint (iii) guarantees that authors do not review themselves. Constraint (iv) generalizes (iii) and avoids conflicts between authors and reviewers. Constraint (v) includes a family of constraints. It

guarantees that a specific reviewer (for some reason) is not available to review more than a quantity of papers that is lower than the maximum M .

Finally, the solution to our problem is $\arg \min_{\theta} \sum_{p \in \mathcal{P}} \sum_{r \in \theta(p)} \text{distance}(p, r)$ satisfying constraints from (i) to (v).

5. Implementation

We opted for ASP to model the problem given its competitive performance, expressiveness, and optimization utilities. Clingo¹ is responsible for grounding and solving and has the capability to abduce all the possible solutions. When Clingo performs optimization, intermediate results are available. The optimal solution is not guaranteed to be unique, and Clingo provides all the possible models. For the experiments, we selected the IRCIDL dataset available from the 20th anniversary in which all the (315) publications of the first twenty editions were available. Authors, titles, and keywords were extracted from the dataset. As per reviewers, we opted for the Program Committee members of the 20th edition of IRCIDL.

In the computer science field, in order to categorize topics we used the ACM Computing Classification System². In principle, this taxonomy is not rooted but we added the root Computer Science for two reasons: (i) we can compute distances between topics not having a common ancestor, instead of complicating the notation and the management with possible undefined values, and (ii) to use the root as a collector for extremely general keywords or research areas in the field. The classification includes 1915 categories and 1928 `is_a` relationships between categories.

In the constraints, we force every paper to be reviewed by at least two authors, and every author cannot be assigned to more than four papers. For every author we have affiliations, and by general rule we consider authors to be in conflict if and only if they belong to the same affiliation (or have one affiliation in common). All experiments have been conducted on an Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz-1.50 GHz processor with 16GB of RAM. The chosen execution is multithreaded with 4 cores. In the implementation, we tended to anticipate all the possible preprocessing phases like finding ancestors in the ACM classification and conflicts among researchers. Full ASP implementation is available here (<https://zenodo.org/records/14733925>).

Listing 1 shows the ASP implementation of the automatic reviews' assignment.

Listing 1: Automatic Reviews' Assignment in ASP

```
1 %sort areas
2 ancestor(X, X, 0) :- area(X).
3 ancestor(X, Y, 1) :- parent(X, Y).
4 ancestor(X, Y, N) :- parent(X, Z), ancestor(Z, Y, N-1).
5
6 has_child_ancestor(X, Y, A) :- parent(A, B), ancestor(B, X, _), ancestor(B, Y, _).
7
```

¹<https://potassco.org/clingo/>

²<https://dl.acm.org/ccs>


```

8 %find first common ancestor
9 1 { first_common_ancestor(X, Y, A) : ancestor(A, X, _), ancestor(A, Y, _) } 1
10     :- area(X), area(Y).
11 :- first_common_ancestor(X, Y, A), has_child_ancestor(X, Y, A).
12
13 %compute distance between two areas
14 distance(X, Y, D1 + D2) :-
15     first_common_ancestor(X, Y, A),
16     ancestor(A, X, D1), ancestor(A, Y, D2).
17
18 %conflicts between researchers of the same affiliation
19 conflict(X, Y) :- X != Y, affiliation(X, U), affiliation(Y, U).
20
21 %maximum of articles to assign to everybody
22 max_articles(4).
23
24 %number of reviews
25 n_reviews(2).
26
27 %minimum distance between a keyword and a reviewer
28 1 { distance_target(K, R, D) } 1 :-
29     keyword(_, K), reviewer(R),
30     D = #min { Dis, R2 : distance(K, R2, Dis),
31     research(R, R2) }.
32
33 %distance between a paper and a reviewer
34 dissimilarity_target(P, R, S) :-
35     paper(P), reviewer(R),
36     S = #sum { D : distance_target(K, R, D),
37     keyword(P, K) }.
38
39 %generate assignments
40 N { assign(P, R, S) : dissimilarity_target(P, R, S) } N :- paper(P), n_reviews(N).
41
42 %avoid conflicts
43 :- assign(P, R, _), author(P, A), conflict(R, A).
44 :- assign(P, R, _), author(P, R).
45
46 %number of assignments per reviewer
47 assigned_reviewer(R, N) :- reviewer(R), N = #count { P,R,S : assign(P, R, S) }.
48
49 %respect limit of reviews per reviewer
50 :- assigned_reviewer(R, N), assign_max(R, M), N > M.
51

