mathematics

IMPACT FACTOR 2.592

CITESCORE 2.9

Article

# A Track-Based Conference Scheduling Problem

Fabian Riquelme, Elizabeth Montero, Leslie Pérez-Cáceres and Nicolás Rojas-Morales

MDPI

https://doi.org/10.3390/math10213976

*mathematics*

MDPI

*Article*

# A Track-Based Conference Scheduling Problem

**Fabian Riquelme [1], Elizabeth Montero [2,*], Leslie Pérez-Cáceres [3] and Nicolás Rojas-Morales [1]**

[1] Departamento de Informática, Universidad Técnica Federico Santa María, Valparaíso 2390123, Chile
[2] Facultad de Ingeniería, Universidad Andres Bello, Viña del Mar 2531015, Chile
[3] Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile
* Correspondence: elizabeth.montero@unab.cl

**Abstract:** The scheduling of conferences is a challenging task that aims at creating successful conference programs that fulfill an often wide variety of requirements. In this work, we focus on the problem of generating conference programs that organize talks into tracks: subevents within the conference that are group-related talks. The main contributions of this work can be organized into three scopes: literature review, problem formulation and benchmarking, and heuristic approach. We provide a literature review of conference scheduling approaches that organizes these approaches within a timetabling problem taxonomy. We also describe the main characteristics of the conference scheduling approaches in the literature and propose a classification scheme for such works. To study the scheduling of conferences that include tracks, we introduce the definition of the track-based conference scheduling problem, a new problem that incorporates tracks in the conference program. We provide a binary integer linear programming model formulation for this problem. Our formulation considers the availability of presenters, chairs, and organizers, the avoidance of parallel tracks, and best paper sessions, among other classical constraints of conference scheduling problems. Additionally, based on our formulation, we propose a simple instance-generation procedure that we apply to generate a set of artificial instances. We complete our work by proposing a heuristic method based on the simulated annealing metaheuristic for solving the track-based conference scheduling problem. We compare the results obtained by our heuristic approach and the Gurobi solver regarding execution time and solution quality. The results show that the proposed heuristic method is a practical approach for tackling the problem as it obtains solutions in a fraction of the time required by Gurobi, while Gurobi is also unable to obtain an optimal solution in the defined time for a subset of the instances. Finally, from a general perspective, this work provides a new conference scheduling problem formulation that can be extended in the future to include other features common in conference programs. Moreover, thanks to the instance generation procedure, this formulation can be used as a benchmark for designing and comparing new solving approaches.

**Keywords:** conference scheduling; track-based conference scheduling; integer linear programming model; simulated annealing

**MSC:** 90C27



**Citation:** Riquelme, F.; Montero, E.; Pérez-Cáceres, L.; Rojas-Morales, N. A Track-Based Conference Scheduling Problem. *Mathematics* **2022**, *10*, 3976. https://doi.org/10.3390/math10213976

Academic Editor: Ripon Kumar Chakrabortty

Received: 30 September 2022
Accepted: 21 October 2022
Published: 26 October 2022



## 1. Introduction

Globalization has fostered the organization of large scientific events in which the scientific community can gather to share their work and discuss new ideas. Many scientists worldwide regularly attend these conference events since they are great opportunities for showcasing their work and networking within the scientific community. Organizing such large conferences is often very challenging since their scheduling must consider several aspects and meet a large set of requirements. Furthermore, technology allows the organization of conference events in hybrid formats, in which attendees may join the event through the internet, and contributions can be presented online and onsite. Nowadays, such formats are highly attractive as they allow the submission of contributions and registration

of attendees that otherwise could not be included in the event. All this makes conference planning an increasingly relevant and interesting subject of study. The classical approach to scheduling a conference consists of bringing together a working group of people that, with whiteboard and paper in hand, organizes and distributes talks over time slots and locations. In its beginnings, in the absence of video conferences, the organizers had to meet in person or write long emails to carry out appropriate planning of the event [1]. This approach can work for small conferences, where human capacity can still obtain good results in a reasonable time. However, when large-size conferences are considered, the planning results in a challenging task.

In this work, we focus on the problem of organizing a conference, particularly the problem of generating the program of a conference event such that it fulfills a set of requirements. The *conference scheduling problem* is a specific problem that belongs to a large family of scheduling problems, the timetabling problems, which include both task assignment and resource allocation. To relate this work with others in the literature, we extend the classification commonly used in the literature for timetabling problems to a taxonomy that includes conference scheduling problems. We aim to provide a methodology for classifying conference scheduling problems and analyzing similarities between related problems and our approach. The taxonomy organizes timetabling problems based on the nature of the main elements to be scheduled. We complement the taxonomy with a classification of conference scheduling problems that considers different aspects of the scheduling. Finally, we perform a literature review describing a set of relevant conference scheduling works and classify them based on their main characteristics.

Conferences often divide the planning day into sessions which may contain several talks and activities such as discussion panels, meetings, and ceremonies, among others. Generating a conference schedule involves assigning these sessions to a set of time slots and locations. The requirements for the scheduling vary widely depending on the event to be organized. These requirements include the attendants' availability and preferences, the distance between related sessions, the location capacity, and the grouping of related talks. In this work, we propose a *track-based conference scheduling problem* that considers subevents within the conference, which are called *tracks*. This problem was inspired by the Genetic and Evolutionary Computation Conference (GECCO) case study. We provide a mathematical formulation of the proposed problem, an instance generation procedure, and a heuristic algorithm to perform the scheduling.

The main contributions of this work are:

- A taxonomy and classification of the existing approaches for conference scheduling;
- The proposal/definition of the track-based conference scheduling problem;
- A binary integer linear mathematical formation for the presented problem;
- A set of instances and an instance generation procedure;
- A simulated annealing-based approach to solve the proposed problem and computational experiments that evaluate the proposed method.

This article is organized as follows, Section 2 provides a literature review of formulations and solving approaches for the conference scheduling problem. We provide a taxonomy of timetabling problems, incorporating conference scheduling problems within the literature and classifying them based on their main characteristics. Section 3 presents our track-based conference scheduling problem proposal and provides a mathematical formulation for the problem. Then, a heuristic approach to solve the proposed problem is described in Section 5. An instance generation procedure is presented in Section 6, and experimental results and their analysis are provided in Section 7. Finally, we present our conclusions in Section 9.

## 2. Literature Review

Planning problems involve allocating resources, locations, and time slots to specific tasks or events. We classify planning problems based on whether the main focus of the scheduling is a set of tasks or events. *Tasks* are activities that must share a set of

locations and resources in a planning horizon, often satisfying a set of precedence rules derived from a goal associated with their completion [2,3]. For example, some planning problems can be related to scheduling a production process comprising several tasks or rescheduling the production in a dynamic environment considering setup time and limited resources (machines and setup workers) [4]. *Events* are activities that must share locations, resources, and participants in a planning horizon. The scheduling of events often requires satisfying exclusivity rules regarding the availability of participants [5,6]. For example, problems related to scheduling the exams of a term in a university. Many similarities can be found between task and event-based planning problems. For example, the multiprocessor task assignment problem can be analogous to a parallel room event assignment problem. Furthermore, the topic-based conference scheduling problem can be considered as scheduling tasks with precedence where one session must come before another, and the next cannot be executed until the previous one is completed.

Despite the wide range of applications of timetabling problems in the literature, little effort has been made to create a general taxonomy for organizing these problems. The lack of such taxonomy is probably due to the large number of variants that can be considered based on each problem's particular constraints. We propose a taxonomy of timetabling problems in Figure 1, which defines two main categories within these problems: event-based problems or task/work-based problems.

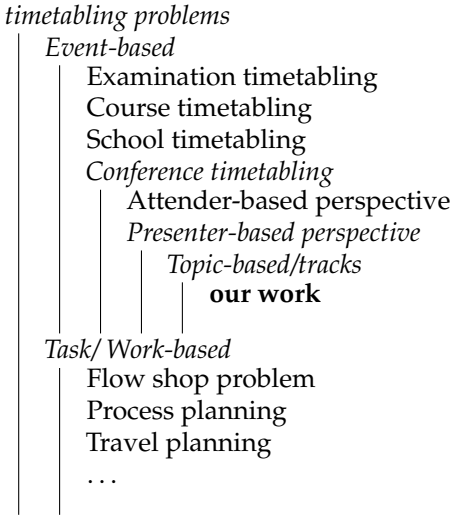


**Figure 1.** Timetabling problems taxonomy.

Our taxonomy puts the conference scheduling problem under the category of event-based timetabling problems together with several academic event planning problems. Such event-based problems range from meetings, championships [7], exhibitions [8], and bus schedule planning [9]. The taxonomy distinguishes between problems formulated with an attender-based or a presenter-based perspective within conference timetabling problems [10]. The attender-based problems seek to plan the conference, maximizing the attendees' satisfaction with the conference program. We note that these types of conference timetabling problems are also known as preference-based problems in the literature. On the other hand, presenter-based problems seek to assign talks into sessions such that the conference program avoids presenters having different talks scheduled simultaneously.

Conference events can be very diverse regarding their program structure and requirements. In conference timetabling problems, a *conference program* consists of a two-dimensional grid that allocates *talks*, commonly organized in *sessions*, to time-space dimensions during the days of the conference event. We define talks as all the contributions or interventions to be scheduled in the conference. Commonly, talks are presentations of the research works selected to be included in the conference. Each talk is associated with one or more *authors*, and it will be presented by (at least) one of them. Spacewise,

sessions are assigned to a set of available *rooms* (or locations). These rooms may have some *features* (usually named facilities) that should be considered in the scheduling (e.g., room capacity and size, among others). Timewise, sessions are assigned to time intervals with a predefined duration; we refer to these time intervals as *blocks*. A conference day is thus divided into blocks, while these blocks are further divided into *time slots*. Sessions can coincide in the same block but in different rooms, these are called *parallel sessions*. Finally, each talk within a session is assigned to a time slot within its block.

In this work, we propose a new conference timetabling problem that can be classified as a presenter-based approach. This problem introduces the notion of *tracks* which compose the conference program. Tracks can be defined as subevents within the conference in which several topic-related works will be presented. The proposed problem considers scheduling requirements associated with speakers, session chairs, and track organizers, hence its classification in the presenter-based category.

The outlined problem is further introduced in Section 3. In the following, we present a literature review of conference scheduling problems.

