# Conference Scheduling

## A Personalized Approach

**Bart Vangerven · Annette M.C. Ficker · Dries R. Goossens · Ward Passchyn · Frits C.R. Spieksma · Gerhard J. Woeginger**

**Abstract** Scientific conferences are an essential part of academic research. It falls upon the organizers to develop a schedule that allows the participants to attend the presentations of their interest. We present a combined approach of assigning presentations to rooms and time slots, grouping presentations 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 MAPSP2015 conference.

**Keywords** Conference scheduling · computational complexity · case study · integer programming

## 1 Introduction

Conferences are an essential aspect of (academic) research, as they allow researchers to present their work and receive feedback, as well as to learn from attending presentations (or discussion panels). On the other hand, conferences require a considerable

This research is supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office.

Bart Vangerven · Annette M.C. Ficker · Ward Passchyn · Frits C.R. Spieksma
KU Leuven, Faculty of Economics and Business, Naamsestraat 69, 3000 Leuven, Belgium
E-mail: {bart.vangerven, ward.passchyn, annette.ficker, frits.spieksma}@kuleuven.be

Dries R. Goossens
Ghent University, Faculty of Economics and Business Administration, Tweekerkenstraat 2, 9000 Ghent, Belgium
E-mail: dries.goossens@ugent.be

Gerhard J. Woeginger
Eindhoven University of Technology, Department of Mathematics, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
E-mail: g.woeginger@tue.nl

effort in terms of time (e.g. preparing presentations, finding time in busy work schedules, traveling time) and money (e.g. registration fees, traveling expenses, hotels) from their attendees. Given these investments, it is the responsibility of the organizers to develop a schedule that allows participants to attend the presentations of their interest. Typically, a conference schedule groups presentations into *sessions*; consecutive sessions are separated by a break. Furthermore, to reduce the duration of the conference, several sessions (presentations) take place at the same moment in time, i.e. they are scheduled *in parallel*. Consequently, the attendees may be confronted with several preferred presentations overlapping at some time (i.e. a *scheduling conflict*), while at other times they find nothing of interest in the schedule.

One popular approach to schedule conferences is *track segmentation* (Sampson 2004). The organizer groups presentations 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 each conference attendee would be interested only in presentations from a single track, then he or she 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 segmentation is not very effective if participant preferences are diverse, and not restricted to one particular topic or method.

In this work, the participant is expected to provide a list of preferred presentations, 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, but as a secondary goal, we want to minimize *session hopping*. Indeed, confronted with multiple presentations of interest scheduled in overlapping sessions, a session hopper moves between several sessions in order to attend as many of his or her preferred presentations as possible. Session hopping is clearly an indication of participants being confronted with a schedule which is not optimal given their preferences, and is typically experienced as disturbing by presenters and their audiences. Moreover, the session hopper still tends to miss parts of the preferred presentations, due to the time it takes to switch rooms and presenters not always starting at exactly the scheduled time. Finally, although our focus is on the attendee, we also take into account presenter availabilities to some extent.

Our contributions with respect to previous research on conference scheduling are as follows. First of all, we consider conference scheduling at the level of the presentations, which is a finer granularity than sessions, or even streams, as is common in most other research papers. We also account for session hopping, which is either assumed to be forbidden or non-existing in the literature, as opposed to regular attendee practice. Furthermore, we present a combined approach of assigning presentations to rooms and time slots, grouping presentations into sessions, and deciding on an optimal itinerary for each participant. Finally, our model has been applied to schedule the MAPSP 2015 conference, which is a medium-sized event, with about 100 participants. It includes 88 non-plenary talks to be scheduled over 5 days and 3 rooms. Before the start of the conference, we collected preferences from registered participants; the schedule resulting from our method has been adopted in practice.

We provide an overview of related work in Section 2. A detailed problem definition, together with computational complexity results are given in Section 3. This is followed by a description of our solution method in Section 4. Finally, we present a case study on the MAPSP 2015 conference in Section 5. We finish with conclusions in Section 6.

## 2 Literature review

Thompson (2002) 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.

### 2.1 Presenter-based perspective

Potthoff and Munger (2003) discuss a problem where sessions need to be assigned to time periods (rooms are ignored). The authors assume that the clustering of presentations 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 (2007), 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 (2013) consider a very similar problem, but at the level of presentations instead of sessions. Each presentation has a given topic, and should be assigned to a session and a time period, such that all presentations in each session have the same topic, and the occurrence of simultaneous sessions with the same topic is minimized. Furthermore, the number of presentations in different sessions with same topic should be balanced, and some presentations 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 presentations on one of 10 topics, to be scheduled into sessions of at most 5 talks, over 12 time periods.