```

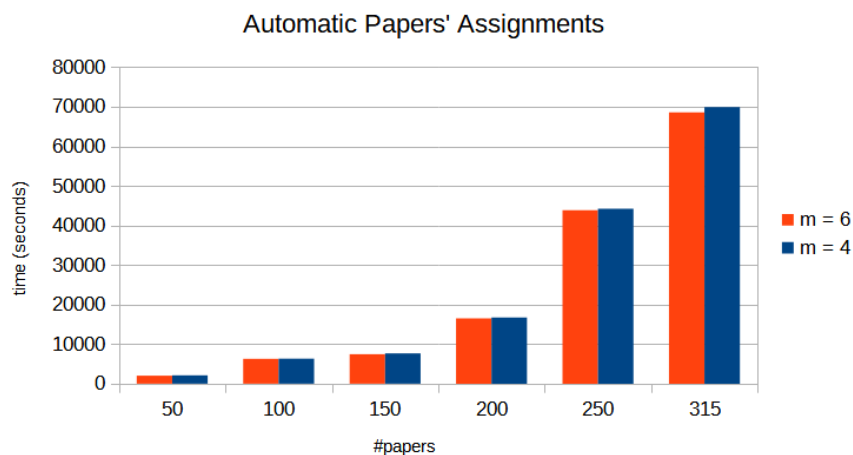


Figure 3: Execution times

```

52 %overall distance in the assignments
53 total_sum(S) :- S = #sum { N,P,R : assign(P, R, N) }.
54
55 %optimize overall distance
56 #minimize {Sum : total_sum(Sum)}.
57
58 #show assign/3.

```

6. Evaluation

For evaluation, we ran multiple experiments varying the number of maximum articles to be assigned to each reviewer and the total number of papers. We do not expect that the maximum number of articles will affect the general performance so much. Conversely, we want to verify that including constraints creates benefits in the process, and we also tested with and without the constrain to avoid conflicts. Times reported in Figure 3 consider only the time to reach the first result (local optimization). Many local optimizations can be output before reaching the global optimization but the time for moving from one optimization to the next one is quite constant.

Results show that times are more than acceptable, especially when the number of papers does not exceed 150. As expected, the number of maximum articles does not affect the results too much, although increasing that value creates benefits in general. More interestingly, removing the condition of conflicts is unhelpful when the number of papers increases, as shown in Figure 4. Our suspect is that removing the computation of conflicts is for sure beneficial at the beginning, but having more pruning helps in the coupling between papers and reviewers by removing lots of possible combinations. To date, it is not possible to quantitatively assess the accuracy of the match given that the dataset of IRCDL has not been labeled according to the reviewer's

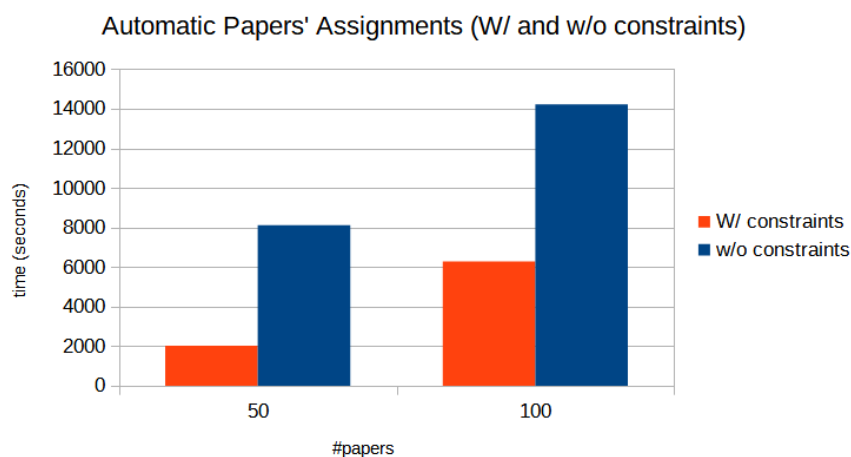


Figure 4: Execution times W/ and w/o conflicts for #papers = 50 and 100

interest. Compared to the state-of-the-art, to the best of our knowledge, this is the first approach capable of taking into account all the possible constraints given its symbolic nature. Although the complexity of ASP with optimization is $\mathcal{O}(2^n \cdot m)$ where m is the number of constraints, it may be worth using it instead of heuristics-based algorithms that cannot take many constraints (apart from the maximum number of assigned papers) into account. It is likely that for huge conferences this approach may be limiting, but it could be used in combination with existing approaches when, for example, allocating fewer papers that are not easy to assign, or solving specific conflicts during assignments.