### 2.1. Attender-Based Approaches

The work in [10] proposes an attender-based approach for scheduling conference sessions to maximize public attendance. The primary objective of this approach is to schedule the talks so that attendance is maximized, enabling the public to attend their preferred articles. The secondary objective is to minimize the number of *jumps* between sessions that occur when an attendee must leave a session to attend another talk in a parallel session. The work described in [10] solved the planning of three different conferences: MathSport, Models and Algorithms for Planning, and Scheduling Problems (MAPSP), and the Belgian Conference on Operations Research (ORBEL). The author proposed a solution based on a two-stage algorithm that used linear programming models and heuristics. The first stage was responsible for building a set of parallel "tuples" of talks that minimized the costs associated with nonattendance. This process was done considering a vector of preferences of each attending person. As a result of this stage, $N$ vectors of parallel talks were obtained. The second stage searched for the parallel talk vectors' schedule that minimized the jumps between sessions. A local search method that exchanged talks between or within the same session was applied. The article concluded that the ORBEL conferences were the most complex instances studied. These conferences considered four parallel sessions and approximately 80 talks to schedule. The results obtained by the proposal reached locally optimal solutions in at most a minute of computation time.

In [11], an extensive study of attender-based planning was presented. The study considered decision strategies such as time flexibility, presenter preferences, talks, sessions, blocks, and spatial and temporal adjacency of talk scheduling. The same work defined the characteristics and decisions that must be taken when planning a conference. The first characteristic was determining the flexibility of the planning, that is, how redundant the conference can be with respect to repeated sessions or talks, the number of sessions, and the number of available time slots. The second characteristic consisted in defining preference types, considering the preferences on time schedules, internal composition of sessions, or spatial adjacency. The problem described in this work was the *2001 Annual Meeting of the Decision Sciences Institute* conference. The case study considered 213 sessions, ten available time slots, and the preferences of 520 people. To tackle the scheduling, the authors proposed a simulated annealing algorithm that required around 40 min to solve the proposed problem.

### 2.2. Presenter-Based Approaches

In [12], the authors considered as a case study the European Conference on Operational Research (*EURO-K*), which is one of the largest conferences in the world (https://www.euro-online.org/web/pages/312/euro-k-conferences (accessed on 24 October 2022)).

In particular, the problem presented in the article was the *EURO-K 2016* planning. The conference had about 2000 participants and approximately 2000 articles to be exhibited, with a total of 463 sessions throughout 25 study areas. In this conference, a set of study areas were defined based on the articles' keywords. These study areas can be considered close to the definition of tracks in our work. The problem of EURO-K considered a conference program that aimed at satisfying multiple needs and objectives for a set of articles being scheduled to a set of rooms, buildings, time slots, days, and people. To solve the problem, first, the organizers of a particular study area generated each of the sessions beforehand and internally plan them; that is, they determine the order of the articles of each session. The organizers created groups of sessions (called *streams*) that had to be planned in sequence in the same room and ideally should not be interrupted. The available rooms might be located in different buildings. We note that most of the *microplanning* was done by the organizers and not by the proposed solution approach. Then, the problem was reduced to assigning (uninterrupted) streams to rooms in different buildings. The authors applied an integer linear programming model that sought to optimize a list of objectives lexicographically. First, it minimized the number of sessions scheduled in different buildings that belonged to the same study area. Second, the number of related study areas allocated to the same building was maximized. Third, it minimized the number of different rooms allocated to each stream. Fourth, the interruptions in the schedule of each stream (time gaps) were minimized. Moreover, it maximized the number of seats available in rooms based on previous attendance records and preferences. The formulation divided the problem into small subproblems, where, for example, the third objective could be solved independently for each building. Experimental results showed that the proposed approach only took a few hours to obtain a solution that substantially met each of the goals listed above.

In [13], the authors studied a problem similar to the one addressed in our work. The case study considered was the *International Conference on Production Research 2018*. Based on this case study, problem instances were generated considering between 120 and 240 articles to be presented. Unlike our work (and the EURO-K), this type of conference did not have a simile of what we call tracks. This work aimed to generate a schedule for the conference, grouping similar articles in sessions with related topics. The talks were grouped by analyzing their similarity based on keywords and available information from the review process. A linear programming model and a heuristic approach based on GRASP and column generation were presented. The proposed heuristic managed to reach its solutions in less than 1 min while a time limit of 1 h was fixed for the mathematical solver (Gurobi) for each problem instance.

In [14], a presenter-based approach that sought to cluster talks within sessions according to their content (e.g., keywords and title) was presented. The clustering was performed constraining cluster size and was evaluated based on the schedule generated by the organization. Two new modifications to clustering algorithms were proposed, one using a linear programming model and another based on a modification of the *K-medoids* algorithm. The article considered the planning of the *13th International Conference on Machine Learning and Applications (ICMLA-14)* and the *27th and 28th Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence (AAAI-13, AAAI-14)*. We note that these conferences included a large variety of topics and did not consider the concept of tracks defined in our work. The proposed solution involved using techniques such as natural language processing followed by clustering algorithms. According to experimental results, the proposed approaches achieved suitable schedules.

In [15], a new version of the problem that considered the perspective of the organizers, participants, and presenters was studied. The problem definition used in this work sought to plan talks with similar topics in joint sessions. The study presented three mathematical models. The first was a linear programming model that could solve small and medium instances. The second was a linear programming approach that modeled the problem as a grouping problem that could solve larger instances than the first approach using *branch and cut* techniques. Finally, the third approach was a clustering model with *set partitioning*

solved by a *branch, cut, and price* algorithm. Real and artificial instances derived from the *XV Latin American Robotics Symposium (LARS)* and the *Brazilian Logic Conference (EBL)*, involving 86 and 96 talks, respectively, were used for evaluation. A time limit of 12 h for the three models was established. The first two models did not manage to solve the relaxation of the problem of the largest artificial instances, while the last model had a much better performance requiring at most 5 h of computation time in the most complex cases.

As already noted, the problem proposed in this work can be classified as a presenter-based approach. Hence, the problem includes a presenter perspective in the formulation and other approaches described in this section. Presenter requirements are handled differently by each approach. The availability of presenters, as well as organizers, is considered in our formulation. We note that the track concept introduces the organizers' role in the scheduling. Even though parallels to the track concept can be found in other approaches [13–15], these tracks are commonly built by the scheduling process, and thus, they are not considered subevents with an internal organization. In our approach, attendance is ensured by considering both availability and commitments across the conference. Thus, tracks are not defined as independent subevents within the conference, as is the case in [12]. In this sense, the proposed problem is unique as it defines tracks (subevents) and considers attendance in a conferencewide fashion. The following section provides more details regarding the main features of the approaches in the literature, highlighting the differences and similarities between them and our work.

### *2.3. Classification*

This section classifies the previously reviewed works which tackle different conference scheduling problems. Table 1 presents the classification of the reviewed works and our problem highlighted in bold. The classification considers a set of characteristics of the problems tackled in each work: the perspective (attender-based or presenter-based), the criterion for grouping talks (topic-based, keyword-based, or track-based), the problem divisibility, the problem objective(s), and the proposed approach (exact or local search methods). Additionally, we identify the real-world conference that was considered a reference to define the conference scheduling problem tackled in each work. Most of the reviewed works can be classified as presenter-based approaches. Regarding the grouping of talks, all reviewed works use either a topic-based or keyword-based approach, while our work introduces the concept of tracks. Only the work presented in [12] can be classified as a divisible problem since the conference should be scheduled in different locations (buildings), which can define an independent scheduling problem per location. Exact methods are the most applied in the reviewed works, while in some cases, a local search procedure is also applied to generate the scheduling. In this context, we classify our work as a presenter-based approach with a track-based strategy for grouping talks, defining a nondivisible problem that minimizes the missing seats and proposes exact and local search approaches. We note that the definition of tracks as subevents is relevant to our work as it differs from the talk-grouping strategy considered by other approaches. We highlight that tracks are independent, having, for example, their organizers and chairs. Despite this, the generated problem is nondivisible as the talks within a track are not scheduled independently for attendance purposes. This feature implies that the people considered in the scheduling are assumed to be interested in attending different tracks within the conference.

**Table 1.** Classification of the reviewed works. In the table header: Perspective—(A): Attender-based or (B): Presenter-based; Talks grouping—(C): Topic-based, (D): Keyword-based or (E): Track-based; Solving method—(F): Exact Search or (G): Local Search.

| Approach | Based on | Perspective | | Talks Grouping | | | Divisibility | | Objective | Solving Method | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | (A) | (B) | (C) | (D) | (E) | Yes | No | | (F) | (G) |
| [10] | ORBEL, MathSport, MAPSP | X | | - | - | - | | X | Maximize attendance, minimize jumps | X | X |
| [11] | 2001 Annual Meeting of the Decision Science Institute | X | | - | - | - | | X | Minimize the redundancy of the scheduling, maximize the preference satisfaction | | X |
| [12] | Euro-K | | X | | X | | X | | Minimize the number of sessions in buildings, maximize the correlation between study areas and buildings, minimize the number of rooms to be used per stream, minimize streams interruption, maximize the available seats in rooms | X | |
| [13] | International Conference on Production Research 18′ | | X | X | X | | | X | Maximize similarity between talks on same sessions | X | X |
| [14] | ICMLA, AAAI | | X | | X | | | X | Maximize intrasessions similarity | X | |
| [15] | LARS, EBL | | X | X | | | | X | Maximize intrasessions similarity | X | |
| **Our work** | **GECCO** | | **X** | | | **X** | | **X** | **Minimize the number of missing seats** | **X** | **X** |

## 3. Track-Based Conference Scheduling Problem

The track-based conference scheduling problem is a problem in which a set of talks must be assigned to a set of locations and time slots. Some conferences define *tracks* as subevents within the conference or groups of talks that share similar topics of interest. Each talk is associated with a particular track. In our formulation, tracks are considered to be predefined by the conference having talks assigned to tracks beforehand. Consequently, the association of a talk to a track is fixed and cannot be modified during the scheduling process. A track is organized into one or more sessions that group its talks. The assignment of sessions to time blocks and their talks to time slots within the corresponding blocks define the *conference program*.

Our work considers the 2019 Genetic and Evolutionary Computation Conference as a case study. The GECCO event comprises several subevents, including tracks, workshops, and other events within the conference. For simplicity, in the following, we consider all these subevents as tracks. Tracks are organized by one or more people responsible for coordinating and decision-making during the organization process. Each track plans a set of talks. These talks can be invited talks or presentations of research works submitted and accepted for the track. Tracks group their talks in sessions moderated by a session chair. In GECCO, some tracks nominate submitted works for the best paper award; thus, some talks in the conference are the best paper nominees (BP talks). Generally, BP talks within a track are assigned to the same session. Sessions that contain BP talks are deemed best paper sessions (BP sessions). In some cases, joint BP sessions are created, including nominees of more than one track. For this, a previous agreement between track organizers is made. Attendees vote during the conference for their best paper favorites; consequently, special attention is given to minimizing the number of BP sessions scheduled in parallel.