Nicholls (2007), like Potthoff and Munger (2003), 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 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. Attendee 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 (2007) 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 (1987), 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 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 (1995) extend the Eglese and Rand (1987) 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 (2004) 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 presentations, which was used to rank the sessions. A post-conference survey reveals 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 resource decision issues that might be faced by a conference organizer. For instance, Sampson and Weiss (1996) discuss tradeoffs between the length of the conference, the number of offerings per session and participant satisfaction, and study the sensitivity of participant satisfaction on room availability and the utilization of time slots and seating capacity.

Gulati and Sengupta (2004) enhance the problem description by Sampson and Weiss (1995), by augmenting the objective function with a prediction of the popularity of talk, based on reviewers' assessments of the submissions and linked with time slot preferences of attendees (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 (2004) develop a solution method called TRACS (TRActable Conference Scheduling), which is essentially a greedy algorithm; no empirical results or computational analysis are reported though.

The conference scheduling problem discussed by Thompson (2002) is also similar to that of Sampson and Weiss (1995). However, Thompson allows meeting rooms to have different capacities and takes into account that not all rooms are available all the time. His method, a constructive heuristic followed by an iteration of perturbation and simulated annealing steps, is not presented in detail. 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 to 5 sessions and each of the 175 participants have provided between 0 and 8 preferred sessions (not ranked, nor weighted). The author finds that his heuristic outperforms randomly, as well as manually generated schedules.

Le Page (1996) 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 session $i$ 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 (2008) focus on a conference scheduling problem where presentations need to be assigned to time slots (spread over a number of days) in 3 parallel tracks. Each presentation belongs to a field, and the schedule should be such that presentations of the same field do not occur simultaneously. Furthermore, it should be avoided to schedule presentations 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 presentations into sessions. Moreover, the sequence of the presentations within a track on one day is of no importance, and all presentations 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 (2015), presentations 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 presentation and must be able to attend each one of them. Some presentations are required to be offered multiple times. Quesnelle and Steffy (2015) 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 presentations. 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 presentations into sessions is not included in this problem, in fact, as in Ibrahim et al (2008), each presentation could be seen as a session.

### 2.3 Related problems

The problem of grouping presentations into coherent sessions, given one or more keywords for each presentation, is discussed by Tanaka et al (2002) and Tanaka and Mori (2002). The objective function is a non-linear utility function of common keywords, with the underlying idea that papers of the same session have as many common keywords as possible, provided that the number of presentations is balanced over the sessions. This problem is tackled using Kohonen's self-organizing maps (Tanaka et al 2002) and a hybrid grouping genetic algorithm (Tanaka and Mori 2002). 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 (2013) ignore session coherence as they attempt to group presentations 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 presentation, based on preferences from the participants, the goal is to assign presentations to sessions, such that the sum of the presentation weights is balanced over the sessions. The authors present a goal programming method, which is applied to one case, involving 60 presentations to be grouped into 15 sessions.

Martin (2005) 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 an web-based approach, keyword preferences are elicited and matched with keywords supplied by presentations, in order to produce an aggregate rating for each presentation. 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

Given a set of talks $X$, $n$ talks are to be scheduled in parallel at each point in time. Equivalently, we will have $n$ parallel series of sessions, where each session consists of $k$ talks. Without loss of generality, we may assume that the number of talks $|X|$ is divisible by $n$ and by $k$, otherwise we add dummy talks until it is. Additionally, we have the set of timeslots $T$, with $|T| = \frac{|X|}{n}$. For every participant $p \in P$, we have a *profile*.

**Definition 1** A profile of a participant is represented by a binary vector of length $|X|$, where each component equals 1 if and only if the participants wishes to attend the corresponding talk. A profile consisting of only 0 entries is called a trivial profile.

In other words, a profile represents the preferences of a participant. The profiles also give us, for each $x \in X$, the parameter $v_x$: the number of attendees for talk $x$ in case when all talks are scheduled consecutively ($n = 1$). Observe that $\sum_{x \in X} v_x$ is an upper bound for the total attendance. We assume that there are $n$ rooms with infinite capacities.

**Definition 2** The *Conference Scheduling Problem with n parallel sessions* (CSP-*n*) seeks to assign every talk to a session and a timeslot, while minimizing the total number of missed attendance, given the participants' profiles.

