Omega 81 (2018) 38–47

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

Conference scheduling — A personalized approach[☆,☆☆]

Bart Vangerven[a,b,*], Annette M.C. Ficker[a], Dries R. Goossens[b], Ward Passchyn[a], Frits C.R. Spieksma[a], Gerhard J. Woeginger[c]

[a] *Faculty of Economics and Business, KU Leuven, Naamsestraat 69, Leuven, 3000, Belgium*
[b] *Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, Ghent, 9000, Belgium*
[c] *Faculty of Mathematics, Computer Science and Natural Sciences, RWTH Aachen, Ahornstr. 55, Aachen, 52056, Germany*

A R T I C L E I N F O

Article history:
Received 1 March 2017
Accepted 20 September 2017
Available online 21 September 2017

Keywords:
Conference scheduling
Computational complexity
Case study
Integer programming

A B S T R A C T

Scientific conferences have become an essential part of academic research and require significant investments (e.g. time and money) from their participants. It falls upon the organizers to develop a schedule that allows the participants to attend the talks of their interest. We present a combined approach of assigning talks to rooms and time slots, grouping talks into sessions, and deciding on an optimal itinerary for each participant. Our goal is to maximize attendance, taking into account the common practice of *session hopping*. On a secondary level, we accommodate presenters' availabilities. We use a hierarchical optimization approach, sequentially solving integer programming models, which has been applied to construct the schedule of the MathSport (2013), MAPSP (2015 and 2017) and ORBEL (2017) conferences.

1. Introduction

Scientific conferences have become an essential aspect of (academic) life. They allow researchers (i) to present their work and receive feedback, (ii) to learn from attending talks, poster sessions, or discussion panels, and (iii) to meet with colleagues, thereby inducing new collaborations. However, attending a conferences requires a considerable effort in terms of time (e.g. preparing talks, traveling time) and money (e.g. registration fees, traveling expenses, hotels) from their participants. Conferences also have a non-negligible environmental impact [2]. In fact, there is some debate about the value of scientific conferences, see e.g. [3], and how to lessen the carbon footprint of a conference [4]. Obtaining exact figures with respect to the amount of money involved in organizing scientific conferences seems difficult; it is written in [5] that "an estimate of more than 100.000 medical meetings per year may not be unrealistic ... the cumulative cost of these events worldwide is not possible to fathom". Note that this figure applies to medical conferences alone.

Given these considerations and investments, it is the responsibility of the organizers to maximize the value of a conference as much as possible. Here we focus on the construction of a conference schedule that allows participants to maximally benefit from participating. Or, making this even more concrete, the schedule should enable participants to attend the talks of their interest. This clearly benefits speakers as well, potentially increasing both the size and the level of interest of their audience. Typically, a conference schedule groups talks into *sessions* (a set of talks taking place consecutively in the same room); consecutive sessions are separated by a break. Furthermore, the vast majority of conferences feature several sessions taking place at the same moment in time, i.e. sessions are scheduled *in parallel*. Consequently, a participant may be confronted with times where several attractive talks compete for his/her attendance (i.e. a *scheduling conflict*), while at other times (s)he finds nothing of interest in the schedule. A small example is given in Fig. 1, which depicts two alternative conference schedules. In schedule 1, the participant needs to choose between preferred talks A/C, and J/L. In other words: that participant can only see half the talks he or she actually wants to see. This is not the case in schedule 2.

One popular approach to schedule conferences is *track segmentation* [6]. The organizer groups talks that cover a similar topic or method into tracks or clusters, which are then assigned to a room and scheduled in parallel. Note that a track can consist of multiple sessions. If a participant were only interested in talks from a single track, then (s)he can stay in that track's room for the duration of the conference without experiencing any scheduling conflict. However, apart from difficulties in forming meaningful clusters, track

☆ An extended abstract corresponding to an earlier version of this paper appeared in [1].

☆☆ This manuscript was processed by Associate Editor Singh.

* Corresponding author at: Faculty of Economics and Business, KU Leuven, Naamsestraat 69, Leuven, 3000, Belgium.

E-mail address: bart.vangerven@kuleuven.be (B. Vangerven).

https://doi.org/10.1016/j.omega.2017.09.007
0305-0483/© 2017 Elsevier Ltd. All rights reserved.

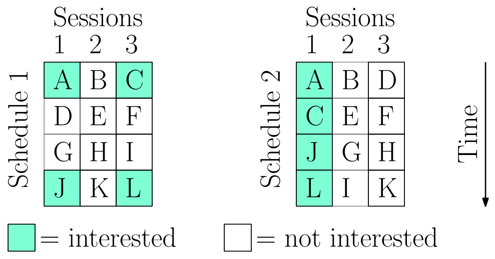

Fig. 1. The impact of the schedule on attendance.

segmentation is not very effective if the participant's preferences are diverse, and not restricted to one particular topic.

In this work, a participant is expected to provide a list of preferred talks, which he or she would like to attend. Our goal is to develop a conference schedule that maximizes the participants' satisfaction. Primarily, this means we want to avoid scheduling conflicts, thereby maximizing total attendance. Next, as a secondary goal, we want to minimize *session hopping*. Indeed, confronted with multiple talks of interest scheduled in different sessions, a participant is forced to move between several sessions in order to attend as many of his or her preferred talks as possible. We call this phenomenon session hopping, and its presence is a clear indication of the existence of strong preferences of participants. Session hopping can be perceived as disturbing by presenters and their audiences. Moreover, the session hopper still tends to miss parts of the preferred talks, due to the time it takes to switch rooms and presenters not always starting at exactly the scheduled time. Finally, motivated by practical considerations, we also take presenter availabilities into account.

Our main contribution is the description of a method for the planning of a (scientific) conference. Based on given preferences of the participants, our method schedules individual talks in order to maximize total attendance; this is in contrast to many other approaches that work on the level of sessions or streams. As a secondary, original criterion, we take session hopping into account, aiming for schedules that allow participants to stay within the same room during a session. We are the first to incorporate session hopping in our scheduling approach, as session hopping is either assumed to be forbidden or non-existing in the literature, as opposed to regular participant practice. Our method has been used to schedule four scientific conferences, namely MathSport 2013, MAPSP 2015, MAPSP2017 and ORBEL2017 — we give a detailed account of our experience with the method.

We provide an overview of related work in Section 2. A detailed problem definition, is given in Section 3, followed by computational complexity results in Section 4. Next, we describe our solution method in Section 5. Finally, we present case studies on the MathSport 2013, MAPSP 2015, MAPSP2017 and ORBEL 2017 conference in Section 6. We finish with conclusions in Section 7.

2. Literature review

Thompson [7] discerns two approaches to conference scheduling: a *presenter-based perspective* (PBP) and an *attender-based perspective* (ABP). With a PBP, the main goal is to meet time preferences and availability restrictions of the presenters. On the other hand, from an ABP, participants' preferences are solicited, in order to maximize their satisfaction. In the rest of this section, we will first discuss contributions that focus on the PBP, continue with papers that follow an ABP, and conclude with a few papers that solve subproblems of conference scheduling. Although we focus here on scheduling scientific conferences, there is also literature on scheduling meetings that are based on preferences of the participants; we mention Yingping et al. [8], Ernst et al. [9] and Ernst et al. [10].

