A Note on How to Build a Conference Schedule

The Case of ECAI-2024

Simon Rey and Ulle Endriss

Institute for Logic, Language and Computation (ILLC)
University of Amsterdam

We describe the approach we followed to construct the schedule for the 27th European Conference on Artificial Intelligence (ECAI), held in Santiago de Compostela in 2024. We provide here all the details of our approach in the hope that others facing similar challenges might be able to replicate what we did, or at least get some initial pointers for how to tackle the problem.

1 The Conference Scheduling Problem

Basic Problem. The problem we are facing is that of constructing a schedule for a conference, *i.e.*, of grouping papers into sessions that are scheduled during specific timeslots.

Formally, the problem resembles a multi-constraint bin-packing problem. We have a set of n papers $\mathcal{P} = \{p_1, \ldots, p_n\}$ that have to be assigned to a set of m sessions $\mathcal{S} = \{s_1, \ldots, s_m\}$, each of which will be scheduled at a specific timeslot. We denote by $\mathcal{T} = \{t_1, \ldots, t_p\}$ the set of all p timeslots. Intuitively, a session is a group of papers and a timeslot represents a moment of the conference (think "Monday am", for instance).

A paper-to-session assignment is a mapping $\varphi : \mathcal{P} \to \mathcal{S}$ allocating each paper to a session. A session-to-timeslot assignment is a mapping $\psi : \mathcal{S} \to \mathcal{T}$ associating each session with a timeslot. A schedule is then described by the combination of a paper-to-session assignment φ and a session-to-timeslot assignment ψ. We denote it by $\pi = (\varphi, \psi)$.

We slightly abuse notation for all mappings and use them interchangeably with their inverses. So for a paper assignment φ and a session $s \in \mathcal{S}$, the term $\varphi(s)$ represents the set of papers assigned to s. Similarly, for a session assignment ψ and a timeslot $t \in \mathcal{T}$, the term $\psi(t)$ denotes the set of sessions assigned to t. We also sometimes interpret $\pi = (\varphi, \psi)$ as the composition of φ and ψ. Following that interpretation, $\pi(p)$ represents the timeslot a paper $p \in \mathcal{T}$ is scheduled in. Similarly, $\pi(t)$ is the set of papers scheduled in a given timeslot $t \in \mathcal{T}$.

Note that, strictly speaking, there would be no need to decompose a schedule into a paper assignment and a session assignment. One could directly map papers to sessions that each have a fixed timeslot. However, for ECAI-2024, we found it helpful to decompose the problem in this manner. It allowed us to first focus on the problem of grouping papers into sessions,

and then on the problem of scheduling those sessions—even if not every possible grouping can later be turned into a schedule that respects all side constraints.

Scheduling Constraints. In addition to the basic problem description, certain extra constraints need to be imposed. Specifically, we considered the constraints described below.

- A session $s \in \mathcal{S}$ can only fit a certain number of papers. We denote by $s^+ \in \mathbb{N}$ this upper limit. For any suitable paper assignment φ we should have $|\varphi(s)| \leq s^+$ for all $s \in \mathcal{S}$.
- Authors presenting the papers have personal constraints making them unavailable to present during certain timeslots. For each paper $p \in \mathcal{P}$, we denote by $\perp_p \subseteq \mathcal{T}$ the set of timeslots during which that paper cannot be scheduled. For any suitable schedule π we should thus have $\pi(p) \notin \perp_p$ for all $p \in \mathcal{P}$.
- Some authors may present more than one paper. For every paper $p \in \mathcal{P}$, denote by $\|_p$ the set of papers presented by the same presenter as p. Then, for any suitable schedule $\pi = (\varphi, \psi)$ we should have that for all papers $p \in \mathcal{P}$ and $p' \in \|_p$ either $\pi(p) \neq \pi(p')$ or $\varphi(p) = \varphi(p')$ (either not the same timeslot or the same session).

2 Automatic Generation of a Schedule

For ECAI-2024 we had close to 600 papers ($n = 600$) to be assigned to 55 sessions ($m = 55$) distributed over 8 timeslots ($p = 8$). Due to the size of the conference, it would have been particularly tedious to design a schedule by hand. Moreover, due to the broad scope of the conference, it would have been impossible for a small group of experts to adequateley judge the topical coherence of the individual sessions in any proposed schedule. Next, we describe the approach we followed to construct the schedule in a semi-automatic manner. We present the approach in general terms, ignoring the specificities of ECAI-2024.

2.1 Assessment of a Schedule

Throughout the whole process we needed a way to assess the quality of the schedules we were building. To do so, we considered pairwise scores between any two papers, which we interpreted as a proxy for the utility an attendee of the conference would enjoy if those two papers were to be presented in the same session. These scores can be thought of as similarity scores between papers. They eventually get aggregated to assess the quality of an entire schedule.