GECCO 2019 included 173 talks, 56 sessions, 16 tracks, 7 time blocks distributed over three days, and eight rooms. More information can be found on the conference website (https://gecco-2019.sigevo.org/index.html/HomePage (accessed on 24 October 2022)). Figure 2 shows the summary of the GECCO 2019 conference program. The diagram has two dimensions, a spatial dimension denoted by the rooms (rows of the table) and a temporal dimension indicating the time blocks available for the schedule (columns of the table). The sessions of each track are shown in a different color. Sessions that include best paper nominee talks are marked with a star in the diagram.

| | Day 1 Block 1 | Day 1 Block 2 | Day 1 Block 3 | Day 2 Block 1 | Day 2 Block 2 | Day 2 Block 3 | Day 3 Block 1 |
|---|---|---|---|---|---|---|---|
| Room 1 | T1 session 1 | T1 session 2 ★ | T8 session 1 ★ | T5 session 4 ★ | Joint session T11 session 3 T7 session 3 ★ | T1 session 4 | T1 session 5 |
| Room 2 | T2 session 1 ★ | T15 session 1 | T3 session 3 ★ | Joint session T14 session1 T12 session 2 T9 session 2 ★ | T13 session 2 ★ | T4 session 5 ★ | T3 session 4 |
| Room 3 | T3 session 1 | T3 session 2 | T13 session 1 | T8 session 2 | T8 session 3 | T8 session 4 | T8 session 5 |
| Room 4 | T6 session 1 | T6 session 2 | T6 session 3 | T16 session 1 | T16 session 2 | T16 session 3 | T16 session 4 |
| Room 5 | T10 session 1 | T9 session 1 | T7 session 1 | T10 session 2 | T5 session 5 | T9 session 3 | |
| Room 6 | T4 session 1 | T4 session 2 | T2 session 3 | T4 session 3 | T4 session 4 | T5 session 7 | T7 session 4 |
| Room 7 | T11 session 1 | T12 session 1 | T2 session 2 | T11 session 2 | T2 session 3 | T2 session 4 | T14 session 2 |
| Room 8 | T5 session 1 | T5 session 2 | T5 session 3 | T7 session 2 | T5 session 6 | T5 session 8 | T5 session 9 |

**Figure 2.** Overview of the conference program of GECCO 2019. Sessions that include best paper nominee talks are marked with a star in the diagram.

In the following, we introduce the track-based conference scheduling problem formulation. We consider a set of *talks*, each of them associated with a *track* within the

conference. These talks should be grouped into *sessions* and scheduled in a set of *rooms* and *time slots*. Each room has an associated *capacity*, measured as the number of available seats. We assume that an estimation of the number of participants per track is available. Such estimation can be obtained from historical attendance records, surveys, or attendance expectations. We also consider a set of actors involved in the schedule: the presenter of each talk, the session chairs, and the track organizers. The *presenters* correspond to the people in charge of presenting the talks. Each of them may present one or more talks at the conference. The *organizers* have mostly an administrative role, but it is sought that at least one organizer can be present in the sessions of their respective tracks. The *chairs* correspond to the masters of ceremonies for each session and must be present for the entire session. The actors involved in the schedule might be unavailable in specific time slots due to external reasons. Figure 3 describes the relationship of the elements considered in the scheduling. We define that each talk has an associated presenter, and each talk belongs to one session that, in turn, belongs to one track. Note that sessions are not predefined, and thus, the assignment of talks to them is part of the scheduling process. We include in our formulation the notion of *best paper nominee talks*. Sessions that contain such BP talks are regarded as *BP sessions*, and their parallel scheduling should be minimized.

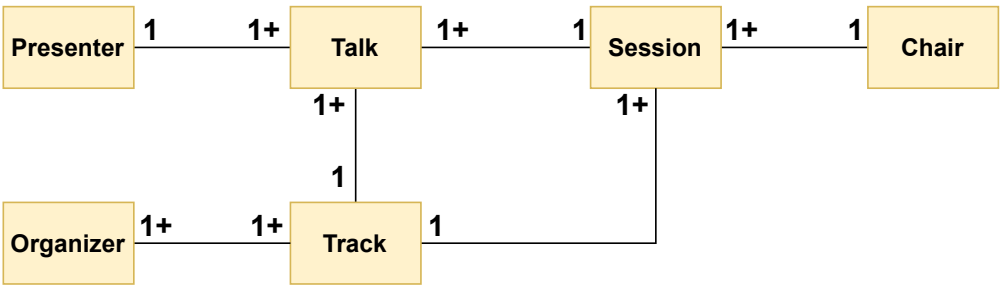


**Figure 3.** Entity–relationship model diagram of the elements considered in the track-based conference scheduling problem formulation.

The objective of our formulation is to find a conference schedule that assigns talks to sessions, and sessions to rooms and time slots *minimizing the number of missing seats* in the sessions. The number of missing seats is calculated based on track attendance estimated based on historical attendance records. Other requirements of the problem are formulated as constraints. Some of these aspects are: ensuring the availability of at least one session chair and organizer per session, ensuring the availability of the presenters in their talks, avoiding sessions of the same track scheduled in parallel, and avoiding best paper sessions planned in parallel.

Figure 4 shows an example of a conference schedule that considers three tracks, each shown in a different color (purple, green, and blue). The example conference considers 16 talks associated with these tracks, grouped into six sessions, using two rooms and three blocks. It is considered that each session can contain (at most) three talks, i.e., three slots. In this example, talks A, B, C, L, M, F, G, and H belong to the *purple* track. We note that scheduled sessions are scattered throughout different blocks, with talks taking place in parallel throughout several rooms. As already mentioned, having sessions of the same track taking place in parallel should be avoided so that the public interested in a given track can attend all its sessions.

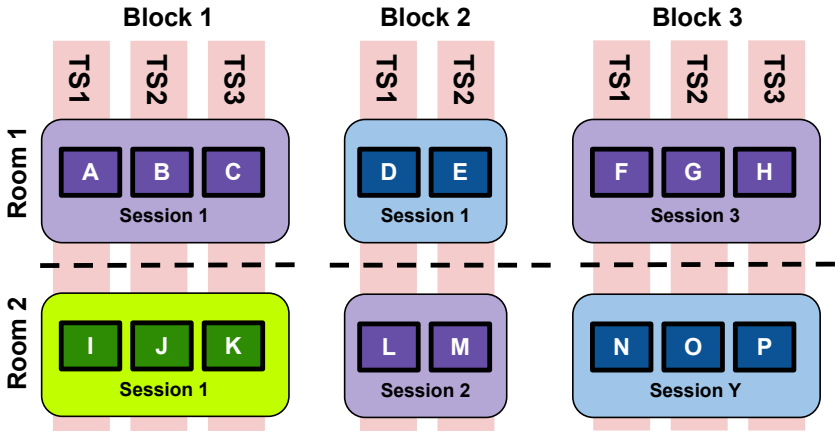


**Figure 4.** Example of a conference with 16 talks, three tracks, three blocks and two rooms.

## 4. Mathematical Model

This section presents the mathematical model of the track-based conference scheduling problem. The conference timeline is divided into a set of time blocks. A grid cell refers to a particular block and room. We consider that sessions are logically divided into a fixed number $N$ of *time slots* to which talks should be assigned.

### *4.1. Variables*

Our formulation defines the following decision variables:

$$x_{\tau sl} \in (0,1) = \text{Talk } \tau \text{ is assigned to session } s \text{ and slot } l \quad (1)$$

$$w_{psl} \in (0,1) = \text{Person } p \text{ presents in session } s \text{ slot } l \quad (2)$$

$$c_{ps} \in (0,1) = \text{Person (chair) } p \text{ participates in session } s \quad (3)$$

$$o_{ps} \in (0,1) = \text{Person (organizer) } p \text{ participates in session } s \quad (4)$$

$$y_{srb} \in (0,1) = \text{Session } s \text{ is scheduled to grid cell } (r,b) \quad (5)$$

$$\sigma_{pslrb} \in (0,1) = \text{Person } p \text{ participates in slot } l \text{ of session } s \text{ scheduled in grid cell } (r,b) \quad (6)$$

Variable (1) defines the talks that belong to a session and their order within the session. Variable (2) indicates that presenters should be available for their talks. Variable (3) and (4) constrain the chairs and organizers to be available for their sessions. Variable (5) defines a session schedule in a grid cell. We define auxiliary variable (6) to represent that the same person cannot be simultaneously in two places.

### *4.2. Parameters*

Our formulation defines the following parameters:

$N \in \mathbb{N}$: Maximum number of talks per session.

$\Gamma_\tau \in \mathbb{N}$: Track to which talk $\tau$ belongs.

$\Lambda_s \in \mathbb{N}$: Track to which session $s$ belongs.

$T_{st} \in (0,1)$: Session $s$ belongs to track $t$.

$C_r \in \mathbb{N}$: Capacity of room $r$.

$F_{sr} \in (0,1)$: Session $s$ can be allocated to room $r$.

$H_s \in \mathbb{N}$: Historical attendance of session $s$.

$$\Delta_{sr} \in \mathbb{N} \begin{cases} H_s - C_r, & \text{if } H_s > C_r \\ 0, & \text{otherwise} \end{cases}$$

$B_\tau \in (0,1)$: Talk $\tau$ is nominated to best paper award.

$Author(\tau)$: Presenter of talk $\tau$.
$Chairs(s)$: Set of available chairs for session $s$.
$Organizers(s)$: Set of available organizers for session $s$.
$Persons$: Set of persons that participate in the event. $Persons$ considers the union of the sets $Author(\tau)$, $Chairs(s)$, and $Organizes(s)$.
$Unavailability(p)$: Set of time blocks where person $p$ is unavailable to participate (present/chair/organize).
$Talks(t)$: Set of talks of track $t$.
$Sessions(t)$: Set of sessions of track $t$.
$BestPaper()$: Set of talks nominated to best paper.
$Y_t$: Number of sessions of track $t$.
$\mathcal{T}$: Set of talks.

$N$ defines the maximum number of talks to be assigned to each session (number of time slots). The model defines this parameter as fixed. However, if sessions of different sizes are required, it would suffice to extend this parameter to define the number of time slots per block ($N_b$) and modify the constraints accordingly to update the model. $\Gamma_\tau$ and $\Lambda_s$ define the track to which a talk and a session belong, respectively. $C_r$ establishes the capacity of a room expressed as the number of available seats. $F_{sr}$ defines if a session can be allocated to a particular room based on the session requirements and the facilities available in a room. $H_s$ is the number of expected participants per session. This value can be estimated from historical attendance records or can be estimated by the organizers based on the expected attendance. $\Delta_{sr}$ defines the expected missing seats when session $s$ is scheduled in room $r$. $B_\tau$ is a binary constant that indicates if talk $\tau$ is nominated for the best paper award or not.

$Author(\tau)$ identifies the presenter of talk $\tau$. $Chairs(s)$ and $Organizers(s)$ identify the sets of available chairs and organizers for session $s$ of a particular track, respectively. The set $Persons$ considers the union of the sets $Author(\tau)$, $Chairs(s)$, and $Organizes(s)$. $Unavailability(p)$ identifies the set of time slots where a person $p$ (presenter, chair or, organizer) is unavailable. $Talks(t)$ and $Sessions(t)$ identify the set of talks and sessions of track $t$, respectively. $BestPaper()$ lists the set of talks nominated for the best paper award.