2.1. Presenter-based perspective

Potthoff and Munger [11] discuss a problem where sessions need to be assigned to time periods (rooms are ignored). The authors assume that the clustering of talks into sessions has already been done, in a way that each session belongs to a subject area. The goal is to find a schedule that spreads the sessions for each subject area among the time slots as evenly as possible, ensuring that no presenter has other duties (e.g. being discussant) in simultaneous sessions. An IP formulation is presented and applied to a problem instance extracted from a past meeting of the Public Choice Society, including 96 sessions and over 300 participants. This problem is revisited by Potthoff and Brams [12], who extend the IP formulation to take into account presenter availabilities. Furthermore, their method is applied to schedule two Public Choice Society meetings, with 76 and 45 sessions.

Edis and Sancar Edis [13] consider a very similar problem, but at the level of talks instead of sessions. Each talk has a given topic, and should be assigned to a session and a time period, such that all talks in each session have the same topic, and the occurrence of simultaneous sessions with the same topic is minimized. Furthermore, the number of talks in different sessions with same topic should be balanced, and some talks cannot be scheduled simultaneously. The authors also discuss an extended setting where presenters have preferred and non-preferred days. An IP formulation is presented, which is used to solve a hypothetical instance, including 170 talks on one of 10 topics, to be scheduled into sessions of at most 5 talks, over 12 time periods.

Nicholls [14], like Potthoff and Munger [11], also assumes that papers have been assigned to sessions beforehand by the organizers, but includes room assignment. The problem at hand is to assign each session to a room and a time period, such that no presenter is scheduled at two sessions simultaneously. The goal is to maximize the number of presenter preferences (e.g. preferred day or time slot) met. Participant preferences are not elicited, but can be included implicitly by the program chair, for instance by allocating appropriate rooms to sessions based on expectations regarding attendance. The author presents an algorithm, which is essentially a step-wise constructive heuristic, complemented with a set of rules to accommodate preferences and resolve conflicts. Nicholls [14] applied his method to schedule a Western Decision Sciences Institute annual conference. This conference had over 300 participants, involving over 80 sessions and spanning 4 days.

2.2. Attender-based perspective

An early attempt to optimize participant satisfaction is by Eglese and Rand [15], who collect a list of 4 preferred sessions (and one reserve session) from each participant. In their conference scheduling problem, sessions need to be assigned to time periods and rooms such that the sum of the weighted violations of session preferences is minimized. Furthermore, sessions can be offered multiple times, a decision which is also part of the problem. Although the number of rooms is limited and some rooms are not equipped with the right facilities for some sessions, room capacity is assumed to be always sufficient. The paper reports the scheduling of the national Tear Fund conference, including 15 distinct sessions, over 4 time periods and 7 rooms. As an IP formulation for a problem of this size was deemed intractable at the time, the problem was solved using simulated annealing.

Sampson and Weiss [16] extend the Eglese and Rand [15] setting as they consider rooms with finite seating capacities. They present a heuristic procedure that simultaneously assigns session offerings to time periods and rooms, and decides for each participant which sessions to attend (assuming that session hopping is forbidden). The procedure is tested on a number of randomly generated problem instances. Sampson [6] describes how an annual meeting of the Decision Sciences Institute with 213 sessions to be scheduled over 10 time slots was handled using this method. Nearly half of the 1086 registered participants submitted ranked preferences for talks, which was used to rank the sessions. A post-conference survey revealed that about one quarter of the participants found the resulting schedule “much better” than in previous meetings. The method is also a part of a simulation to numerically address other issues that might be faced by a conference organizer. For instance, Sampson and Weiss [17] discuss tradeoffs between the length of the conference, the number of offerings per session and participant satisfaction. They also investigate how seating capacity, room availability, and the utilization of time slots impact participant satisfaction.

Gulati and Sengupta [18] enhance the problem description by Sampson and Weiss [16] by augmenting the objective function with a prediction of the popularity of a talk, based on reviewers' assessments of the submissions and linked with time slot preferences of participants (e.g. late and last-day time slots are often poorly attended). The overall goal is to maximize the total session attendance. Gulati and Sengupta [18] develop a solution method called TRACS (TRActable Conference Scheduling), which is essentially a greedy algorithm; no empirical results or computational analysis are reported.

The conference scheduling problem discussed by Thompson [7] is also similar to that of Sampson and Weiss [16]. However, in [7], meeting rooms may have different capacities, and may not always be available. He presents a method that employs a constructive heuristic followed by a simulated annealing procedure. The author performs a number of computational experiments, based on randomly generated data as well as data from a real, yet unspecified, conference. The latter includes 47 distinct sessions (some of which were to be offered 2 or 3 times), 8 time slots, and 8 rooms with different capacities. Presenters present in 1–5 sessions and each of the 175 participants have provided between 0 and 8 preferred sessions (neither ranked nor weighted). The author finds that his heuristic outperforms randomly as well as manually generated schedules.

Le Page [19] assumes that each participant provides a list with a given number of sessions he or she wishes to attend. This allows to create a *conflict matrix*, where each matrix element $c_{i,j}$ represents the number of participants that wish to attend both sessions i and j. The problem is to assign the sessions to time slots and rooms (with different capacities), such that the sum of conflicts between simultaneous sessions is minimized. Furthermore, sessions with the same topic must be assigned to the same room, and some sessions need to be planned consecutively on the same day. The author develops a semi-automated heuristic in four steps, which is used to schedule a meeting of the American Crystallographic Association. This meeting includes 35 sessions, to be assigned to 5 rooms and 7 time periods. Months before the conference, preferences were solicited from the 1100 participants; about 10% of them provided a list of 7 preferred sessions. Most popularity predictions based on this input turned out to be accurate during the actual conference.

Ibrahim et al. [20] focus on a conference scheduling problem where talks need to be assigned to time slots (spread over a number of days) in 3 parallel tracks. Each talk belongs to a field, and the schedule should be such that talks of the same field do not occur simultaneously. Furthermore, it should be avoided to schedule talks belonging to the same pair of fields in parallel more than once on the same day. The authors discuss construction methods, based on results from combinatorial design theory, for 3 cases. One case is based on data from the National Conference in Decision Science and includes 73 sessions, belonging to 8 fields, to be scheduled over 26 time slots and 2 days. Note that this setting does not involve grouping talks into sessions. Moreover, the sequence of the talks within a track on one day is of no importance, and all talks from the same field can be swapped without changing the solution quality.

In the so-called *preference conference optimization problem* (PCOP) as defined by Quesnelle and Steffy [21], talks need to be assigned to a time slot and a room, such that scheduling conflicts are minimized. Furthermore, room and presenter availabilities need to be taken into account, including the fact that some presenters are involved in more than one talk and must be able to attend each one of them. Some talks are required to be offered multiple times. Quesnelle and Steffy [21] show that PCOP is NP-hard and discuss an IP formulation, together with a number of performance considerations such as symmetry reduction. They apply their method on a problem instance, based on a PenguiCon conference with 253 talks. As no individual participant preferences were available, the authors have randomly generated this data from historical attendance data, for various choices of the standard deviation of the number of preferred talks per participant. Notice that the issue of grouping talks into sessions is not included in this problem, in fact, as in Ibrahim et al. [20], each talk could be seen as a session.