Similarity Scores for Papers. The score is modelled by means of a scoring function $score : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ such that the pairwise score of any two papers $p, p' \in \mathcal{P}$ is $score(p, p')$. This function is symmetric, *i.e.*, $score(p, p') = score(p', p)$ for all $p, p' \in \mathcal{P}$.

For ECAI-2024 we determined the pairwise score between any two papers by using the topics selected by the authors of a paper at submission time and the bidding data submitted by the reviewers. Specifically, for any two papers $p, p' \in \mathcal{P}$ the bidding similarity and the topic

similarity are defined as follows:

$$sim_{bidding}(p, p') = \frac{\text{Number of reviewers bidding for both } p \text{ and } p'}{\text{Number of reviewers bidding for either } p \text{ or } p' \text{ or both}},$$

$$sim_{topic}(p, p') = \frac{\text{Number of topics selected for both } p \text{ and } p'}{\text{Number of topics selected for either } p \text{ or } p' \text{ or both}}.$$

These two similarity functions thus correspond to Jaccard distances between the corresponding sets. We then defined the final pairwise score between any two papers $p, p' \in \mathcal{P}$ as follows:

$$score(p, p') = \begin{cases} \lfloor sim_{bidding}(p, p') \rfloor \times 1000 & \textit{if } sim_{bidding}(p, p') > 0.05, \\ \lfloor sim_{topic}(p, p') \rfloor \times 100 & \textit{else if } sim_{topic}(p, p') > 0.035, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, we gave more importance to bidding similarity than topic similarity. We interpreted bidding similarity as follows: a reviewer—who we are using as a proxy for a conference attendee—bidding for two given papers will likely also be happy to see those two papers being presented in the same session. In case this similarity is likely to encode noise rather than actual information (if the similarity is very low), we consider the similarity in terms of topics. Finally, we force a score of 0 in case both similarity measures are very low. The values of 0.05 and 0.035 used as thresholds were hand-picked in view of observed behaviour.

Comparing Schedules. In order to be able to compare schedules, one needs to aggregate the pairwise scores. The first aggregation happens at the paper-level where we use the average aggregator. The score of a paper $p \in \mathcal{P}$ for a given paper assignment φ in which $\varphi(p) = s$ is:

$$score(p, \varphi) = \frac{1}{|\varphi(s)|} \times \sum_{p' \in \varphi(s)} score(p, p').$$

To compare two schedules we then used leximin comparisons. Consider a paper assignment φ. We introduce the vector $\vec{score}(\pi) = (score(p, \varphi))_{p \in \mathcal{P}}$ and the vector $\vec{score}^{\uparrow}(\varphi)$ corresponding to $\vec{score}(\varphi)$ ordered from the lowest value to the highest value. Then, a paper assignment φ is preferred to another paper assignment φ', denoted by $\varphi \succ \varphi'$, whenever $\vec{score}^{\uparrow}(\varphi)$ leximin-dominates $\vec{score}^{\uparrow}(\varphi')$, *i.e.*, when there exists a position $k \in \{1, \ldots, n\}$ such that $\vec{score}^{\uparrow}(\varphi)_k > \vec{score}^{\uparrow}(\varphi')_k$ and for all $k' \in \{1, \ldots, k-1\}$ it is the case that $\vec{score}^{\uparrow}(\varphi)_{k'} = \vec{score}^{\uparrow}(\varphi')_{k'}$. In words, the two vectors $\vec{score}^{\uparrow}(\varphi)$ and $\vec{score}^{\uparrow}(\varphi')$ have the same values for the first $k-1$ positions and for the kth position, $\vec{score}^{\uparrow}(\varphi)$ is better.

Schedules are compared by means of their paper assignments. For any two schedules $\pi = (\varphi, \psi)$ and $\pi' = (\varphi', \psi')$ we have $\pi \succ \pi'$ whenever $\varphi \succ \varphi'$. Indeed, according to the interpretation we have of the score, the session-to-timeslot assignment ψ should not have any impact on this score.

Comparing paper assignments, and thus schedules, in leximin terms means that emphasis is put on trying to cater for the "worst-off" papers. This may for instance imply that some papers may be assigned to less-fitting sessions in order to help other papers reach a higher score.

The Goal. The idea thus is to find good schedules as defined above. In practice we did not go fully with a straightforward optimisation approach. This is due to the size of the problem, but also to the limitations of the score, as we shall discuss in some detail later on.

2.2 Paper Assignment

To construct paper assignments, *i.e.*, to allocate papers to sessions, we found that the solution yielding the best results was to rely on clustering algorithms to detect groups of papers that are similar. Next, we present the various steps for this process.