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 reduce CSP-2 to a minimum weight perfect matching problem, which is polynomially solvable (Lovász and Plummer 1986). Given a graph $G = (V,E)$, a matching in $G$ is a set of pairwise non-adjacent edges. A perfect matching then 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. Construct a complete graph $G = (V,E)$, with $|V| = |X|$ and talk in CSP-2 corresponding to exactly one node in $G$. For every distinct pair of talks $i$ and $j \in X$: calculate a coefficient $c_{i,j}$ denoting how much attendance is missed if both talks $i$ and $j$ are planned simultaneously, based on the participants' profiles. There is an edge $e \in E$ in $G$ for every distinct pair of nodes $i$ and $j$, and every such edge has as coefficient $c_{i,j}$. Now it is easy to see that a minimum weight perfect matching in this graph $G$, corresponds to an optimal solution of the CSP-2: simply plan the matched talks simultaneously. Then, assign the parallel talks to timeslots in any order. $\square$

**Theorem 2** *CSP-n is an NP-hard problem 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 a transformation from the *Triangle Partition Problem* (TPP), which is known to be NP-complete even for graphs with a maximal degree of at most four (van Rooij et al 2012). An instance of the TPP is a graph $G = (V,E)$ with $|V| = 3q$, where $q \in \mathbb{N}_0^+$. 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 $q$ disjoint sets $V_1, V_2, \ldots, V_q$ 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. For each pair of nodes $(i, j)$ with $(i, j) \notin E$ (i.e. a non-existing edge), we add a participant with exactly two preferences, namely those talks corresponding to $i$ and $j$. We have now completely specified an instance of CSP-3.

Suppose we have a yes-instance of TPP, then $q$ disjoint sets exist each containing exactly 3 vertices. These vertices correspond to three parallel talks as follows: the talks corresponding to the nodes in the triangle are scheduled simultaneously; parallel talks are assigned to timeslots in any order. Note that no participant misses a talk, because of the connected triangles; edges correspond to talks that can be scheduled together without missing any attendance. Thus, a yes-instance of CSP-3 is obtained. Suppose we have a yes-instance of the decision variant of CSP-3, then we know which talks are in parallel. From this, we can find a partition into triangles; simply select the nodes corresponding to the parallel talks as the nodes of a triangle. These nodes correspond to triangles, because otherwise there would have been missed attendance. From this it immediately follows that CSP-$n$ is NP-hard for $n \geq 3$. $\square$

Note that 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 (the problem would be trivial if each participant had only one preferred talk). Furthermore, our result adds to the result that the preference conference optimization problem (PCOP) is NP-hard by Quesnelle and Steffy (2015). Indeed, PCOP takes into account room and presenter availabilities, as well as presenters that can present more than once and presentations offered multiple times. In CSP-$n$, we assume that rooms have infinite capacity and are available throughout the conference. CSP-$n$ also assumes that every presentation takes place exactly once and has one presenter, who has no availability restrictions. Hence, we have shown that the complexity of preference-based conference scheduling arises already when focussing solely on maximizing attendance.

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 have no impact on attendance. Hence, we aim to minimize the number of session hops as a secondary goal. This will settle the composition of the sessions, taking into account the presentations that are to be scheduled in parallel in order to maximize attendance. However, the resulting schedule still leaves room for deciding the timeslots on which these sessions are scheduled. We allow each presenter $q \in Q \subseteq P$ to express his or her availabilities with respect to the timeslots: $a_{q,t}$ equals 1 if presenter $q$ is available to give a talk on timeslot $t$, and 0 otherwise. Finally, we assign the (parallel) sessions to timeslots, minimizing the number of violated presenter availabilities. Thus, the resulting conference scheduling problem has three objectives, maximizing attendance, minimizing session hopping, and satisfying presenter availabilities, which are considered hierarchically and in this order.

## 4 Method

In this section we will explain a hierarchical three-phased approach to scheduling conferences. In the first phase (see Section 4.1), we maximize attendance, based on the participants' profiles, which corresponds to solving CSP-$n$. In the second phase, we seek to minimize the number of session hops, given that total attendance is maximal. This will be explained in Section 4.2. 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. This is described in Section 4.3.