2.3. Related problems

The problem of grouping talks into coherent sessions, given one or more keywords for each talk, is discussed by Tanaka et al. [22] and Tanaka and Mori [23]. The objective function is a nonlinear utility function of common keywords, with the underlying idea that papers in the same session have as many common keywords as possible, provided that the number of talks is balanced over the sessions. This problem is tackled using Kohonen's self-organizing maps [22] and a hybrid grouping genetic algorithm [23]. Both methods are tested on data from a conference of the Institute of Systems, Control and Information Engineers in Japan with 313 papers and 86 keywords.

Zulkipli et al. [24] ignore session coherence as they attempt to group talks into equally popular sessions. The underlying idea is that in a setting with rooms of similar size and assuming that session hopping is forbidden, this will maximize participants' satisfaction in terms of seating capacity. Given a weight for each talk, based on preferences from the participants, the goal is to assign talks to sessions, such that the sum of the talk weights is balanced over the sessions. The authors present a goal programming method, which is applied to one case, involving 60 talks to be grouped into 15 sessions.

Martin [25] elaborates on the sessions selection problem for the participant, given the conference schedule. He develops a decision support system for participants to determine their itinerary. Using a web-based approach, keyword preferences are elicited and matched with keywords supplied by talks, in order to produce an aggregate rating for each talk. This approach, which does not involve an optimization algorithm, has been used for a conference of the UK Academy of Information Systems. About one third of the 118 participants made use of the decision support system, however, the author was not able to predict session attendance based on the keyword ratings.

3. Problem description

There are a number of crucial ingredients in our problem. First, there is a set of talks that needs to be scheduled; the set of talks is denoted by X. Second there is a set of timeslots, denoted by T; a timeslot refers to a period in time during which a number of talks are held in parallel — we assume that the number of talks that are held in parallel (i.e. the number of parallel sessions) is given and we denote that number by n. Further, we assume without loss of generality that the number of talks $|X|$ is a multiple of n (if necessary, this can be achieved by adding dummy talks) – notice that this means that $|T| = \frac{|X|}{n}$. A final ingredient of our problem are the participants, denoted by the set P, and their *profiles*.

Definition 1. A profile of a participant $p \in P$ is represented by a binary vector $q(p)$ where $q(p)_i$ equals 1 if and only if participant p wishes to attend talk $i \in X$. A profile consisting of only 0 entries is called a trivial profile.

In other words, a profile represents the preferences of a participant. All these ingredients allow us to formally state our problem. We assume that talks that are held in parallel cannot both be attended by the same participant, and that talks that are assigned to distinct timeslots can be attended by the same participant. The *attendance* of a talk denotes the number of participants attending a talk, and total attendance refers to the summed attendances over the talks. We assume that a participant, when a preferred talk is scheduled, will attend this talk, and in case multiple preferred talks are presented at the same time, (s)he will arbitrarily choose one of these talks to attend. Finally, we assume all participants attend the entirety of the conference.

Definition 2. Given the participants' profiles $q(p)$, and given the number of parallel sessions n, the *Conference Scheduling Problem* with n parallel sessions (CSP-n) seeks to assign every talk to a timeslot such that each timeslot receives exactly n talks, while maximizing total attendance.

The profiles allow us to compute the parameter $v_i := \Sigma_{p \in P} q(p)_i$: the number of participants who wish to attend talk i. Clearly, total attendance is upper bounded by $\Sigma_{i \in X} v_i$; notice that this number can be realized in case there are no parallel sessions, i.e. when $n = 1$.

From a computational complexity standpoint, CSP-n is an NP-hard optimization problem, already in case of three parallel sessions. We address this issue in more detail in Section 4.

It is clear that there will be several optimal solutions to CSP-n, as the grouping of the parallel talks into sessions and their order within a session do not impact attendance. Hence, as a secondary goal, we aim to minimize the total number of session hops. To describe this problem more precisely, let the length of a session be the number of consecutive talks in the session. Typically, a session consists of k consecutive talks, with $k \in \{2, 3, 4\}$. We use K to denote the set of session lengths present in the conference. We assume that sessions running in parallel must have the same length, and we call a set of n parallel sessions of length $k \in K$, a k-block. The *format* of a conference specifies, for each relevant session length $k \in K$, the number of k-blocks in the conference; this number is referred to as r_k. Given the format of the conference, we settle the composition of the sessions, taking into account the talks that are to be scheduled in parallel in order to maximize attendance.

However, the resulting schedule still leaves room to decide during which timeslots these parallel sessions are scheduled. This gives freedom to accommodate potential restrictions on the availabilities of speakers. Thus, in a final phase, we assign the k-blocks to timeslots, minimizing the number of violated presenter availabilities.

Concluding, the resulting conference scheduling problem has three objectives: maximizing attendance (5.1), minimizing session hopping (5.2), and satisfying presenter availabilities (5.3), which are considered hierarchically in this order. Notice that we have not taken room capacities into account; in Section 6 we describe how, if necessary, this issue can still be dealt with.

4. Computational complexity of CSP-*n*

In the following two theorems, we respectively show that the CSP-2 is polynomially solvable, and that the CSP-n is NP-hard for $n \geq 3$.

Theorem 1. *CSP-2 is solvable in polynomial time.*

Proof. We show that CSP-2 is a special case of the minimum weight perfect matching problem, which is polynomially solvable [26]. Given a graph $G = (V, E)$, a matching in G is a set of pairwise non-adjacent edges. A perfect matching is a matching which matches all nodes of G, i.e., every node is incident to exactly one edge of the matching.

The reduction goes as follows. Given an instance of CSP-2, we construct a complete, edge-weighted, graph $G = (V, E)$ such that each talk in CSP-2 corresponds to exactly one node in G; thus $V := X$. For every distinct pair of talks i and $j \in X$, we calculate a coefficient $c_{i,j}$ capturing how much attendance is missed if both talks i and j are planned simultaneously, i.e., we set $c_{i,j} := |\{p \in P : q(p)_i = q(p)_j = 1\}|$. Since (i) any solution to CSP-2 can be regarded as $\frac{|X|}{2}$ pairs of talks, and hence is a perfect matching, and (ii) the coefficient $c_{i,j}$ equals the missed attendance when talks i and j are planned simultaneously, the result follows. □

Observe that while the proof of Theorem 1 shows that CSP-2 is a special case of minimum weight perfect matching, the converse is true as well. Indeed, when we interpret, in a given instance of minimum weight perfect matching specified by a graph $H = (W, F)$, each node in W as a talk, and the cost associated to an edge $e = (i, j) \in F$ as the number of participants wishing to see both talks i and j, it is not difficult to see that a matching with a certain cost corresponds to a schedule with that cost as the missed attendance. All this essentially implies that CSP-2 cannot be solved faster than minimum weight perfect matching.

Theorem 2. *CSP-n is NP-hard for each fixed $n \geq 3$.*

Proof. We will prove that the decision variant of CSP-3, which asks the question "Given a number of talks and a set of participants with corresponding profiles, does a schedule consisting of 3 parallel sessions exist such that no attendance is missed?", is NP-complete. To prove this, we will use the *Triangle Partition Problem* (TPP), which is known to be NP-complete even for graphs with a maximal degree of at most four [27]. An instance of the TPP is a graph $G = (V, E)$ with $|V| = 3\ell$, where $\ell \in \mathbb{N}_0^+$. A triangle is a collection of three nodes in G such that each pair is connected by an edge. The question that TPP asks is then: "Can the nodes of G be partitioned into ℓ disjoint sets $V_1, V_2, \ldots, V_\ell$ each containing exactly 3 nodes, such that each of these V_i is the node set of a triangle in G?".