Iterative Merge Clustering. The idea in this first procedure is to merge papers that should clearly be grouped together within the same session. We use a clustering algorithm for this purpose. The first step is thus to define a semimetric over the space of papers (a "metric" that does not satisfy the triangle inequality). We based the one we used on the pairwise score defined above, which intuitively encodes similarity between papers. Specifically, for any two papers $p, p' \in \mathcal{P}$ we used the following notion of distance:

$$dist(p, p') = \begin{cases} 1 - \frac{score(p,p')}{\max(score(p'',p''')|p'',p'''\in\mathcal{P})} & \text{if } p \neq p', \\ 0 & \text{otherwise.} \end{cases}$$

We used this semimetric together with the k-means algorithm to compute clusters of papers. The exact procedure is described in Algorithm 1.

Algorithm 1 Iterative Merge Clustering

```
Initialise papers with all papers
Initialise new_papers with the empty set
Let α be the maximum number of papers that can be merged into one
Let β be the ratio of |papers| that defines the maximum number of clusters computed
while True do
    for n_clusters = 2 to ⌊β × |papers|⌋ do
        Cluster the set papers into n_clusters clusters using k-means
        for each cluster c do
            if the total combined number of papers in c is α or less then
                Merge the papers in c into p*
                Remove the papers in c from papers
                Add p* to new_papers
        if a cluster has been merged then
            Break from the for-loop
    if no cluster has been merged then
        if |new_papers| > 0 then
            Add the content of new_papers to papers
            Assign the empty set to new_papers
        else
            Break from the while-loop
```

This procedure iteratively merges papers by identifying clusters of papers and merging those

that have a suitable size. Newly created "papers" (representing small clusters of actual papers) are added into the mix when no clusters have been merged for all potential numbers of clusters. This allows for them to be merged again further down the procedure. The procedure stops when nothing new has happened for a full round.

The value of α is the typical number of papers per session, which was 12 in the case of ECAI-2024. The value of β should be somewhat small since the k-means algorithm always returns the required number of clusters (so it may find clusters just to satisfy the constraint, even if they are not "good" clusters). We used $\beta = 0.1$.

For every merged paper, the score function $score$ is extended so that the score of a merged paper $p^\star$ for another paper p is the average of the scores of the papers contained in $p^\star$. In particular, this implies that $score(p^\star, p^\star)$ does not have to be 0 as we need to account for the score yielded by the papers composing $p^\star$ being together.

Greedy Assignment to Sessions. Once the paper merging is done, papers need to be assigned to sessions. This was done in a greedy fashion. Papers are iteratively assigned to sessions. At each iteration, the paper that yields the best new score of the schedule (in terms of the leximin comparison) is added to the corresponding session. The assignment of paper to session respects the size constraint: no paper is added if a session contains already a certain number of papers (12 in the case of ECAI-2024). This takes into consideration the fact that some "papers" have a larger weight than others due to the merging.

After the first greedy assignment, papers that were merged during the clustering step have to be split up again. The greedy assignment was then run a second time to assign papers that were not assigned in the first run of the procedure (because their weight was too large to fit into any session, for instance).

Local-Search Improvements. Two local-search routines were then applied to the schedule in order to improve it. The first one is a reassignment procedure in which a paper is reassigned to another session that is not yet full. This step is repeated iteratively, and at each iteration the reassignment that would lead the best schedule score is applied. Only reassignments that would not violate the session capacity constraints were taken into account.

The second local-search procedure we applied is a swap procedure in which two papers from two different sessions are exchanged. Once again, at every iteration the swap leading to the best schedule score was implemented.

Of course one could do more than reassignments and swaps, but in practice these two local-search procedures were leading satisfactory results. As we will discuss later on, focusing too much on trying to achieve the perfect score may not be desirable because of the shortcomings of the scoring function.

2.3 Sessions Scheduling

Once the sessions have been constructed, *i.e.*, once we have a paper assignment φ, they need to be assigned to timeslots to obtain an actual schedule. We computed the session-to-timeslot assignment via an Integer Linear Program (ILP).

For every session $s \in \mathcal{S}$ and timeslot $t \in \mathcal{T}$, we introduce a binary variable $x_{s,t} \in \{0, 1\}$ indicating that session s has been assigned to timeslot t. We then consider the following constraints:

- All sessions have to be assigned to exactly one timeslot, so for all session $s \in \mathcal{S}$, we have:

$$\sum_{t \in \mathcal{T}} x_{s,t} = 1;$$

- Availability constraints for authors are to be respected, so for all timeslot $t \in \mathcal{T}$ and paper $p \in \mathcal{P}$ such that $t \in \perp_p$, then we have:

$$x_{\varphi(p),t} = 0;$$

- Authors cannot be in two parallel sessions at the same time, so for all papers $p \in \mathcal{P}$ and $p' \in \|_p$, if $\varphi(p) \neq \varphi(p')$, then for every timeslot $t \subseteq \mathcal{T}$, we have:

$$x_{\varphi(p),t} + x_{\varphi(p'),t} \leq 1.$$

This ILP might not have a solution and some of the constraints can be turned into objectives to minimise: minimising the number of authors needing to move between session during a single timeslot for instance. For ECAI-2024 we were able to satisfy all of these constraints as we only received a small number of requests regarding availability constraints (probably because we announced that we would not be able to promise respecting them).