The problem of assigning reviews can be generalized as a classification problem. When thinking about digital libraries in the general run of things, it is possible to categorize documents into categories. For instance, it is possible to map categories of documents with categories in the context of specific libraries. Given the physical limitations of libraries, constraints come into play also in this context. Apart from ontology mapping which is always time-consuming, the general methodology is reusable and the new constraints can be adapted without many difficulties.

7. Conclusions and Future Work

In conclusion, in this paper, we showed an alternative modelization in the process of automatic paper assignments to reviewers. This strategy, given its symbolic nature, can take into account realistic constraints for the conferences, solve conflicts automatically, and introduce personalization in the decision process. Experiments demonstrated that for not big conferences the performances are more than acceptable, showing also how increasing the number of constraints affects execution in general.

We look forward to receiving the results of the assignment process by asking IRCDL reviewers to evaluate the quality of the assignment process. Extensions of this work are the evaluation of

other datasets, for which accuracy can be computed, and explore also different strategies (e.g. pure CSP) and solvers.

References

- [1] N. Guarino, C. Welty, A formal ontology of properties, in: International Conference on Knowledge Engineering and Knowledge Management, Springer, 2000, pp. 97–112.
- [2] F. Rossi, P. Van Beek, T. Walsh, Handbook of constraint programming, Elsevier, 2006.
- [3] C. Schulte, M. Lagerkvist, G. Tack, Gecode, Software download and online material at the website: <http://www.gecode.org> (2006) 11–13.
- [4] R. van Driel, N. Yorke-Smith, Towards unsatisfiable core learning for chuffed, in: CP’20 Workshop on Progress Towards the Holy Grail, 2020.
- [5] P. Bonatti, F. Calimeri, N. Leone, F. Ricca, Answer set programming, A 25-Year Perspective on Logic Programming: Achievements of the Italian Association for Logic Programming, GULP (2010) 159–182.
- [6] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New generation computing* 9 (1991) 365–385.
- [7] T. J. Green, S. S. Huang, B. T. Loo, W. Zhou, et al., Datalog and recursive query processing, *Foundations and Trends® in Databases* 5 (2013) 105–195.
- [8] S. T. Dumais, J. Nielsen, Automating the assignment of submitted manuscripts to reviewers, in: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, 1992, pp. 233–244.
- [9] B. Rosario, Latent semantic indexing: An overview, *Techn. rep. INFOSYS 240* (2000) 1–16.
- [10] K. Erk, Vector space models of word meaning and phrase meaning: A survey, *Language and Linguistics Compass* 6 (2012) 635–653.
- [11] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, *Journal of machine Learning research* 3 (2003) 993–1022.
- [12] H. Cai, V. W. Zheng, K. C.-C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, *IEEE transactions on knowledge and data engineering* 30 (2018) 1616–1637.
- [13] S. Price, P. A. Flach, Computational support for academic peer review: A perspective from artificial intelligence, *Communications of the ACM* 60 (2017) 70–79.
- [14] L. Charlin, R. Zemel, The toronto paper matching system: an automated paper-reviewer assignment system (2013).
- [15] J. Nguyen, G. Sánchez-Hernández, N. Agell, X. Rovira, C. Angulo, A decision support tool using order weighted averaging for conference review assignment, *Pattern Recognition Letters* 105 (2018) 114–120.
- [16] F. Rahutomo, T. Kitasuka, M. Aritsugi, et al., Semantic cosine similarity, in: The 7th international student conference on advanced science and technology ICAST, volume 4, University of Seoul South Korea, 2012, p. 1.
- [17] Y. Kalmukov, An algorithm for automatic assignment of reviewers to papers, *Scientometrics* 124 (2020) 1811–1850.

- [18] G. Yu, J. Yang, On the robust shortest path problem, *Computers & operations research* 25 (1998) 457–468.
- [19] J. A. Baktash, M. Dawodi, Gpt-4: A review on advancements and opportunities in natural language processing, *arXiv preprint arXiv:2305.03195* (2023).
- [20] F. P. Miller, A. F. Vandome, J. McBrewster, *Levenshtein distance: Information theory, computer science, string (computer science), string metric, damerau? levenshtein distance, spell checker, hamming distance*, 2009.