We transform an arbitrary instance of TPP into an instance of the CSP-3. Each node in G will correspond to a talk in CSP-3, i.e. $X := V$. We set $P := \{(i, j): i, j \in V, i \neq j, (i, j) \notin E\}$. Next, for all $p \in P$, say $p = (i, j)$, we have

$$q(p)_x = \begin{cases} 1 & \text{if } x = i \text{ or } x = j, \\ 0 & \text{otherwise.} \end{cases}$$

Informally, for every non-existing edge in the instance of TPP, we have a participant in CSP-3 who wishes to see the corresponding two talks. This completely specifies an instance of CSP-3.

Suppose the instance of TPP is a yes-instance. Then, by definition, ℓ disjoint triangles must exist in G. The three vertices of each triangle correspond to three talks that we schedule simultaneously, i.e., in parallel. Next, the parallel talks are assigned to timeslots in any order. Note that no participant misses a talk, since no participant exists that wishes to see two (or more) talks from the triple of talks scheduled in parallel. Thus, the resulting instance of CSP-3 is a yes-instance. Conversely, suppose that we have a yes-instance of the decision variant of CSP-3. Hence, we know which talks are scheduled in parallel. From this, we easily find a partition into triangles: simply select the nodes corresponding to the parallel talks as the nodes of a triangle. These nodes must correspond to triangles in G, because otherwise there would have been a missed attendance. From this it follows that CSP-n is NP-hard for $n \geq 3$. □

This complexity result is tight, in the sense that CSP-n is NP-hard even if each participant has only 2 preferred talks in his/her profile (and the problem becomes trivial if each participant has at most one preferred talk). Furthermore, our result strengthens the result that the preference conference optimization problem (PCOP) is NP-hard by Quesnelle and Steffy [21]. Indeed, their result is based on the presence of room and presenter availabilities, while in CSP-n every talk can be allocated to any timeslot.

5. Method

In this section we will explain a hierarchical three-phased approach to scheduling conferences. In the first phase (see Section 5.1), we maximize total attendance, based on the participants' profiles, i.e., we solve CSP-n. In the second phase, we seek to minimize the number of session hops, given that total attendance is maximal (see Section 5.2). Finally, in a third phase, we take into account presenter availabilities. We do this by minimizing the number of violated availability constraints, while fixing the total attendance and number of session hops at the levels obtained in the previous two phases (see Section 5.3).

5.1. Phase 1: maximizing total attendance

It should be obvious that maximizing total attendance is equivalent to minimizing total missed attendance. Informally, it is best to avoid scheduling talks that are on a same profile in parallel, but we still need to join talks together in a tuple.

Let H denote the set of all n-tuples consisting of distinct talks, $H \subseteq X^n$. For each $e \in H$, we set $c_e := \sum_{p \in P} \max\{0, \sum_{i \in e} q(p)_i - 1\}$. In words: the coefficient c_e denotes the total missed attendance if the talks in the n-tuple e are scheduled in parallel. Notice that the coefficients c_e are in fact a generalization of the conflict matrix used in [19]. Indeed, the conflict matrix indicates the missed attendance at the level of a session if two talks are scheduled in parallel sessions, while our coefficient does the same for any n parallel talks.

Next, we set up an integer programming model using the binary variable x_e which is 1 if and only if all talks in n-tuple e are planned in parallel.

$$\text{Min} \sum_{e \in H} c_e x_e \tag{1}$$

$$\text{s.t.} \quad \sum_{e \in H : i \in e} x_e = 1 \qquad \forall\, i \in X \tag{2}$$

$$x_e \in \{0, 1\} \qquad \forall\, e \in H \tag{3}$$

Clearly, the objective function, Eq. (1), minimizes missed attendance. The first set of constraints, Eq. (2), ensures that every talk is included in exactly one n-tuple. Finally, Eq. (3) indicates that our decision variables x_e are binary.

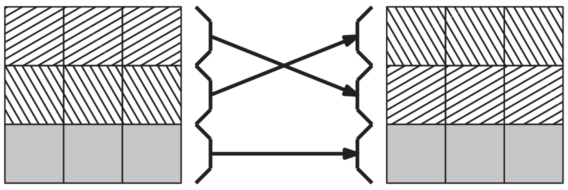
Fig. 2. Permuting two 3-tuples (rows) in a 3-block.

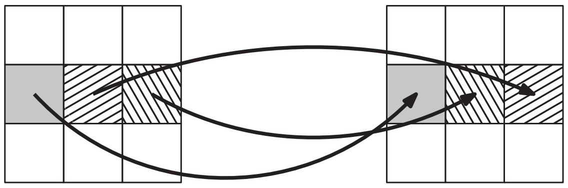
Fig. 3. Permuting two talks in a 3-tuple.

5.2. Phase 2: minimizing session hopping

Recall that an n-tuple refers to n talks that will take place in parallel. Phase 1 gives us $|T| = \frac{|X|}{n}$ such n-tuples; we use H^* to denote the set of selected n-tuples from phase 1. Here, in phase 2, our goal is to assemble these n-tuples into so-called k-blocks. Recall that, in line with the terminology introduced in Section 3, a k-block can be seen as an ordered set of k ordered n-tuples, thereby yielding n parallel sessions, each session consisting of k consecutive talks.

Consider a set of k n-tuples found in phase 1, say $e_1, e_2, \ldots, e_k$. Clearly, there are different ways to organize a set of k n-tuples into a k-block: one can permute the sequence of n-tuples e_1, e_2, ..., e_k, as illustrated in Fig. 2, and one can permute the n talks within each n-tuple e_i, as illustrated in Fig. 3. In total this gives $k!(n!)^k$ possibilities, i.e., given a set of k n-tuples, there are $k!(n!)^k$ distinct k-blocks corresponding to that set.

We remark here that some permutations are equivalent with respect to session hopping. Indeed, we need not consider permutations obtained by (i) reversing the order of the n-tuples, or (ii) changing the order of the talks in the first n-tuple.

We now define the *hopping number* of a participant in a k-block as the minimum number of session hops needed by that participant to attend the maximum number of talks in this k-block (s)he is interested in. We find the *hop coefficient* of a k-block by summing the corresponding hopping numbers over all participants.

Fig. 4 illustrates this using a 3-block with three parallel sessions, and a number of profiles as examples. The top left example shows that the participant is interested in the first and last talk in session 1. The hopping number for this profile will therefore be 0, as indicated by the full line; the participant can stay in session 1 and not miss any of his/her preferred talks. The top right example shows the profile of a participant interested in the first talk in session 2 and the last talk in session 1. In order to attend both talks, this participant will have to switch rooms exactly once, leading to a hopping number of 1. There are two alternative ways this participant can switch, one is indicated using the full line, the other using the dashed line. Similarly, in the bottom left example, a participant with this profile will have to switch exactly twice to attend all talks of his or her interest, as indicated by the full line. The final example, on the bottom right, shows a profile and a k-block where at a particular moment in time, more than one talk of interest is planned. In that case, we assume that the participant chooses talks such that (s)he can attend the maximum number of talks of his/her interest, while minimizing the number of required session switches. In the bottom right example this means that the participant will choose to stay in session 1 for the second