## 4.1 Phase 1: minimizing total missed attendance

It should be obvious that when minimizing total missed attendance, it is best to avoid scheduling talks on the same profile in parallel. Let us define the following cost-coefficient for each $n$-tuple of distinct talks $e$. The coefficient $c_e$ denotes the total missed attendance if the talks in the $n$-tuple $e$ are scheduled in parallel. It should also be clear that the profiles allow us to compute $c_e$. Note that our coefficient $c_e$ is in fact a generalization of the conflict matrix (Le Page 1996). 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. In other words, our concept is a generalization of the concept of the conflict matrix, looking at the level of talks, which is a finer granularity than sessions. The coefficient $c_e$ is easily calculated as follows. For every participant $p$, we calculate the number of indicated talks in the $n$-tuple $e$, and we subtract one from that number if it is greater than or equal to 1. The sum of this number over all participants for the $n$-tuple $e$ will be $c_e$.

Next, we set up an integer programming model using an index $e$ for each $n$-tuple, containing distinct talks. The variable $x_e$ is 1 if all talks in the corresponding $n$-tuple are planned in parallel, and 0 otherwise.

$$\text{Min} \sum_e c_e x_e \tag{1}$$

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

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

Clearly, the objective function, Equation 1, minimizes the missed attendance. The first set of constraints, Equation 2, ensures that every presentation is included in exactly one $n$-tuple. Finally, Equation 3 indicates that our decision variables $x_e$ are binary. Of course, the number of variables in this formulation is a potential bottleneck. There are exactly $\binom{|X|}{n} = \frac{|X|!}{(|X|-n)!n!}$ variables and $|X|$ constraints.

## 4.2 Phase 2: minimizing session hopping

Recall that an $n$-tuple refers to $n$ talks that will take place in parallel. Phase 1 gives us $\frac{|X|}{n}$ such $n$-tuples. Here, in phase 2, our goal is to assemble the $n$-tuples into $k$-blocks.

**Definition 3** A $k$-block consists of a set of $nk$ presentations, grouped into $n$ parallel sessions, such that each session consists of $k$ ordered talks.

Consider now $k$ $n$-tuples from phase 1, say $e_1$, $e_2$, ... and $e_k$. We will associate a variable to each set of $k$ distinct $n$-tuples. 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 $e_1$, $e_2$, ...,

$e_k$, and one can permute the $n$ talks within each $n$-tuple $e_i$. In total this gives $k!(n!)^k$ possibilities, i.e., given $k$ $n$-tuples there are $k!(n!)^k$ distinct $k$-blocks corresponding to it. For each $k$-block, it is possible to compute how many times a participant with a particular profile needs to switch between different sessions in order to attend the maximum number of talks in this $k$-block he/she is interested in. More precise, we 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 indicated talks in that $k$-block. Given a profile, and a $k$-block, we can compute this profile's