We define $Y_t$ as the fixed number of sessions per track. Nevertheless, the number of required sessions can be computed for each instance as the number of sessions needed to plan all the talks of each track (Equations (7) and (8)).

$$Y_t = \left\lceil \frac{Talks(t)}{|\mathcal{T}|} \right\rceil \tag{7}$$

$$\text{The total number of sessions} = \sum_t Y_t \tag{8}$$

Thus, the number of sessions is considered a constant value throughout the resolution of the linear programming model.

### *4.3. Constraints*

The constraints related to the track-based conference scheduling problem are listed below. Constraint (9) establishes that each talk must be scheduled exactly once. Constraint (10) controls the maximum number of talks $N$ that can be allocated to each session. Constraint (11) establishes that each slot can be allocated to at most one article.

$$\sum_{s}\sum_{l} x_{\tau sl} = 1 \quad \forall \tau \tag{9}$$

$$\sum_{\tau}\sum_{l} x_{\tau sl} \leq N \quad \forall s \tag{10}$$

$$\sum_{\tau} x_{\tau sl} \leq 1 \quad \forall s, l \tag{11}$$

Constraints (12) and (13) indicate that each session can be allocated to a unique grid cell $(r, b)$, and each grid cell can be used by at most one session. Constraint (14) forbids the assignment of a talk to a session that belongs to a different track. Constraint (15) controls the relationship between the scheduling of a talk and its presenter in the same session. Constraint (16) indicates that all best paper talks of the same track must be scheduled in a unique session. Constraint (17) avoids the simultaneous scheduling of sessions that belong to the same track. Sessions can be allocated to the same block using the set of available rooms. To avoid same-track sessions in the same time slots, the constraint accounts only for sessions in a given track using the parameter $T_{st}$, allowing at most one in each time slot.

Constraint (18) avoids the schedule of talks/sessions to unavailable time blocks of presenters, chairs, and organizers. In this case, we assume that there is a list of unavailable time blocks for each person $p$ participating in the conference.

$$\sum_{r}\sum_{b} y_{srb} = 1 \quad \forall s \tag{12}$$

$$\sum_{s} y_{srb} \leq 1 \quad \forall r, b \tag{13}$$

$$\sum_{l} x_{\tau sl} = 0 \quad \forall \tau, s, \Gamma_\tau \neq \Lambda_s \tag{14}$$

$$x_{\tau sl} \leq w_{psl} \quad \forall \tau, s, l, p \in Author(\tau) \tag{15}$$

$$\sum_{l} x_{\tau sl} = 1 \quad \forall t, \tau \in BestPaper(), s \in Sessions(t) \tag{16}$$

$$\sum_{s}\sum_{r} y_{srb} \cdot T_{st} \leq 1 \quad \forall t, b \tag{17}$$

$$w_{psl} + y_{srb} \leq 1 \quad \forall p, s, l, r, b \in Unavailability(p) \tag{18}$$

Constraint (18) forbids the assignment of a session $s$ to a block $b$ (which is unavailable for person $p$) when person $p$ is presenting a talk in any slot $l$ of session $s$. We note that person $p$ could have a presentation in one or more slots of session $s$, and thus, there is a constraint per slot. For example, assume that person $p_1$ is not available in the first block $b_1$ of the conference ($Unavailability(p_1) = \{b_1\}$), and their talk is assigned to session $s_2$, which has two slots, $l_1$ and $l_2$. Then, for each room $r$ of the conference, two constraints are generated:

$$w_{p_1,s_2,l_1} + y_{s_2,r,b_1} \leq 1 \tag{19}$$

$$w_{p_1,s_2,l_2} + y_{s_2,r,b_1} \leq 1 \tag{20}$$

Constraints (21)–(24) avoid a person simultaneously attending more than one session. Constraint (21) specifies that each person participates in only one session in each time slot. Auxiliary variables $\sigma_{pslrb}$ are used to ensure the linear nature of the formulation of these constraints. Variables $\sigma_{pslrb}$ activate variables $w_{psl}$ and $y_{srb}$ using constraints (22)–(24).

$$\sum_{s}\sum_{r} \sigma_{pslrb} \leq 1 \quad \forall p, b, l \tag{21}$$

$$w_{psl} + y_{srb} - \sigma_{pslrb} \leq 1 \quad \forall p, s, l, r, b \tag{22}$$

$$\sigma_{pslrb} \leq w_{psl} \quad \forall p, s, l, r, b \tag{23}$$

$$\sigma_{pslrb} \leq y_{srb} \quad \forall p, s, l, r, b \tag{24}$$

Constraints (25)–(28) indicates that a particular chair and organizer should attend their associated sessions.

$$y_{srb} \leq \sum_{p} c_{ps} \quad \forall s, r, b, p \in Chairs(s) \tag{25}$$

$$N \cdot c_{ps} \leq \sum_{l} w_{psl} \quad \forall p, s \tag{26}$$

$$y_{srb} \leq \sum_{p} o_{ps} \quad \forall s, r, b, p \in Organizers(s) \tag{27}$$

$$N \cdot o_{ps} \leq \sum_{l} w_{psl} \quad \forall p, s \tag{28}$$

### *4.4. Objective Function*

The objective function considers the minimization of the missing seats of the program schedule, according to the historical attendance of track sessions. The idea here is to perform the best allocation of sessions to available rooms minimizing the missing seats in all sessions. In other words, we aim to reduce, as much as possible, the number of persons who cannot attend a session due to a lack of seats in the assigned rooms. In a real-world situation, the requirements defined by previous attendance records may not be met by the rooms currently available.

Moreover, a variance can be expected between the historical attendance record and the attendance of the current conference event. A quiz can be performed during the conference registration to reduce the expected variation and improve the quality of the solutions and the use of available rooms. For this, $\Delta_{sr}$ indicates the difference between the expected attendance and room capacity. The objective function summarizes these differences in Equation (29).

$$\min z = \sum_{s} \sum_{r} \sum_{b} y_{srb} \cdot \Delta_{sr} \tag{29}$$

## 5. Heuristic Approach

The heuristic solution proposed in this work searches for good-quality solutions by allowing the algorithm to traverse infeasible areas of the search space. An initial session assignment is generated and submitted to a local search algorithm based on simulated annealing (SA) to improve its quality. SA [16] is a well-known metaheuristic that has been used to solve many different problems such as routing problems [17–20], symbolic regression [21], feature selection and/or hyperparameter tuning for classification algorithms [22–24], influence maximization on social networks [25], and many other problems [26,27]. Furthermore, SA has been implemented for solving many different scheduling problems related to machine scheduling problems [28], scheduling of relief teams in natural disasters [29], for the multiobjective job-shop problem [30], in scheduling tasks in cloud computing applications [31], among others.

The SA algorithm implements proportional cooling, periodic overheating, stagnation detection mechanisms, and an adaptive setting of initial temperatures.

### *5.1. Solution Representation*

Solutions are represented as matrices of size rooms $\times$ blocks. The value of each matrix cell indicates the session allocated to a room $i$ in block $j$. Zero values indicate that no session is allocated to a specific cell.

Figure 5 shows an example of a solution representation. The schedule represented by this solution considers five time blocks, three rooms, and five tracks (light blue, yellow, purple, red, and green). Each track is composed of a different number of sessions; for example, the track shown in light blue has four sessions. The value 0 indicates no session assigned to that particular combination of room and time block.

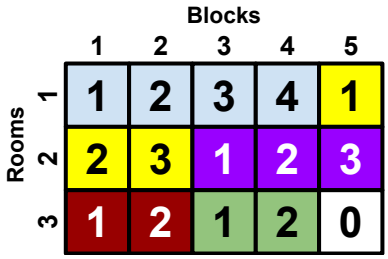


**Figure 5.** A program built by sequential assignment left-right top-bottom (the numbers represent the order of insertion in the grid).

### *5.2. Initial Solution Generation*

The initial session assignment lists the whole set of talks $\mathcal{T}$, grouped only by *track*. $Y_t$ is computed as previously explained in Section 4, and the total number of sessions required to schedule the conference is then computed. The talks are assigned to sessions according to the following criteria:

1. All best paper talks are assigned to the same session.
2. The remaining talks are iteratively assigned to sessions such that the papers maximize the joint availability of the session. The joint availability of a session is expressed as the total number of slots to which talks can be assigned.
3. When a talk can not be assigned to any session, given that it causes the schedule to be unfeasible due to null joint availability. The talk is assigned to the first session with enough space.

An initial solution is generated by populating the program representation matrix with the sessions ordered by track. The sessions are assigned sequentially by room, going from left to right through the time blocks. Tracks are randomly selected to be allocated. For example, in Figure 5, the sessions of each track were assigned sequentially, starting with the light blue track and then the yellow, purple, red, and green tracks. This procedure allows initially assigning tracks to the minimum number of rooms and reducing the number of parallel session assignments for each track. We note that this procedure does not consider the sessions' availability, and thus, it can generate infeasible programs, assigning sessions to their unavailable blocks. Moreover, the process does not consider the objective of minimizing the number of missing seats; thus, it can assign sessions to rooms that largely do not provide the number of seats required for the session.

Once all talks are assigned to sessions, one chair and organizer are randomly allocated to each session.

### *5.3. Evaluation Function*

This work's search task corresponds to solving a constraint satisfaction problem that also seeks to minimize the missing seats according to historical attendance. We apply the evaluation function given in Equation (30), which is composed of the sum of the objective function $z$ defined in Equation (29) plus a penalization function related to the unsatisfied hard constraints weighted by a factor ($\gamma > 1$).

$$f(x) = z(x) + \eta\gamma \tag{30}$$

Considering a complete instantiation, where all variables have a value assigned, the total number of unsatisfied hard constraints $\eta$ is calculated as the sum of the following elements:

- The number of persons in blocks of time they are not available.
- The number of pairs of sessions that share a chair or organizer scheduled in parallel.
- The number of pairs of the same track sessions scheduled in parallel.
- The number of presenters scheduled as chair/organizer and presenter in parallel.
- The number of presenters that have two talks programmed in the same time slot.

### *5.4. Local Search*

Algorithm 1 shows the pseudocode of our proposal. Considering the initial session assignment, the algorithm constructs an initial solution ($S$). Then, the algorithm locally searches for neighbors choosing one of four different operators (lines 4–5). Each operator has a probability $p_j$ of being selected. The details of each operator are presented in the next section. Solutions that improve the best-so-far solution are always accepted, while worsening solutions are accepted according to the classical Metropolis acceptance criterion defined by simulated annealing approaches (lines 6 to 11). A geometric cooling scheme is applied to reduce the temperature in each iteration by a factor of $\alpha$. The temperature is increased in each reheat following a linear behavior proportional to the number of iterations. $T_{i+1}$ += $\beta - (\beta - 1) \cdot \frac{i}{N}$, where $i$ is the current iteration and $N$ is the total number of iterations to be performed by the search procedure. As the process advances, reheating is iteratively less intense (line 13).