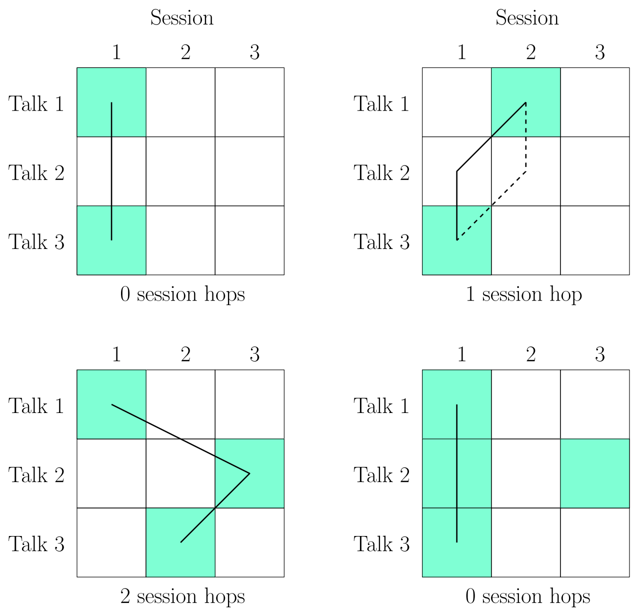

Fig. 4. Four session hopping examples using 3-blocks.

talk, as indicated by the full line, instead of switching to session 3 and then back to session 1.

In the following subsections, we describe two ways of actually constructing the k-blocks from the given n-tuples from phase 1: an exact approach and a heuristic approach. The exact approach refers to the fact that, given the set H^*, a solution is computed for which the sum of the hop coefficients is minimum. The heuristic approach is an approach where an approximation of the hopping number is used, and hence, no guarantee of optimality can be given. However, the heuristic approach will be very efficient computationally.

5.2.1. Exact dynamic programming approach

For each block consisting of n parallel sessions of length k, we can determine the minimum number of hops required by a participant p using a dynamic programming algorithm.

We define the set $N = \{1, \dots, n\}$ as the set of parallel sessions, indexed by j, and we define the binary indicator $A(t, j)$ which equals 1 if the profile of participant p indicates that (s)he would like to attend the talk in session j during the tth timeslot of this block, and zero otherwise. Let $H(t, j)$ be the minimum number of hops required by participant p in order to attend the talk in session j during the tth timeslot, while also attending a talk of his/her preference during each of the timeslots $1, \dots, t-1$ for which at least one talk is preferred by this participant. We can express the hopping number for participant p in this block b as follows:

$$H_p^b = \min_{\substack{j \in N: \\ A(x,j)=1}} H(x, j),$$

where x is the latest timeslot in the block for which this participant has at least one preferred talk. If a participant does not have a preference for any of the talks in a block b, then $H_p^b = 0$.

It is trivial to see that $H(1, j) = 0$ for all $j \in N$. Furthermore, it is not difficult to argue that a participant, in order to obtain a minimum number of hops up to timeslot i, should not switch from their current session (say j) to a different session (say j') unless in timeslot t (s)he has a preference for a talk in session j', while having no preference for the talk in session j. We use $\Delta_{j,j'}$ with j, $j' \in N$ as a binary indicator, which is 0 if $j' = j$ and 1 if $j' \neq j$. More formally, the following recursion holds for $i > 1$:

$$H(t, j) = \begin{cases} H(t-1, j) & \text{if } \sum_{j' \in N} A(t-1, j') = 0, \\ \min_{\substack{j' \in N: \\ A(t-1,j')=1}} \left(H(t-1, j') + \Delta_{j,j'}\right) & \text{if } \sum_{j' \in N} A(t-1, j') \neq 0. \end{cases}$$

The above implies that, for each participant p and for each block b, the hopping number can be determined using a dynamic programming algorithm. Thus, for each set of k n-tuples, we can compute a k-block b that minimizes the hop coefficient. The resulting value is denoted by w_b and represents the total number of session switches that will result from having this block b as part of the conference schedule.

We now build an integer programming model to minimize the number of session hops. We use $B(k)$ to denote the set of k-blocks ($k \in K$); recall from Section 3 that the format of the conference (r_k) is given. Let $\mathcal{B} = \cup_{k \in K} B(k)$, further, we write $e \in b$ to denote that block b contains e as an n-tuple. The binary variable y_b equals 1 if and only block b is included in the schedule.

$$\text{Min} \sum_{b \in \mathcal{B}} w_b y_b \tag{4}$$

$$\text{s.t.} \quad \sum_{b \in B(k)} y_b = r_k \qquad \forall\, k \in K \tag{5}$$

$$\sum_{b \in \mathcal{B}: e \in b} y_b = 1 \qquad \forall\, e \in H^* \tag{6}$$

$$y_b \in \{0, 1\} \qquad \forall\, b \in \mathcal{B} \tag{7}$$

The objective function minimizes the number of hops over all k-blocks. The first set of constraints, Eq. (5), ensure that we select the proper number of each k-block, according to the conference format. Eq. (6) makes certain that every n-tuple (input from phase 1) is used exactly once. The final constraint set enforces that our decision variables y_b are binary.

5.2.2. Heuristic approach

We now give an informal description of a heuristic that we used to construct the required k-blocks. For ease of exposition, we assume first that we need only 2-blocks or only 4-blocks; later, we will indicate how to modify the method when k-blocks with other values of k are required as well. Recall that we are given $|T|$ n-tuples from phase 1. Let us now build a complete, undirected, edge-weighted graph $G = (V, E)$ as follows: there is a node in V for each $e \in H^*$. The cost of a pair of nodes in V that correspond to, say, e_1, $e_2 \in H^*$, is computed as follows. Consider a pair of talks i and j such that talk i is from n-tuple e_1, and talk j is from n-tuple e_2. We count the number of participants that (i) wish to attend talk i, and do not wish to attend talk j but wish to attend some other talk from n-tuple e_2, and (ii) wish to attend talk j, and do not wish to attend talk i but wish to attend some other talk from n-tuple e_1. In this way we have identified the number of participants that will incur a hop when talks i and j are scheduled consecutively in a same session - let us call this number $d_{i,j}$. We compute this number for every pair of talks, where one talk is from e_1, and the other talk is from e_2, leading to n^2 $d_{i,j}$ numbers. To compute the cost for the edge in G between nodes in V that correspond to e_1 and e_2, we solve an assignment problem based on these $d_{i,j}$ numbers. Indeed, the resulting solution value of this assignment problem is the cost of the edge in V between the n-tuples e_1 and e_2; in addition, there is an optimal assignment of talks known in case this edge from G is selected. Having built the graph G, we now compute a minimum cost perfect matching in this graph, giving us pairs of n-tuples, i.e., 2-blocks. Notice that in case the schedule would consist of 2-blocks only, the heuristic could stop here. We now proceed by applying the procedure recursively: we consider each pair of n-tuples as an entity by itself. That is, we now build a graph, say G', where each pair of n-tuples just found corresponds to a node in V'. Again, we solve an assignment problem to find the cost of selecting a pair of 2-blocks in G'. More precise, given two pairs of 2-blocks (e_1, e_2) and (e_3, e_4), there are exactly four ways in which we can concatenate them: (e_1, e_2, e_3, e_4), (e_1, e_2, e_4, e_3), (e_2, e_1, e_3, e_4) and (e_2, e_1, e_4, e_3). For each of these four ways, we solve an assignment problem based on the (known) attendance in the second and third n-tuple. We keep the solution which has the smallest cost as the cost of an edge in G'. Now we compute a minimum cost perfect matching in G', giving us pairs of 2-blocks, i.e., 4-blocks, and we have found a solution.