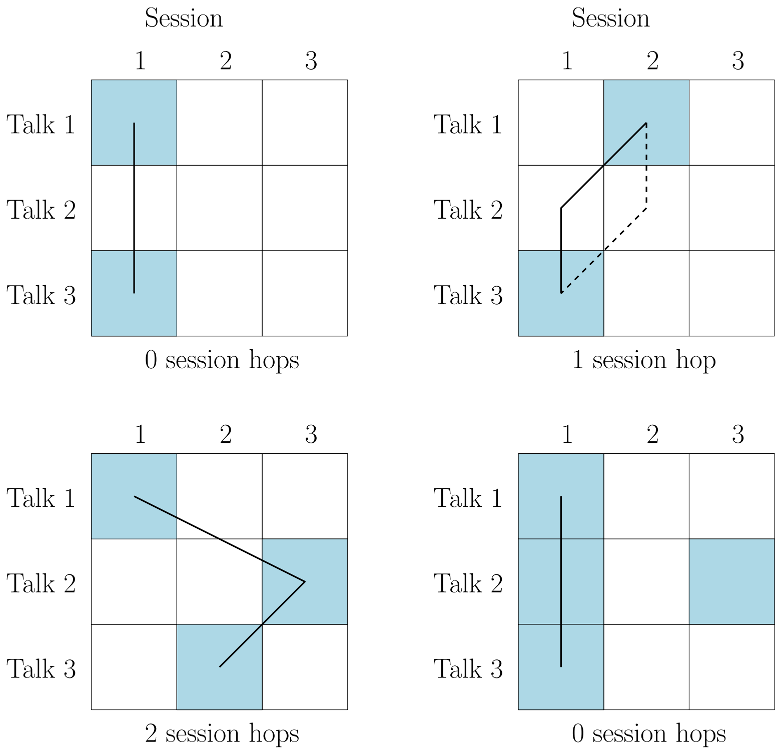

**Fig. 1** Session hopping examples: 3-blocks with profiles of a single participant.

hopping number. Figure 1 illustrates this using 3-blocks 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 indicated talks. The top right example shows the profile of a participant interested in the first presentation in session 2 and the last presentation in session 1. In order to attend both talks, this participant would have to switch rooms exactly once. 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, the participant with this profile would 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 an interesting profile which requires more thought. It is possible that a profile is such that at one particular moment in time, more than one talk of interest is planned. If that is the case, we assume that the participant chooses talks such that he or she can attend the maximum number of talks of his interest, while minimizing the number of required session switches. In the bottom right example that means that the participant will choose to stay in session 1 for the second talk, as indicated by the full line, instead of switching to session 3 and then back to session 1. Given a $k$-block, we obtain its hop coefficient by summing the corresponding hopping numbers over all profiles.

For each set of $k$ $n$-tuples, we compute the $k$-block $b$ that minimizes the hopping number. The resulting value is denoted by $w_b$ and represents the total number of session switches that will result from having this block as part of the conference schedule. We use the following integer programming model to, given the maximum attendance found in the first phase, minimize the number of session hops. The variable $y_b$ equals 1 if block $b$ is included in the schedule, and 0 otherwise.

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

$$\text{s.t.} \sum_{b:e\in b} y_b = 1 \qquad \forall e \tag{5}$$

$$y_b \in \{0,1\} \qquad \forall b \tag{6}$$

The objective function, Equation 4, minimizes the number of hops over all possible blocks. The first set of constraints, Equation 5, are such that every $n$-tuple (input from phase 1) is used exactly once. Finally, Equation 6 enforces that our decision variables $y_b$ are binary. The number of variables in this formulation is $\binom{|X|/n}{k}$; the number of constraints is $\frac{|X|}{n}$.

After phase 2 we have composed the $k$-blocks that, given $n$-tuples that maximize attendance, minimize session hopping. Once we have the $k$-blocks, 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 indicated presentations 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 for all participants and present each participant an individual itinerary.

## 4.3 Phase 3: presenter availabilities.

In this phase, we assign the blocks from phase 2 to block-timeslots while minimizing the number of violated speaker availabilities (a block-timeslot consists of $k$ consecutive timeslots as defined in section 3). As the order of the talks within a block has been settled previously, this phase will effectively assign each talk to a timeslot. The number of violated availabilities if block $b$ is assigned to block-timeslot $t'$ is denoted by $u_{b,t'}$, and can easily be computed from the presenters' availabilities $a_{q,t}$. We use an assignment based integer programming formulation where $z_{b,t'} = 1$ if block $b$ is scheduled in block-timeslot $t'$, and 0 otherwise.

$$\text{Min} \sum_{b,t} u_{b,t'} z_{b,t'} \tag{7}$$

$$\text{s.t.} \sum_{b} z_{b,t'} = 1 \qquad \forall t' \in T^b \tag{8}$$

$$\sum_{t'} z_{b,t'} = 1 \qquad \forall b \in B \tag{9}$$

$$z_{b,t'} \in \{0,1\} \qquad \forall b \in B, t' \in T^b \tag{10}$$

The objective function, Equation 7, minimizes the total number of violated availabilities. The first set of constraints, Equation 8, ensures that every block-timeslot gets assigned exactly one block. The second set of constraints, Equation 9, ensures that every block is assigned exactly once. Finally, Equation 10 enforces our variables $x_{b,t}$ to be binary.

## 5 Case Study: MAPSP 2015

In this section we will apply our three-phased approach of scheduling conferences to a practical case: the scheduling of the 12th workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2015). MAPSP is a biennial workshop dedicated to scheduling, planning, and timetabling. It was held on June 8-12 2015 in La Roche-en-Ardenne. MAPSP featured three parallel sessions during five days.

Specifically for MAPSP, on Monday morning there is a block of three parallel sessions, each consisting of three consecutive talks (a 3-block), there is a 3-block on Monday afternoon, and there is a 2-block on Monday afternoon (leading to $9 + 9 + 6 = 24$ talks on Monday). Tuesday and Thursday are like Monday, while both Wednesday and Friday feature one 3-block. Thus, the total capacity for talks equals $24 + 24 + 9 + 24 + 9 = 90$. 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. This structure can be seen in Figure 2. After the registration deadline passed, we sent an e-mail to all registered participants enquiring each participant for his/her profile (anonymously, when preferred). As a result, we received 78 nontrivial profiles from the 120 participants. The total number of indicated preferences was 1576. Some other interesting statistics can be found in Table 1.