**Algorithm 1:** Simulated annealing procedure outline.

```
S = initial solution;
T_i = initial temperature;
for i = 1 → N do
    O = choose which operator(p_1, p_2, p_3, p_4);
    S_n = choose random from neighborhood(S, O);
    Δ = f(S_n) - f(S);
    if Δ < 0 then
        S = S_n
    else
        if exp(-Δ / T_i) > rand(0, 1) then
            S = S_n
        end
    end
    if χ iterations without improvement then
        T_{i+1} += β - (β - 1) · i / N
    end
    T_{i+1} = T_i · α
end
```

The proposed algorithm considers four operators in the local search process: (1) talk swap, (2) chair exchange, (3) organizer exchange, and (4) session swap. In each iteration, one operator is randomly selected with probability $p_j$ ($j \in [1, 4]$).

### *5.5. Talk Swap*

Two random sessions of the same track are chosen; a talk is randomly selected for each of these sessions, and then these talks are swapped. The objective of this operator is to perturb the session assignment to avoid the premature stagnation of the search process.

### *5.6. Chair Exchange*

A session is randomly selected, and the chair assigned to this session is replaced by a randomly selected chair of the respective track. This operator allows a search for a good session chair assignment that covers an entire time block (a column within the grid).

### *5.7. Organizer Exchange*

A randomly selected organizer is reassigned to a randomly selected session. This operator allows a search for an organizer assignment in which organizers do not have more than one responsibility in the same time block.

### *5.8. Session Swap*

Two random sessions from the grid are selected, and their rooms and time blocks $(r, b)$ are exchanged. We note that sessions can belong to any track. This operator contributes to the exploration of the search space, and it can be expected to generate most of the improvements in the early stages of the algorithm.

The three first operators (talk swap, chair exchange and organizer exchange) perform changes within a program session, while the last operator (session swap) performs changes between program sessions. Consequently, the first three operators perform small changes to a solution (complete variable instantiations). On the other hand, the session swap operator performs the largest modifications since it changes the schedule of many talks at the same time.

## 6. Experimental Setup

In this section, we present our experimental setup. We first introduce the problem-instance generation process and then describe the parameter tuning process performed for the heuristic approach.

### *6.1. Instance Generation*

To evaluate our proposal, we implemented a track-based conference scheduling instance generation procedure. The instance generator was programmed in Python 3.9.5, and its input features were the number of talks $|\mathcal{T}|$ , the number of people involved in the schedule $P$, the number of blocks $B$, the number of rooms $R$, the number of tracks $T$, the number of talks per session $L$, and the percentage of unavailability $C$.

We generated a set of test instances considering dimensions similar to GECCO 2019. These instances were generated using a fixed number of talks and tracks, while other parameters were set to different values. The total number of instances generated was determined by the permutation of the following parameters:

- Number of talks: set to the number of talks in GECCO 2019 = 173 talks.
- Number of tracks: set to the number of talks in GECCO 2019 = 14 tracks.
- Number of persons: set to either 130, 150, or 170 people to be considered for the schedule.
- Number of rooms: set to either five, six, seven, eight, or nine rooms available. We note that GECCO 2019 defined eight rooms.
- Number of blocks: set to either six, seven, eight, or nine time blocks. We note that GECCO 2019 defined seven time blocks.
- Talks per session: set to either four or five talks per session.
- Percentage of unavailability: set to either 10%, 20%, or 30% of the overall time slots for each person considered in the schedule. From preliminary experiments, we concluded that any value greater than 30% often led to infeasible conference schedules and, thus, infeasible problem instances.

The total number of generated instances could be computed by $\mathcal{T} \cdot R \cdot B \cdot L \cdot P \cdot C = 5 \times 4 \times 2 \times 3 \times 3 = 360$. Some of these instances were discarded because they considered a number of sessions higher than the size of the program grid (time slots $\times$ rooms), making them clearly unfeasible. Finally, a subset of 45 instances was considered for the experiments in our study. In the following, instances are identified by the parameter values provided as input for its generation. For example, instance $\mathcal{T}$173-P170-R9-B6-L4-T14-C0.2 considers 173 talks, 170 persons, 9 rooms, 6 blocks, 4 talks per session, 14 tracks and 20% of unavailability. The problem instances used in this study are available in [32].

### *6.2. Parameter Tuning*

We applied the *ParamILS* [33] configurator to adjust the parameters of our heuristic approach before performing the experiments to evaluate its performance. ParamILS is an iterated local search algorithm designed to search for high-performing parameter settings. The advantage of using such a procedure is that, in general, configurators require evaluating

fewer instances than traditional manual approaches, and it optimizes the computational effort involved in adjusting parameters. We provided ParamILS 10,000 target algorithm executions as the computational budget available to perform the tuning of our algorithm. Each execution of our heuristic approach applied a termination criterion of 10 iterations. The tuning was performed over a training set composed of four instances. These instances were selected since, experimentally, they appeared to be difficult to solve by traditional exact approaches. The instances chosen for training are listed in Table 2.

**Table 2.** Details of the training instances provided for the tuning process.

| | Talks | Tracks | Persons | Rooms | Blocks | Time Slots | Unavailability |
|---|---|---|---|---|---|---|---|
| (0) | 173 | 14 | 130 | 5 | 8 | 5 | 30% |
| (1) | 173 | 14 | 170 | 6 | 9 | 4 | 20% |
| (2) | 173 | 14 | 130 | 9 | 6 | 4 | 30% |
| (3) | 173 | 14 | 150 | 7 | 6 | 5 | 20% |
| (4) | 173 | 14 | 170 | 9 | 6 | 4 | 10% |

The parameters tuned and their domains are listed in Table 3. We note that ParamILS requires all parameter domains to be categorical; thus, numerical parameters were discretized based on preliminary experiments. Additionally, ParamILS requires defining a default value for each parameter which is used to initialize the search. Default parameter values are underlined in Table 3. Finally, the values selected by ParamILS are shown in bold.

**Table 3.** Parameter values domains defined for the parameter tuning process of the simulated annealing approach. Default parameter values are underlined and selected values are shown in bold.

| Parameter | Range of Values |
|---|---|
| Cooling rate | $\alpha = \{\mathbf{0.90}, 0.95, .999\}$ |
| Penalty multiplier | $\gamma = \{100, \underline{\mathbf{1000}}, 10{,}000\}$ |
| Iterations to reheat | $\chi = \{\underline{\mathbf{1000}}, 0{,}000\}$ |
| Reheat multiplier | $\beta = \{1, 100, \underline{\mathbf{1000}}\}$ |
| Operator probabilities | $p_1 = \{0.00, \underline{\mathbf{0.25}}, 0.50, 0.75, 1.00\}$<br>$p_2 = \{0.00, \underline{\mathbf{0.25}}, 0.50, 0.75, 1.00\}$<br>$p_3 = \{0.00, \underline{\mathbf{0.25}}, 0.50, 0.75, 1.00\}$<br>$p_4 = 1.00 - p_1 - p_2 - p_3$ |

## 7. Experimental Results

This section presents the results obtained from the evaluation of the proposed algorithm. The implementation of the proposed algorithm is available in [32]. The algorithm was implemented in C++ and compiled using the g++ compiler (Ubuntu 6.2.0-5ubuntu12) 6.2.0. Our heuristic procedure was evaluated by performing 25 executions over each of the 45 test instances with a $10^7$ iterations termination criteria. The experiments reported in this work were executed on a 3.50 GHz i5-4690K device with 16 GB RAM. ParamILS was applied to configure the heuristic algorithm, and the parameter settings suggested by the configurator were used in the reported experiments. Our mathematical formulation was also solved by the Gurobi solver version 9.1.1 using default settings and considering a maximum execution time of 3600 s = 1 h for each problem instance. First, we attempted to solve the generated problem instances using the Gurobi solver and then, we applied the proposed heuristic algorithm to assess its performance compared to the exact solver.

### *7.1. Gurobi Results*

We applied the Gurobi solver to the 45 instances generated with the procedure described in Section 6. Tables 4–6 provide the results obtained by the solver. From these experiments, we classified the set of problem instances according to whether the solver was able to solve them. We defined the following classes: (a) *optimally solved* instances (optimal

solutions were obtained by Gurobi), (b) *unsolved* instances (Gurobi was unable to solve the instance in the provided time), or (c) *infeasible* instances (Gurobi determined infeasibility).

Table 4 shows the results obtained for the instances in which Gurobi was able to find the optimal solution. For each instance, this table shows the value of the objective function ($z$), the lower bound of $z$ determined by Gurobi (bound), the percentage difference between these values (gap), the execution time spent to find $z$ (time (s)) and the final status of the instance. We note that all instances listed in Table 4 have a gap equal to zero (Gurobi determined the optimal solution), and instances are sorted by total execution time in descending order. Given that the solver was able to solve these instances, they can be classified as "easier" generated instances. The solving times required by Gurobi differed between instances: 3 instances required times > 2700 s, 36 instances took 1800–2700 s, 7 instances took 900–1800 s, and 12 of 21 instances took 0–900 s. The solution time required for solving the instances did not seem to be influenced by the number of persons, the number of rooms, the number of blocks, or the time slots considered in the scheduling. Interestingly, most of the instances in this table have an unavailability level of 10% or 20%. Furthermore, instances at the bottom of the table tend to have a 10% of unavailability, signaling that they are generally easier to solve. We interpret this as an indication that the unavailability parameter directly influences the difficulty of the generated instance. Such influence is easily explainable since larger unavailability levels increase the probability of generating conflicts when generating a schedule. Despite this, we note that this is not always the case, as in some cases, large levels of unavailability might make the scheduling process easier as it might reduce the number of possible assignments.