It is clear that the cost that we base our computations on, is in fact an approximation of the true hopping coefficients. This means that the outcome of the heuristic is not necessarily leading to a solution with a minimum number of hops. Observe that the method is efficient: in order to compute all cost coefficients, we need to solve $O(\binom{|T|}{n})$ assignment problems of size $n \times n$ (where, in practice, n will be small), and two minimum cost matching problems.

Let us finally indicate how to modify the method when k-blocks for other values of k need to be found – we assume that $k \in \{2, 3, 4\}$, as is the case for all conferences we encountered. First, we run the heuristic as described. We select from the resulting solution the r_4 best 4-blocks. Next, we remove these n-tuples from the instance, and run the first step of the heuristic to find a set of 2-blocks. We select the best r_2 blocks, and remove the corresponding n-tuples from the instance. Then we add r_3 dummy n-tuples to the instance, where a dummy n-tuple corresponds to a node in V that has zero cost to each non-dummy n-tuple, and a high cost to another dummy node. Solving a minimum cost matching in G results in $2r_3$ 2-blocks, which we use to construct G'. However, we assign a high cost to edges between two 2-blocks that both have a dummy n-tuple, and of the four ways in which two pairs can be concatenated, we only consider options where the dummy n-tuple is in the first or the last position. Computing a minimum cost matching in G' results in r_3 4-blocks containing a single dummy n-tuple, which we can remove to arrive at a solution with the prescribed number of 3-blocks.

5.3. Phase 3: presenter availabilities

In this phase, we assign the blocks found in Phase 2 to timeslots while minimizing the number of violated speaker availabilities. As the order of the talks within a block has been settled, this phase will assign each selected k-block, and hence each talk, to a timeslot. We define $T_S \subseteq T$ as the set of timeslots that correspond to session starting times. The number of violated availabilities if block b is assigned to timeslot $t \in T_S$ is denoted by $u_{b,t}$, and can easily be computed from known presenters availabilities.

We use an assignment based integer programming formulation where $z_{b,t} = 1$ if block b is scheduled to start in timeslot $t \in T_S$, and 0 otherwise.

$$\text{Min} \sum_b \sum_t u_{b,t} z_{b,t} \tag{8}$$

$$\text{s.t.} \quad \sum_b z_{b,t} = 1 \qquad \forall\, t \in T_S \tag{9}$$

$$\sum_t z_{b,t} = 1 \qquad \forall\, b \in \mathcal{B} \tag{10}$$

$$z_{b,t} \in \{0, 1\} \qquad \forall\, b \in \mathcal{B}, t \in T_S \tag{11}$$

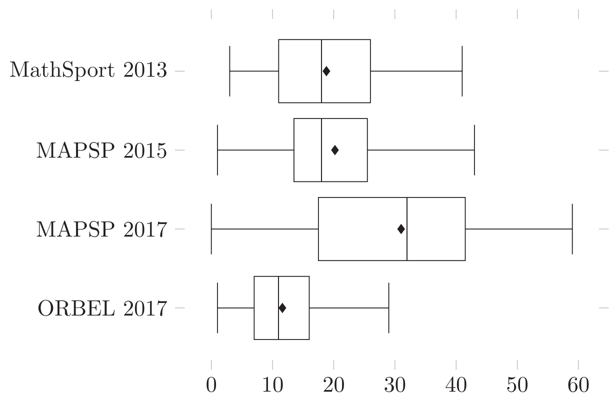

Fig. 5. Box plots of preferences/participant.

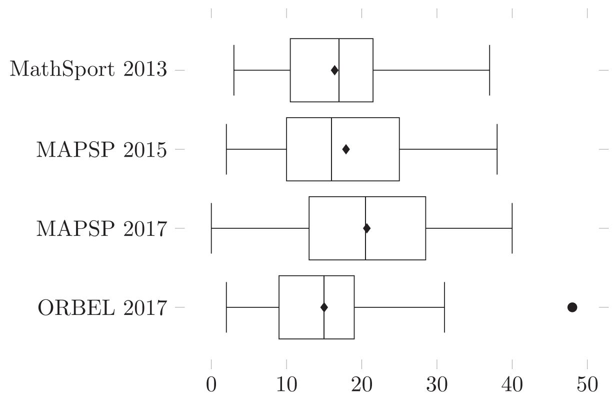

Fig. 6. Box plots of preferences/talk.

The objective function minimizes the total number of violated availabilities. The first set of constraints, Eq. (9), ensures that every timeslot at which sessions start gets assigned one block. The second set of constraints, Eq. (10), ensures that every block is assigned exactly once. Finally, we enforce our variables $z_{b,t}$ to be binary.

After phases 1, 2 and 3, we have composed the schedule for the conference. Now it is easy to see how a personalized optimal itinerary can be constructed for every participant. By constructing the k-blocks for the second phase, we already know the maximum number of preferred talks each participant can attend in every k-block, as well as how many session hops are required in order to actually attain that attendance. As a result, we can simply combine this information with the timeslots that were chosen in Phase 3 to present each participant with an individual itinerary.

6. Practical applications

In this section we will apply our three-phased conference scheduling approach to four practical cases: MathSport International (2013), MAPSP (2015 and 2017), and ORBEL (2017). These are medium-sized conferences with 2, 3, and 4 parallel sessions respectively. For each of these conferences, we sent an e-mail to all registered participants enquiring each participant for his/her profile. Figs. 5 and 6 show boxplots of the number of preferences given per participant and the number of preferences per talk respectively. One observation is that for ORBEL, a conference with talks covering a wide variety of topics, there were fewer preferences given by participants, and fewer preferences per talk compared to MathSport and MAPSP, which are more focused on specific topics. These (anonymous) profiles are publicly available.[1] The schedule obtained by applying our method was adopted by the conference organizers in each case. Furthermore, based on the profiles, we were able to select suitable session chairs for each session. Indeed, for each session we selected chairs among the participants that had expressed an interest for a maximal number of talks in that session.

6.1. MathSport 2013

MathSport International is a biennial conference dedicated to all topics where mathematics and sport meet. The fourth edition was organized in Leuven (Belgium) on June 5–7, 2013 and attracted 76 talks (apart from 3 keynote talks) and 97 participants. The conference featured two parallel sessions, and consisted of five 4-blocks and six 3-blocks.

Our preference elicitation resulted in 68 nontrivial profiles (a response rate of 70%), amounting to 1279 indicated preferences in total.

The first phase of scheduling MathSport, maximizing attendance, is an instance of CSP-2, which can be solved as a minimum weight perfect matching problem (see Theorem 1). For each pair of talks, the weight of an edge boils down to the number of participants that want to attend both talks. However, we had 4 speakers presenting two talks. These pairs of talks could obviously not be part of the matching, which we enforced by giving them a very high weight. We tackled this phase using a straightforward IP formulation. The optimal solution involved 42 scheduling conflicts, allowing the participants to attend 96.7% of the talks in their profile on average.

The MathSport 2013 conference was characterized by several presenter availability restrictions. Specifically, 10 talks could not be scheduled on given days, and for 2 other talks only 1 particular timeslot was acceptable. Hence, we gave priority to phase 3 over phase 2, meaning that we first grouped the pairs of talks into parallel sessions, for which the starting times were already set (using a slightly modified version of formulation (8)–(11)). The result was a timetable that did not violate any presenter unavailability.