For ECAI-2024, we also assigned "tracks" to the sessions (*e.g.*, Computer Vision, Natural Language Processing). We then used extra constraints forcing not to have too many sessions in parallel for each track. We typically forced not having more than one session in parallel for a track (especially those with "small" numbers of sessions).

After this last step, a schedule is constructed and the problem is solved.

2.4 Cosmetic Improvements and Other Tools

Once the schedule for ECAI-2024 had been built, we developed a number of additional tools to improve its presentation and help with the rest of the process.

Representative Topics of a Timeslot. At submission time, authors were asked to selected a number of topics for their papers (from a fixed but very long list). For each session we then selected a set of representative topics. This is a simple covering problem that we solved using an ILP. For each timeslot, the objective of the ILP was as follows:

- Find a smallest set of topics X that covers all the papers, *i.e.*, that has the property that each paper presented in the session has at least one of its topics in X.

- Amongst all the smallest set of topics covering all the papers select the one that has the highest cumulative coverage, *i.e.*, the highest sum over all papers presented in the session

of the number of selected topics that are also assigned to the paper (so a paper covered by multiple topics will impact the score multiple times).

By displaying the set of representative topics in the schedule, we hoped to make it easier for audience members to identify the sessions they want to attend. We also hoped that it would help onlookers to rationalise why a given number of papers had been clustered together into a session.

Suggestion of Session Chairs. Each session needs to be assigned a chair. To help with this task we again used the bidding data. We considered the set of individuals already registered for the conference (around 6 weeks before the conference) who were also members of the programme committee. For each session we then made a shortlist including only individuals who had bid for papers presented in that session. From this list, we hand-picked individuals to invite to chair that particular session. Most of these invitations were accepted, which allowed us to very quickly assign chairs to around half of all sessions, and to do so with individuals who are especially interested in and qualified to chair those sessions.

It is worth noting that even with around 150 PC members having registered at the time of the session chair assignment (and 55 sessions to find a chair for), there was a significant number of sessions that did not include any papers anyone attending the conference had bid for. In those cases, we used the traditional approach of contacting all remaining registered PC members and asking for volunteers.

Estimated Attendance of a Session. Sessions have to be assigned to the available rooms, which typically will not all have the same size. To help with this task we tried to estimate the likely attendance of session. We did so by using the bidding data once more.

Assuming that the reviewers are representative of the attendees of the conference, for each timeslot we estimated the probability for each reviewer to attend each session scheduled during that timeslot. To do so, we considered all the papers assigned to a given session. The probability for a reviewer to attend a given session would then be proportional to the number of papers presented in that session that the reviewer had bid on. One can run this exercise either with the set of all reviewers or only with the set of reviewers who also registered to attend. In the case of ECAI-2024, both approaches yielded broadly similar predictions.

Using this approach, we were able to compare the relative popularity of sessions and thus assign the most popular sessions to the biggest rooms.

Of course, likely popularity is not the only consideration. It also makes sense to try and assign sessions belonging to the same track to the same room.

3 Limitations of the Approach

The most stringent limitation is the fact that this approach is not fully automated: there are many choices that are made by hand and that can only be made after trying out different possibilities. In particular:

- Something that really helped us was to run the process, select some good sessions to lock in, and then to run the process again. Iterating the process like this allowed us to identify strong sessions and to avoid some of the "bad" choices that can be made throughout the process.
- The mixing of the different similarity scores was tested by hand to select the most promising way of combining them. In particular, the decision of focusing mostly on the bidding data came from the observation that it seemed more reliable than the topic data.
- The similarity scores can be very noisy as they are based on very partial information. For instance, the bidding data is in itself biased as not all reviewers would have looked through all papers. The topic data should also be considered carefully as all authors may not have the same understanding of what falls under a given topic. Moreover, since the list of topics is pre-determined, authors may not be able to select the perfect topic for their paper and thus may choose related-but-not-fully-fitting topics.
- Because of the noise in the pairwise scores, optimising the scoring function does not necessarily lead to great schedules. In practice, some papers are a great fit together even though this is not reflected by the scores. Human intervention is often needed here.

One of the main lessons we learned by designing the schedule for ECAI-2024 is that the absence of perfectly reliable information makes it impossible to have a fully automated solution. There thus is need for regular human input. Still, developing this approach allowed us to be more efficient in the process. This, hopefully, led to the design of a good schedule for the conference.