**Table 4.** Gurobi results for optimally solved problem instances.

| Case | Instance | $z$ | Bound | Gap | Time (s) | Status |
|---|---|---|---|---|---|---|
| (5) | $\mathcal{T}$173-P170-R9-B6-L4-T14-C0.2 | 44 | 44 | 0 | 2983 | optimal |
| (6) | $\mathcal{T}$173-P130-R9-B6-L4-T14-C0.1 | 65 | 65 | 0 | 2915 | optimal |
| (7) | $\mathcal{T}$173-P170-R6-B9-L4-T14-C0.3 | 139 | 139 | 0 | 2541 | optimal |
| (8) | $\mathcal{T}$173-P150-R9-B6-L4-T14-C0.2 | 21 | 21 | 0 | 2521 | optimal |
| (9) | $\mathcal{T}$173-P130-R7-B6-L5-T14-C0.2 | 30 | 30 | 0 | 2184 | optimal |
| (10) | $\mathcal{T}$173-P130-R6-B7-L5-T14-C0.2 | 15 | 15 | 0 | 2165 | optimal |
| (11) | $\mathcal{T}$173-P150-R6-B9-L4-T14-C0.2 | 261 | 261 | 0 | 2095 | optimal |
| (12) | $\mathcal{T}$173-P130-R5-B8-L5-T14-C0.1 | 134 | 134 | 0 | 2094 | optimal |
| (13) | $\mathcal{T}$173-P170-R7-B6-L5-T14-C0.2 | 30 | 30 | 0 | 1655 | optimal |
| (14) | $\mathcal{T}$173-P150-R6-B7-L5-T14-C0.2 | 166 | 166 | 0 | 1498 | optimal |
| (15) | $\mathcal{T}$173-P170-R6-B7-L5-T14-C0.1 | 21 | 21 | 0 | 1450 | optimal |
| (16) | $\mathcal{T}$173-P130-R6-B9-L4-T14-C0.3 | 61 | 61 | 0 | 1448 | optimal |
| (17) | $\mathcal{T}$173-P130-R6-B9-L4-T14-C0.2 | 33 | 33 | 0 | 1381 | optimal |
| (18) | $\mathcal{T}$173-P150-R9-B6-L4-T14-C0.1 | 22 | 22 | 0 | 1186 | optimal |
| (19) | $\mathcal{T}$173-P150-R7-B6-L5-T14-C0.1 | 15 | 15 | 0 | 1111 | optimal |
| (20) | $\mathcal{T}$173-P170-R6-B7-L5-T14-C0.2 | 52 | 52 | 0 | 1067 | optimal |
| (23) | $\mathcal{T}$173-P170-R5-B8-L5-T14-C0.2 | 41 | 41 | 0 | 834 | optimal |
| (30) | $\mathcal{T}$173-P150-R5-B8-L5-T14-C0.2 | 52 | 52 | 0 | 633 | optimal |
| (31) | $\mathcal{T}$173-P170-R6-B9-L4-T14-C0.1 | 76 | 76 | 0 | 632 | optimal |
| (32) | $\mathcal{T}$173-P130-R7-B6-L5-T14-C0.1 | 24 | 24 | 0 | 629 | optimal |
| (33) | $\mathcal{T}$173-P170-R5-B8-L5-T14-C0.1 | 40 | 40 | 0 | 611 | optimal |
| (34) | $\mathcal{T}$173-P150-R6-B9-L4-T14-C0.1 | 32 | 32 | 0 | 600 | optimal |
| (38) | $\mathcal{T}$173-P130-R6-B9-L4-T14-C0.1 | 175 | 175 | 0 | 485 | optimal |
| (39) | $\mathcal{T}$173-P130-R5-B8-L5-T14-C0.2 | 21 | 21 | 0 | 431 | optimal |
| (40) | $\mathcal{T}$173-P170-R7-B6-L5-T14-C0.1 | 119 | 119 | 0 | 417 | optimal |
| (41) | $\mathcal{T}$173-P150-R5-B8-L5-T14-C0.1 | 17 | 17 | 0 | 403 | optimal |
| (42) | $\mathcal{T}$173-P150-R6-B7-L5-T14-C0.1 | 37 | 37 | 0 | 396 | optimal |
| (43) | $\mathcal{T}$173-P130-R6-B7-L5-T14-C0.1 | 5 | 5 | 0 | 326 | optimal |

Table 5 shows the set of test instances for which Gurobi could not reach an optimal solution in the specified computation time. The subset included 5 out of the 45 generated instances, which seemed to have varied features. These instances were the most difficult to

solve in the generated set. We believe that the difficulty of these instances is related to multiple factors, and thus, it cannot be linked to only one generation parameter input. For three instances, Gurobi could not find an estimate for the lower bound, and an undefined $z$ value was reported indicating that Gurobi failed to find any solutions for them. On the other hand, for two instances, Gurobi found solutions very close to the estimated lower bound.

**Table 5.** Gurobi results for unsolved problem instances.

| Case | Instance | $z$ | Bound | Gap | Time (s) | Status |
|---|---|---|---|---|---|---|
| (0) | $\mathcal{T}$173-P130-R5-B8-L5-T14-C0.3 | 37 | 36 | ∼0 | 3619 | time |
| (1) | $\mathcal{T}$173-P170-R6-B9-L4-T14-C0.2 | 176 | 175 | ∼0 | 3601 | time |
| (2) | $\mathcal{T}$173-P130-R9-B6-L4-T14-C0.3 | - | - | - | 3600 | time |
| (3) | $\mathcal{T}$173-P150-R7-B6-L5-T14-C0.2 | - | - | - | 3600 | time |
| (4) | $\mathcal{T}$173-P170-R9-B6-L4-T14-C0.1 | - | - | - | 3600 | time |

Finally, Table 6 provides the set of instances for which Gurobi indicated that there was no feasible solution. We note that most of the infeasible instances define a 30% unavailability level, while the one with the lowest execution time is the only instance with a 20% unavailability level. This points to the fact that the number of conflicts associated with the unavailability of the people involved in the conference program is decisive in defining if a problem has a solution. Nevertheless, as we already mentioned, this single parameter is not enough to understand instance difficulty.

**Table 6.** Gurobi results for infeasible problem instances.

| Case | Instance | Time (s) | Status |
|---|---|---|---|
| (21) | $\mathcal{T}$173-P170-R9-B6-L4-T14-C0.3 | 1032 | infeasible |
| (22) | $\mathcal{T}$173-P150-R9-B6-L4-T14-C0.3 | 852 | infeasible |
| (24) | $\mathcal{T}$173-P170-R7-B6-L5-T14-C0.3 | 799 | infeasible |
| (25) | $\mathcal{T}$173-P150-R6-B9-L4-T14-C0.3 | 769 | infeasible |
| (26) | $\mathcal{T}$173-P150-R5-B8-L5-T14-C0.3 | 734 | infeasible |
| (27) | $\mathcal{T}$173-P150-R7-B6-L5-T14-C0.3 | 686 | infeasible |
| (28) | $\mathcal{T}$173-P170-R6-B7-L5-T14-C0.3 | 681 | infeasible |
| (29) | $\mathcal{T}$173-P170-R5-B8-L5-T14-C0.3 | 639 | infeasible |
| (35) | $\mathcal{T}$173-P150-R6-B7-L5-T14-C0.3 | 599 | infeasible |
| (36) | $\mathcal{T}$173-P130-R7-B6-L5-T14-C0.3 | 595 | infeasible |
| (37) | $\mathcal{T}$173-P130-R6-B7-L5-T14-C0.3 | 486 | infeasible |
| (44) | $\mathcal{T}$173-P130-R9-B6-L4-T14-C0.2 | 106 | infeasible |

### *7.2. Heuristic Results*

In this section, we present the experimental results obtained in the evaluation of our heuristic method when applied to solve the generated instances. Tables 7–9 show the results of the heuristic approach for each instance subset. For each instance, we present the minimum (min(f)), the maximum (max(f)), and the average (Av(f)) evaluation function values obtained by the heuristic approach. In addition, the average execution time (Av(T)) and the status quality of the process returned by Gurobi are shown for reference. In Table 7, we included the optimal objective function found by Gurobi. Results in bold highlight when max(f) = min(f) = Av(f).

From Table 7, we note that for every optimally solved instance, the heuristic method was able to find good-quality solutions. In all instances, the heuristic method reached the optimal solution found by Gurobi. In addition, 82% of the results are shown in bold, which indicates that the heuristic method found the same quality value in all the independent executions. Therefore, this suggests a certain stability of the proposed heuristic algorithm regarding the obtained results. On the other hand, the remaining instances presented small differences between their minimum and maximum $f$ values. Regarding execution time, the heuristic method drastically reduced the time required to obtain optimal solutions

compared to the Gurobi solver. The time reduction ranged between 97% of reduction in the easiest instance to 99% in the hardest instance.

**Table 7.** Heuristic results for optimally solved problem instances. Results in bold highlight when max(f) = min(f) = Av(f).

| Case | Instance | min(f) | max(f) | Av(f) | b | z |
|---|---|---|---|---|---|---|
| (5) | $\mathcal{T}$173-P170-R9-B6-L4-T14-C0.2 | **44** | **44** | **44.0** | 12 | 44 |
| (6) | $\mathcal{T}$173-P130-R9-B6-L4-T14-C0.1 | **65** | **65** | **65.0** | 12 | 65 |
| (7) | $\mathcal{T}$173-P170-R6-B9-L4-T14-C0.3 | 139 | 140 | 139.0 | 6 | 139 |
| (8) | $\mathcal{T}$173-P150-R9-B6-L4-T14-C0.2 | 21 | 22 | 21.0 | 11 | 21 |
| (9) | $\mathcal{T}$173-P130-R7-B6-L5-T14-C0.2 | 30 | 31 | 30.6 | 8 | 30 |
| (10) | $\mathcal{T}$173-P130-R6-B7-L5-T14-C0.2 | **15** | **15** | **15.0** | 7 | 15 |
| (11) | $\mathcal{T}$173-P150-R6-B9-L4-T14-C0.2 | **261** | **261** | **261.0** | 5 | 261 |
| (12) | $\mathcal{T}$173-P130-R5-B8-L5-T14-C0.1 | **134** | **134** | **134.0** | 6 | 134 |
| (13) | $\mathcal{T}$173-P170-R7-B6-L5-T14-C0.2 | **30** | **30** | **30.0** | 8 | 30 |
| (14) | $\mathcal{T}$173-P150-R6-B7-L5-T14-C0.2 | **166** | **166** | **166.0** | 6 | 166 |
| (15) | $\mathcal{T}$173-P170-R6-B7-L5-T14-C0.1 | **21** | **21** | **21.0** | 7 | 21 |
| (16) | $\mathcal{T}$173-P130-R6-B9-L4-T14-C0.3 | 61 | 62 | 61.1 | 6 | 61 |
| (17) | $\mathcal{T}$173-P130-R6-B9-L4-T14-C0.2 | 33 | 34 | 33.1 | 6 | 33 |
| (18) | $\mathcal{T}$173-P150-R9-B6-L4-T14-C0.1 | **22** | **22** | **22.0** | 11 | 22 |
| (19) | $\mathcal{T}$173-P150-R7-B6-L5-T14-C0.1 | **15** | **15** | **15.0** | 8 | 15 |
| (20) | $\mathcal{T}$173-P170-R6-B7-L5-T14-C0.2 | **52** | **52** | **52.0** | 6 | 52 |
| (23) | $\mathcal{T}$173-P170-R5-B8-L5-T14-C0.2 | **41** | **41** | **41.0** | 5 | 41 |
| (30) | $\mathcal{T}$173-P150-R5-B8-L5-T14-C0.2 | **52** | **52** | **52.0** | 4 | 52 |
| (31) | $\mathcal{T}$173-P170-R6-B9-L4-T14-C0.1 | **76** | **76** | **76.0** | 5 | 76 |
| (32) | $\mathcal{T}$173-P130-R7-B6-L5-T14-C0.1 | **24** | **24** | **24.0** | 8 | 24 |
| (33) | $\mathcal{T}$173-P170-R5-B8-L5-T14-C0.1 | **40** | **40** | **40.0** | 5 | 40 |
| (34) | $\mathcal{T}$173-P150-R6-B9-L4-T14-C0.1 | **32** | **32** | **32.0** | 8 | 32 |
| (38) | $\mathcal{T}$173-P130-R6-B9-L4-T14-C0.1 | **175** | **175** | **175.0** | 6 | 175 |
| (39) | $\mathcal{T}$173-P130-R5-B8-L5-T14-C0.2 | **21** | **21** | **21.0** | 5 | 21 |
| (40) | $\mathcal{T}$173-P170-R7-B6-L5-T14-C0.1 | **119** | **119** | **119.0** | 8 | 119 |
| (41) | $\mathcal{T}$173-P150-R5-B8-L5-T14-C0.1 | **17** | **17** | **17.0** | 5 | 17 |
| (42) | $\mathcal{T}$173-P150-R6-B7-L5-T14-C0.1 | **37** | **37** | **37.0** | 7 | 37 |
| (43) | $\mathcal{T}$173-P130-R6-B7-L5-T14-C0.1 | **5** | **5** | **5.0** | 7 | 5 |