Finally, we needed to decide for each group of paired talks to which session — the one in room A or the one in room B — the talks should be assigned, and in what order. As the capacity of both rooms was identical, minimizing session hopping was the only concern. Since there are only 8 (4) ways to organize a group of 4(3) pairs of talks into sessions, and 24 (6) different orders of these pairs within a session of 4(3) talks, this step took near-zero computation time.

6.2. MAPSP 2015 and MAPSP 2017

The workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP) is a biennial conference dedicated to scheduling, planning, and timetabling. The 12th edition of MAPSP was held on June 8–12 2015 in La Roche-en-Ardenne (Belgium) and the 13th edition was held on June 12–16 in Seeon-Seebruck (Germany). Both conferences featured three parallel series of sessions, spread over five days.

Specifically for MAPSP 2015, there were eight 3-blocks and three 2-blocks available for scheduling talks, leading to a total capacity for talks of 90. The MAPSP program committee accepted 88 talks, to which we added 2 dummy talks (corresponding to empty spaces in the conference schedule) in order to match the capacity. For MAPSP 2017, there were 87 talks, to be scheduled in seven 3-blocks and four 2-blocks.

We collected 78 nontrivial profiles for MAPSP 2015 and 58 for MAPSP 2017. The total number of indicated preferences were 1576 and 1799 respectively for MAPSP 2015 and 2017.

The first phase of scheduling MAPSP, maximizing attendance, is an instance of CSP-3. Remembering the coefficient c_e, as defined in Section 5.1, we have for each triple (3-tuple) of distinct talks $i, j, k \in X$ the number of missed attendance if talks i, j and k are scheduled in parallel: $c_{i,j,k}$. Note that this is easily computed using the profiles, as indicated in Section 5.1.

We used formulation (1)–(3), which for MAPSP 2015 amounts to $\binom{90}{3} = 117480$ variables and 90 constraints, and for MAPSP 2017 amounts to $\binom{87}{3} = 105995$ variables and 87 constraints. For MAPSP 2015, we obtained an optimal objective value of 155. Equivalently, the obtained triples allowed the participants to attend 1421 of the 1576 preferred talks according to the profiles. For MAPSP 2017, the optimal objective value was 478.

In the second phase of scheduling MAPSP, our goal is to assemble the triples into 3-blocks and 2-blocks such that session hopping is minimized. Recall that a 3-block, as well as a 2-block, consists of three parallel sessions, each taking place in a different room. The optimal objective value of the second phase is 120 for MAPSP 2015, which is the total number of hops for all participants. For MAPSP 2015, the minimum number of hops was 145.

Note that in order to arrive at a schedule, there is still freedom in the allocation of sessions to rooms. Indeed, the allocation of sessions to rooms does not influence the attendance or the number of session hops. This allows us to take the room capacity into account to some extent. In Section 5 we assumed that the rooms have infinite capacity. The available rooms at the MAPSP 2015 conference each had a different (and finite) capacity: these capacities equaled 170, 100 and 40 seats. So, with the three sessions of each 3- and 2-block known, we used the following strategy to allocate sessions to rooms. First, find in each session the talk with the largest number of votes, i.e., the talk i with maximum v_i. The session for which this number is minimal goes to the smallest capacitated room. Next, for the two remaining sessions, we sum the v_i's of the talks in each session, and put the session with the largest sum in the largest room. This system of allocation offers no hard guarantee that the room capacities are respected. However, it turned out that, using the system described above, room capacity was not an issue.

Finally, in the third phase, we need to identify a talk with a speaker and take into account the various availabilities of speakers in order to assign 2-blocks and 3-blocks to timeslots. For MAPSP 2015 a total of 13 speakers had availability restrictions (i.e. not being available on certain days), whereas MAPSP 2017 only had 4 speaker restrictions. This phase was easily handled by solving formulation (8)–(11). The solution resulted in schedules respecting all speaker availabilities, which were then implemented for MAPSP 2015 and 2017.

6.3. ORBEL 2017

ORBEL is the annual conference of the Belgian Operational Research Society, and serves as a meeting place for researchers working in Operational Research, Statistics, Computer Science and related fields. ORBEL 2017 took place February 2–3 2017 in Brussels (Belgium), and was the 31st edition of ORBEL. It featured four parallel series of sessions, spread over two days. Specifically, a total of one 2-block, two 3-blocks, and three 4-blocks were available to schedule talks, leading to a total capacity of 80 talks, which exactly corresponds to the number of accepted talks.

We collected 101 non-trivial profiles from 140 participants, leading to a total of 1200 indicated preferences.

The first phase of scheduling ORBEL, which maximizes attendance, corresponds to an instance of CSP-4. This leads to a total of $\binom{80}{4} = 1,581,580$ quadruples (4-tuples). However, not all these

[1] Data can be found at the following URL: http://feb.kuleuven.be/public/u0037710/cspinstances.zip.

Table 1
Computation times (in seconds).

	Phase 1	Phase 2
MathSport 2013	0.05	0.06
MAPSP 2015	3.89	0.12
MAPSP 2017	3.53	0.11
ORBEL 2017	61.03	0.03

Table 2
Hop results for exact and heuristic approach.

	Exact	Heur	Heur/Exact
MathSport 2013	141	141	1.00
MAPSP 2015	120	292	2.43
MAPSP 2017	145	207	1.43
ORBEL 2017	281	344	1.22

quadruples were feasible and hence needed to be filtered out. 30 out of 80 talks were classified as being ‘COMEX talks’. The organizing committee requested that the schedule ensures sessions consisting of only COMEX talks, such that the number of parallel COMEX sessions is minimal. Hence, we kept quadruples that had exactly 1 or 2 COMEX talks in parallel. In addition, 4 particular talks were to be grouped in an ‘ORBEL Award’ session. As they could not be scheduled in parallel, we filtered out all quadruples containing 2 or more such ORBEL Award talks. One of the jury members of the ORBEL Award also gave a talk at ORBEL, and hence could not be scheduled in parallel with ORBEL Award talks. Two presenters each had two talks, which could consequently not be in the same quadruple. Finally, 9 participants could not be present on Friday, which was the day the ORBEL Award took place. In other words: the talks of these participants were not scheduled in parallel to the ORBEL Award talks. In the end, we ended up with 889,573 feasible quadruples, a significant reduction from the 1,581,580 possible quadruples. We used formulation (1)-(3), and quickly found an optimal solution with objective value 100. In other words: participants could see 1100 of their 1200 preferred talks.

In the second phase of scheduling ORBEL, we assembled the 20 quadruples resulting from phase 1 into one 2-block, two 3-blocks and three 4-blocks in a way that minimizes session hopping, that ensures COMEX talks are together in 1 or 2 parallel sessions, and that groups the ORBEL Award talks in a single session. This resulted in an optimal solution of 281, the total number of hops for all participants, such that they can attend the maximum number of preferred talks. The most challenging aspect of exact approach of the second phase is undoubtedly to determine a k-block with minimal session hopping for each set of k quadruples (with $k \in \{2, 3, 4\}$), which took several hours.

Finally, in the third phase, we needed to take into account availabilities constraints in order to assign the various 2-, 3-, and 4-blocks to specific time-slots. Using an assignment-based formulation, similar to the one used for MAPSP, we found a schedule, respecting all availability constraints.