**Table 8.** Heuristic results for unsolved problem instances.

| Case | Instance | min(f) | max(f) | Av(f) | Av(T) (s) | Status |
|---|---|---|---|---|---|---|
| (0) | $\mathcal{T}$173-P130-R5-B8-L5-T14-C0.3 | **36** | **36** | **36.0** | 5 | time |
| (1) | $\mathcal{T}$173-P170-R6-B9-L4-T14-C0.2 | **175** | **175** | **175.0** | 6 | time |
| (2) | $\mathcal{T}$173-P130-R9-B6-L4-T14-C0.3 | 10 | 4011 | 810.7 | 13 | time |
| (3) | $\mathcal{T}$173-P150-R7-B6-L5-T14-C0.2 | **42** | **42** | **42.0** | 8 | time |
| (4) | $\mathcal{T}$173-P170-R9-B6-L4-T14-C0.1 | **168** | **168** | **168.0** | 12 | time |

Table 8 presents the results of the heuristic approach for the instance set that remained unsolved by Gurobi. For four of these instances, the heuristic approach always found solutions that had the same $f$ value. Two of these $f$ values were equal to the lower bound estimated by Gurobi (see Table 5), indicating that these solutions were globally optimal. Such results show the advantages of using a heuristic method like the one proposed in this work, as good solutions can be found in a fraction of the time it takes to find a globally optimal solution with a solver like Gurobi. Within this instance subset, instance (2) seemed to be challenging for the heuristic procedure, obtaining results with considerable variability. For this instance, the heuristic found a maximum value of $f = 4011$, which indicated that this solution was infeasible. Moreover, the mean $f$ obtained for this instance was Av(f) = 810.7, indicating that only a few executions yielded infeasible solutions. Consequently, we considered that instance (2) was the most difficult for our heuristic algorithm since it was not always possible to obtain feasible solutions. This instance

considerably differed from the others in the set, as no other instance presented this result variability. When checking the instance's conflict matrix, we observed that the conflicts were fairly distributed among the persons considered in the problem. Consequently, we conclude that the difficulty of the instance was due to the interaction between these conflicts. Regarding the execution time, there was not enough information to observe a relationship between the time required by the heuristic algorithm and the instance features (persons, rooms, etc.). Furthermore, comparing this instance set and the optimally solved instances, we did not see such a relationship between the execution times and the instances.

**Table 9.** Heuristic results for infeasible problem instances.

| Case | Instance | min(f) | max(f) | Av(f) | Av(T) (s) | Status |
|---|---|---|---|---|---|---|
| (21) | $\mathcal{T}$173-P170-R9-B6-L4-T14-C0.3 | 8069 | 12,073 | 8870.9 | 12 | infeasible |
| (22) | $\mathcal{T}$173-P150-R9-B6-L4-T14-C0.3 | 4175 | 8176 | 4975.4 | 12 | infeasible |
| (24) | $\mathcal{T}$173-P170-R7-B6-L5-T14-C0.3 | 5043 | 5049 | 5044.4 | 9 | infeasible |
| (25) | $\mathcal{T}$173-P150-R6-B9-L4-T14-C0.3 | 8152 | 8153 | 8152.1 | 5 | infeasible |
| (26) | $\mathcal{T}$173-P150-R5-B8-L5-T14-C0.3 | 10,064 | 15,068 | 12,264.6 | 5 | infeasible |
| (27) | $\mathcal{T}$173-P150-R7-B6-L5-T14-C0.3 | 10,121 | 10,122 | 10,121.0 | 8 | infeasible |
| (28) | $\mathcal{T}$173-P170-R6-B7-L5-T14-C0.3 | 10,032 | 10,032 | 10,032.0 | 6 | infeasible |
| (29) | $\mathcal{T}$173-P170-R5-B8-L5-T14-C0.3 | 10,063 | 10,065 | 10,063.8 | 5 | infeasible |
| (35) | $\mathcal{T}$173-P150-R6-B7-L5-T14-C0.3 | 10,102 | 10,103 | 10,102.2 | 6 | infeasible |
| (36) | $\mathcal{T}$173-P130-R7-B6-L5-T14-C0.3 | 5012 | 10,013 | 5213.2 | 8 | infeasible |
| (37) | $\mathcal{T}$173-P130-R6-B7-L5-T14-C0.3 | 5039 | 5039 | 5039.0 | 7 | infeasible |
| (44) | $\mathcal{T}$173-P130-R9-B6-L4-T14-C0.2 | 8024 | 8025 | 8024.0 | 12 | infeasible |

Table 9 shows the results of the heuristic method for the infeasible instance set. In these cases, the heuristic method obtained larger $f$ values compared to the other instance sets. These larger values were related to the infeasibility of the problem instances and to the penalty applied to unsatisfied hard constraints. The results showed that in this set, most of the instances did not share the values of min(f) and max(f). This showed that the heuristic method had a hard time converging to one area of the solution space. This can be due to the number of "good-quality" solutions that exist in these instances, given the large numeric difference between the penalty value applied in the evaluation function $f$ and the values of the objective function $z$. Since solutions in these spaces are not feasible, several solutions with the same number of constraint violations might share similar $f$ values despite their different structures. As it is shown in Section 6.2, the penalty multiplier applied in these experiments was $\gamma = 1000$. From this, the number of unsatisfied hard constraints could be estimated as $\sim \frac{f}{1000}$. For instance (24), the heuristic method converged to solutions with $f$ values around 5000, indicating that most of the instances should have around five constraint violations. It is important to mention that the number of unsatisfied constraints was low as it involved only five conflicts that, in a real situation, could be directly arranged with the involved people. This information could be useful for organizers in a real-case scenario to request more availability or flexibility from presenters and organizers, request more rooms, and evaluate changing the venue, among other actions. We note that the min(f) and max(f) values had the same order of magnitude in most of these instances, and thus, it can be considered that the heuristic method converged in terms of orders of magnitude. The instances that had a larger difference in orders of magnitude were instances (21), (22), (26), and (36). Of these instances, the only instance that had a mean Av(f) that was much larger than the min(f) was instance (26), indicating that for this instance, the heuristic method did not always converge to solutions with a similar level of constraint satisfaction. Even though heuristic methods were not able to detect when instances did not have a feasible solution, we considered that these consistent results provided evidence of the lack of feasible solutions.

## 8. Discussion

In this work, we proposed the track-based conference scheduling problem, a conference scheduling problem that considered the notion of tracks in the scheduling process. The proposed problem allows one to perform the scheduling of conferences that define subevents (tracks) which can have their own organization and requirements but should not be scheduled independently (nondivisible problem). We remark that, to our knowledge, this is the first approach that includes such a talk notion. Our proposal included a binary linear mathematical model that considered the minimization of the number of missing seats as its objective. Our model aimed at satisfying the availability of presenters, chairs and organizers, avoiding parallel track and best paper sessions, among other constraints.

We also proposed an instance generation procedure for generating instances for the track-based conference scheduling problem. This procedure allows the generation of instances with different input features, such as the number of talks, tracks, rooms, blocks, time slots, people, and the unavailability level of the people involved in the scheduling. In this work, we generated a set of 45 test instances using a fixed number of talks (173) and tracks (14) and varying the other features. To compare the difficulty of the generated instances, we applied the Gurobi solver to solve them. We identified three subsets of instances based on the results obtained by Gurobi. The largest subset contained instances that the solver was able to solve optimally. The second subset contained instances that Gurobi determined as infeasible, and the smallest was the instance subset that Gurobi could not solve in the given time. These results showed that the generated instance set contained instances with different characteristics, not only in terms of their input features but also in terms of their difficulty levels. The difference in difficulty between the generated instances was interesting, as these differences were not necessarily correlated with the input features. Such difficult differences were probably due to the conflicts arising from the availability of the people considered in the scheduling. In this context, a future line of work would be improving the generation procedure, maybe including other features that can offer more refined control over the difficulty level of the generated instances. Regarding the instance set generated in this work, we consider that it is a good benchmark tool for evaluating solution approaches as it contains instances with different difficulty levels.

Aiming to provide a practical method for solving the most complex instances in the generated instance set, we presented a simulated-annealing-based algorithm for the proposed problem. We evaluated our procedure on the generated set of instances to assess the algorithm's ability to tackle different instances. The results showed that the algorithm could solve, in a fraction of time, all the instances for which Gurobi found the optimal solution. In the case of the instances where Gurobi did not find a solution in the provided time, the heuristic method seemed to have converged to near-optimal solutions. Not surprisingly, the results obtained for the infeasible instances were the most varied of the test. Despite this, we could observe that the heuristic approach converged to "low" infeasibility levels for these instances. This showed that the heuristic approach could provide complete instantiations that, even though they were not feasible, offered a conflict-reduced alternative that could be used as a base to improve upon, for example, considering additional people (alternative presenters, new organizers) in the conference schedule. We note that such infeasible instance scenarios are possible in real settings, and thus, alternatives should be at hand to create the final conference program. Overall, experimental results indicated that the proposed heuristic was an adequate approach for the track-based conference scheduling problem. The proposed approach allowed us to obtain good solutions quicker than the Gurobi solver, while also offering better solving perspectives for more complex or larger instances.

The work presented in this article aimed at including in the problem formulation the most distinctive features of the conference program of our reference real-world case: the Genetic and Evolutionary Computation Conference. The proposed formulation focused on obtaining feasible solutions (conference programs), and thus, it did not allow completely infeasible instantiations to be considered valid program alternatives. This is a clear limi-

tation when considering scenarios in which feasible schedules are not possible and that, consequently, require someone to intervene to generate favorable problem conditions or make compromises regarding the program. In future work, we would like to incorporate the minimization of infeasibility as an objective in our mathematical formulation. In this way, an exact solver like Gurobi would be able to provide complete instantiations that minimize infeasibility when dealing with infeasible instances. In the same context, the problem formulation can be extended to include other program features that can be considered for the scheduling. In particular, our problem definition currently does not include the use of joint sessions and other preferences related to the scheduling of the sessions and the talks within them. We aim to include such features in future work. Such work would contribute to providing a more flexible problem definition that can handle additional conference program requirements.