6.4. Computation times and heuristic results

The computation times for solving the IP formulations in phase 1 and 2 for the different conferences can be found in Table 1. All calculations were done using CPLEX 12.6.3 on a laptop with an Intel Core i7-4800 MP CPU @ 2.70 Ghz processor and 8GB RAM.

The results of the exact and the heuristic approach to solve phase 2 in terms of total number of hops are presented in Table 2. Notice that the numbers reported for the heuristic are not based on the approximated hop coefficients, but reflect the true number of hops. The quality of the heuristic is acceptable, with solution quality varying between 1 and 2.43 times the optimum. While the dynamic programming approach does not scale well with the size of the conference, the required computation times for the heuristic are always 1 s or less.

7. Conclusions

In this paper, we argue that conference scheduling is an important and relevant problem. Indeed, conferences require significant investments (time, money) from participants, which strongly motivates a good schedule. We identify the Conference Scheduling Problem, where the goal is to maximize total attendance based on given preferences of the speakers. This problem is shown to be easy in case of two parallel sessions, and becomes NP-hard for three or more parallel sessions. The main motivation for this research however, comes from a pragmatic origin: scheduling actual conferences. We describe how we applied our three phase scheduling method to four different conferences, MathSport 2013, MAPSP 2015 & 2017 and ORBEL 2017, and discuss these cases extensively.

A possible consequence of our method could be that the resulting sessions are incoherent, since their composition is based solely on participant preferences, and not on the topic of the talk. However, as the profiles of the participants tended to contain talks on similar topics, the resulting sessions were still relatively coherent.

Another concern is that the current approach does not treat all participants equally. By preferring many talks, a participant may have a larger impact on the conference schedule than a participant with a small number of preferences. This can be remedied by giving an appropriate weight to the preferences of each participant, or limiting the number of preferences each participant can express to e.g. the number of timeslots of the conference.

A final issue is the scalability of our approach. Although our method has been developed for medium-size conferences, the question arises to what extent it scales to much larger conferences. This is relevant both when eliciting participants’ preferences, as well as computationally. The latter can be accommodated by solving the second phase using our heuristic approach, as the bottleneck appears to be the computation of the hopping coefficient for all k-blocks, rather than solving the IP formulations. In case preference elicitation is unpractical due to the high amount of talks, our approach could be applied on the level of streams or tracks. Indeed, talks in large conferences are often from the beginning (i.e., when submitting an abstract) assigned to streams, and the conference schedule is typically based on track segmentation. As not all streams cover the full length of the conference, there would be possibilities to minimize overlap between pairs of streams that many participants would like to attend. Furthermore, if overlap is unavoidable, the streams could be allocated to rooms which are close to each other, facilitating session hopping. In fact, eliciting stream preferences would provide a good idea of the required room capacities for each stream.

Acknowledgments

This research was supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy P7/36. The authors wish to thank the referees for their constructive remarks.

References

[1] Vangerven B, Ficker A, Goossens D, Passchyn W, Spieksma F, Woeginger G. Conference scheduling - a personalized approach. In: Proceedings of the 11th international conference on practice and theory of automated timetabling (PATAT-2016); 2016. p. 371–83.
[2] Gremillet D. Paradox of flying to meetings to protect the environment. Nature 2008;455(7217):1175.
[3] Grant P. The dilemma of attending (or not) scientific conferences. Can J Physiol Pharmacol 2014;92(1):v.

[4] Nathans J, Sterling P. Point of view: how scientists can reduce their carbon footprint. eLIFE 2016.
[5] Ioannidis JP. Are medical conferences useful? and for whom? J Am Med Assoc 2012;307(12):1257–8.
[6] Sampson S. Practical implications of preference-based conference scheduling. Prod Oper Manage 2004;13(3):205–15.
[7] Thompson G. Improving conferences through session scheduling. Cornell Hotel Restaurant Administration Q 2002;43(3):71–6.
[8] Yingping H, Zhang X, Alexander PS. A heuristic algorithm for optimizing business matchmaking scheduling. Int J Oper Res Inf Syst 2012;3.4:59–73.
[9] Ernst AT, Mills R, Welgama P. Scheduling appointments at trade events for the australian tourist commission.. Interfaces 2003;33(3):12–23.
[10] Ernst AT, Singh G, Weiskircher R. Scheduling meetings at trade events with complex preferences. In: Proceedings of 18th international conference on automated planning and scheduling (ICAPS); 2008. p. 76–82.
[11] Potthoff R, Munger M. Use of integer programming to optimize the scheduling of panels at annual meetings of the Public Choice Society. Public Choice 2003;117(1):163–75.
[12] Potthoff R, Brams S. Scheduling of panels by integer programming: results for the 2005 and 2006 New Orleans meetings. Public Choice 2007;131(2):465–8.
[13] Edis E, Sancar Edis R. An integer programming model for the conference timetabling problem. CBU J Sci 2013;9(2):55–62.
[14] Nicholls M. A small-to-medium-sized conference scheduling heuristic incorporating presenter and limited attendee preferences. J Oper Res Soc 2007;58(3):301–8.
[15] Eglese R, Rand G. Conference seminar timetabling. J Oper Res Soc 1987;38(7):591–8.
[16] Sampson S, Weiss E. Increasing service levels in conference and educational scheduling: a heuristic approach. Manage Sci 1995;41(11):1816–25.
[17] Sampson S, Weiss E. Designing conferences to improve resource utilization and participant satisfaction. J Oper Res Soc 1996;47(2):297–314.
[18] Gulati M, Sengupta A. TRACS: tractable conference scheduling. In: Proceedings of the decision sciences institute annual meeting (DSI 2004); 2004. p. 3161–6.
[19] Le Page Y. Optimized schedule for large crystallography meetings. J Appl Crystallogr 1996;29(3):291–5.
[20] Ibrahim H, Ramli R, Hassan M. Combinatorial design for a conference: constructing a balanced three-parallel session schedule. J Discrete Math Sci Cryptography 2008;11(3):305–17.
[21] Quesnelle J, Steffy D. Scheduling a conference to minimize attendee preference conflicts. In: Hanzálek Z, Kendall G, McCollum B, Sũcha P, editors. Proceedings of the 7th multidisciplinary international conference on scheduling: theory and applications (MISTA); 2015. p. 379–92.
[22] Tanaka M, Mori Y, Bargiela A. Granulation of keywords into sessions for timetabling conferences. In: Proceedings of soft computing and intelligent systems (SCIS 2002); 2002. p. 1–5.
[23] Tanaka M, Mori Y. a hybrid grouping genetic algorithm for timetabling of conference programs. In: De Causmaecker P, Burke E, editors. Proceedings of the 4th international conference on the practice and theory of automated timetabling (PATAT 2002); 2002. p. 421–40.
[24] Zulkipli F, Ibrahim H, Benjamin A. Optimization capacity planning problem on conference scheduling. In: Proceedings of the business engineering and industrial applications colloquium (BEIAC 2013); 2013. p. 911–15.
[25] Martin A. A personalised conference programme advisor. OR Insight 2005;18(2):22–30.
[26] Lovász L, Plummer MD. Matching theory. Annals of discrete mathematics, vol. 29. AMS Chelsea Publishing; 1986.
[27] van Rooij JMM, van Kooten Niekerk ME, Bodlaender HL. Partition into triangles on bounded degree graphs. Theory Comput Syst 2012;52(4):687–718.