A key aspect of this work is the instance generation procedure. This method, though simple, provides reasonable test instances with different difficulty levels enabling the evaluation and comparison of approaches. In this context, we plan to improve the instance-generation procedure, including new features that provide better control over the generated instances. Furthermore, generating a more diverse set of instances in terms of input features and complexity is interesting as it would help to establish an instance benchmark set for this problem. Finally, we aim to improve the search performance of our algorithm in future work. In particular, we would like to focus on solving large and more diverse instances in reasonable computation times.

## 9. Conclusions

The task of scheduling a conference event can be defined as an event-based timetabling problem that aims at finding an adequate conference program that assigns a set of talks to rooms and time slots. In this work, we focused on such conference scheduling approaches, contributing to it by (1) providing a comprehensive literature review that placed conference scheduling problems within the timetabling problem family, (2) proposing the track-based conference scheduling problem that included the notion tracks as its main feature, (3) providing a binary integer linear mathematical formulation that modeled the problem, (4) proposing a simple feature-based instance generation procedure, (5) proposing a simulated annealing-based approach to solve the problem, and (6) providing an experimental evaluation of the proposed method and a comparison with the results obtained by the Gurobi Solver. We consider that the results obtained are promising both regarding our problem formulation and the results obtained by the proposed heuristic method. Future research directions will focus on improving the problem formulation extending it to include new features and objectives, establishing a varied instance benchmark, and improving the proposed heuristic approach.

**Author Contributions:** Conceptualization, L.P.-C., F.R., N.R.-M. and E.M.; investigation, F.R. and E.M.; methodology, F.R., L.P.-C., N.R.-M. and E.M.; software, F.R. and E.M.; supervision, E.M.; validation, F.R. and E.M.; writing—original draft, L.P.-C., N.R.-M. and E.M.; writing—review and editing, L.P.-C., N.R.-M. and E.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** Leslie Pérez-Cáceres was funded by FONDECYT project number 11190154. Nicolás Rojas-Morales was funded by Universidad Técnica Federico Santa María DGIIP project number PI_LII_2022_03.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Problem instances are available at the specified website.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Doshi, V.; Tuteja, S.; Bharadwaj, K.; Tantillo, D.; Marrinan, T.; Patton, J.; Marai, G.E. StickySchedule: An Interactive Multi-User Application for Conference Scheduling on Large-Scale Shared Displays. In Proceedings of the 6th ACM International Symposium on Pervasive Displays, PerDis '17, Lugano, Switzerland, 7–9 June 2017; Association for Computing Machinery: New York, NY, USA, 2017. [CrossRef]
2. Yuraszeck, F.; Mejía, G.; Pereira, J.; Vilà, M. A Novel Constraint Programming Decomposition Approach for the Total Flow Time Fixed Group Shop Scheduling Problem. *Mathematics* **2022**, *10*, 329. [CrossRef]
3. Zhang, H.; Buchmeister, B.; Li, X.; Ojstersek, R. Advanced Metaheuristic Method for Decision-Making in a Dynamic Job Shop Scheduling Environment. *Mathematics* **2021**, *9*, 909. [CrossRef]
4. Li, Y.; Carabelli, S.; Fadda, E.; Manerba, D.; Tadei, R.; Terzo, O. Machine learning and optimization for production rescheduling in Industry 4.0. *Int. J. Adv. Manuf. Technol.* **2020**, *110*, 2445–2463. [CrossRef]
5. Chávez-Bosquez, O.; Hernández-Torruco, J.; Hernández-Ocaña, B.; Canul-Reich, J. Modeling and Solving a Latin American University Course Timetabling Problem Instance. *Mathematics* **2020**, *8*, 1833. [CrossRef]
6. Zhu, K.; Li, L.D.; Li, M. School Timetabling Optimisation Using Artificial Bee Colony Algorithm Based on a Virtual Searching Space Method. *Mathematics* **2022**, *10*, 73. [CrossRef]
7. Schönberger, J.; Mattfeld, D.; Kopfer, H. Memetic Algorithm timetabling for non-commercial sport leagues. *Eur. J. Oper. Res.* **2004**, *153*, 102–116. [CrossRef]
8. Lee, H.Y.; Lin, Y.C. A decision support model for scheduling exhibition projects in art museums. *Expert Syst. Appl.* **2010**, *37*, 919–925. [CrossRef]
9. Zhang, W.; Xia, D.; Liu, T.; Fu, Y.; Ma, J. Optimization of single-line bus timetables considering time-dependent travel times: A case study of Beijing, China. *Comput. Ind. Eng.* **2021**, *158*, 107444. [CrossRef]
10. Vangerven, B.; Ficker, A.M.; Goossens, D.R.; Passchyn, W.; Spieksma, F.C.; Woeginger, G.J. Conference scheduling—A personalized approach. *Omega* **2018**, *81*, 38–47. [CrossRef]
11. Sampson, S.E. Practical Implications of Preference-Based Conference Scheduling. *Prod. Oper. Manag.* **2004**, *13*, 205–215. [CrossRef]
12. Stidsen, T.; Pisinger, D.; Vigo, D. Scheduling EURO-k conferences. *Eur. J. Oper. Res.* **2018**, *270*, 1138–1147. [CrossRef]
13. Castaño, F.; Velasco, N.; Carvajal, J. Content-Based Conference Scheduling Optimization. *IEEE Lat. Am. Trans.* **2019**, *17*, 597–606. [CrossRef]
14. Vallejo-Huanga, D.; Morillo, P.; Ferri, C. Semi-Supervised Clustering Algorithms for Grouping Scientific Articles. *Procedia Comput. Sci.* **2017**, *108*, 325–334. [CrossRef]
15. Bulhões, T.; Correia, R.; Subramanian, A. Conference scheduling: A clustering-based approach. *Eur. J. Oper. Res.* **2021**, *297*, 15–26. [CrossRef]
16. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
17. Lin, B.; Zhao, Y.; Lin, R.; Liu, C. Integrating traffic routing optimization and train formation plan using simulated annealing algorithm. *Appl. Math. Model.* **2021**, *93*, 811–830. [CrossRef]
18. Yu, V.F.; Winarno, W.; Maulidin, A.; Redi, P.; Lin, S.W.; Yang, C.L. Simulated Annealing with Restart Strategy for the Path Cover Problem with Time Windows. *Mathematics* **2021**, *9*, 1625. [CrossRef]
19. Ilhan, I. An improved simulated annealing algorithm with crossover operator for capacitated vehicle routing problem. *Swarm Evol. Comput.* **2021**, *64*, 100911. [CrossRef]
20. Sajid, M.; Mittal, H.; Pare, S.; Prasad, M. Routing and scheduling optimization for UAV assisted delivery system: A hybrid approach. *Appl. Soft Comput.* **2022**, *126*, 109225. [CrossRef]
21. Kantor, D.; Von Zuben, F.; Olivetti de França, F. Simulated annealing for symbolic regression. In Proceedings of the GECCO '21: Genetic and Evolutionary Computation Conference, Lille, France, 10–14 July 2021; Chicano, F., Krawiec, K., Eds.; ACM: New York, NY, USA; 2021; pp. 592–599. [CrossRef]
22. Lin, S.; Lee, Z.; Chen, S.; Tseng, T. Parameter determination of support vector machine and feature selection using simulated annealing approach. *Appl. Soft Comput.* **2008**, *8*, 1505–1512. [CrossRef]
23. Elgamal, Z.; Yasin, N.; Tubishat, M.; Alswaitti, M.; Mirjalili, S. An Improved Harris Hawks Optimization Algorithm With Simulated Annealing for Feature Selection in the Medical Field. *IEEE Access* **2020**, *8*, 186638–186652. [CrossRef]
24. Onyezewe, A.; Kana, A.F.; Abdullahi, F.B.; Abdulsalami, A.O. An enhanced adaptive K-nearest neighbor classifier using simulated annealing. *Int. J. Intell. Syst. App.* **2021**, *13*, 34–44. [CrossRef]
25. Biswas, T.; Abbasi, A.; Chakrabortty, R. An MCDM integrated adaptive simulated annealing approach for influence maximization in social networks. *Inf. Sci.* **2021**, *556*, 27–48. [CrossRef]
26. Zhou, A.; Zhu, L.; Hu, B.; Deng, S.; Song, Y.; Qiu, H.; Pan, S. Traveling-Salesman-Problem Algorithm Based on Simulated Annealing and Gene-Expression Programming. *Information* **2019**, *10*, 7. [CrossRef]
27. Moradi, N.; Kayvanfar, V.; Rafiee, M. An efficient population-based simulated annealing algorithm for 0-1 knapsack problem. *Eng. Comput.* **2022**, *38*, 2771–2790. [CrossRef]
28. Jouhari, H.; Lei, D.; AA Al-qaness, M.; Abd Elaziz, M.; Ewees, A.A.; Farouk, O. Sine-cosine algorithm to enhance simulated annealing for unrelated parallel machine scheduling with setup times. *Mathematics* **2019**, *7*, 1120. [CrossRef]
29. Nayeri, S.; Tavakkoli-Moghaddam, R.; Sazvar, Z.; Heydari, J. A heuristic-based simulated annealing algorithm for the scheduling of relief teams in natural disasters. *Soft Comput.* **2022**, *26*, 1825–1843. [CrossRef]

30. Frausto-Solis, J.; Hernández-Ramírez, L.; Castilla-Valdez, G.; González-Barbosa, J.J.; Sánchez-Hernández, J.P. Chaotic Multi-Objective Simulated Annealing and Threshold Accepting for Job Shop Scheduling Problem. *Math. Comput. Appl.* **2021**, *26*, 8. [CrossRef]
31. Attiya, I.; Abualigah, L.; Alshathri, S.; Elsadek, D.; Abd Elaziz, M. Dynamic Jellyfish Search Algorithm Based on Simulated Annealing and Disruption Operators for Global Optimization with Applications to Cloud Task Scheduling. *Mathematics* **2022**, *10*, 1894. [CrossRef]
32. Montero, E. Track-Based Conference Scheduling Problem Instances. 2022. Available online: https://github.com/elimail/Track-BasedProblemInstances (accessed on 7 September 2022).
33. Hutter, F.; Hoos, H.H.; Stützle, T. Automatic Algorithm Configuration based on Local Search. In Proceedings of the Twenty-Second Conference on Artificial Intelligence of the Association for the Advancement of Artificial Intelligence 2007 (AAAI), Vancouver, BC, Canada, 22–26 July 2007; pp. 1152–1157.