**Table 1** MAPSP profile statistics

| | Minimum | Average | Maximum |
|---|---|---|---|
| Preferences/participant | 1 | 20,2 | 43 |
| Preferences/presentation | 2 | 17,91 | 38 |

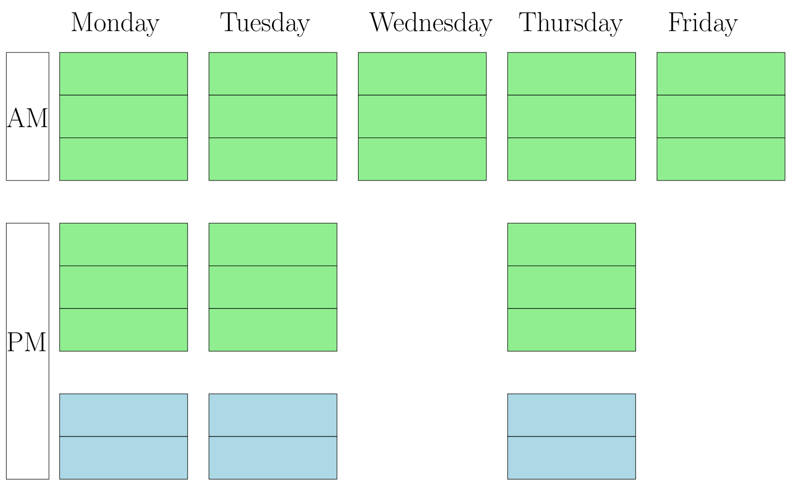

**Fig. 2** A general overview of the MAPSP parallel sessions.

## 5.1 Phase 1

The first phase of scheduling MAPSP, maximizing attendance, is an instance of CSP-3. Remembering the coefficient $c_e$, as defined in Section 4.1, 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 4.1.

We used formulation (1)-(3), which for this problem instance amounts to $\binom{90}{3} =$ 117480 variables and 90 constraints. Although we were prepared to use column generation, and even - if necessary - branch-and-price to solve the model above for our instance, it turned out that, using Cplex 12.5.1 (on a laptop with an Intel Core i7-4800 MP CPU @ 2.70Ghz processor and 8 GB RAM), we could solve this instance in less than 8 seconds; a very reasonable computation time. We obtained an (optimal) objective value of 155. Equivalently, the obtained triples allow the participants to attend 1421 of the 1576 indicated preferred talks according to the profiles.

## 5.2 Phase 2

Recall that each triple from phase 1 refers to three talks that will take place in parallel. In phase 2, our goal is to assemble the triples into eight 3-blocks, and three 2-blocks. Recall that a 3-block, as well as a 2-block, consists of three parallel sessions, each taking place in a different room.

Observe that the model as presented in Section 4.2 is easily adjusted for generating both 3-blocks and 2-blocks. To that end, we use $w_b$ to denote the hopping number of a 3-block $b$ and $w_{b'}$ to denote the hopping number of a 2-block $b'$. The modified formulation, that generates exactly the eight 3-blocks and three 2-blocks that are required, is presented below. The variable $y_b$ equals 1 if 3-block $b$ is included in the schedule, and 0 otherwise. The variable $y'_{b'}$ equals 1 if 2-block $b'$ is included in the schedule, and 0 otherwise.

$$
\begin{aligned}
\text{Min} & \sum_{b} w_b y_b + \sum_{b'} w_{b'} y'_{b'} & \\
\text{s.t.} & \sum_{b} y_b = 8 & \\
& \sum_{b'} y'_{b'} = 3 & \\
& \sum_{b:e\in b} y_b + \sum_{b':e\in b'} y'_{b'} = 1 & \forall e \\
& y_b \in \{0,1\} & \forall b \\
& y'_{b'} \in \{0,1\} & \forall b'
\end{aligned}
$$

This modified formulation resulted in 3683 binary variables and 32 constraints. Notice that the number of variables is less compared to the first formulation: we could solve our second phase MAPSP 2015 instance using Cplex 12.5.1 in less than 0.5 seconds (again on a laptop with an Intel Core i7-4800 MP CPU @ 2.70Ghz processor and 8 GB RAM). The optimal objective value of the second phase is 120, which is the total number of hops for all participants.

It is interesting to note that the resulting sessions could be incoherent, since their composition is based solely on participant preferences, and not on the topic of the presentation. If one were to use track-based scheduling, this would be much less of a problem (provided of course that every talk is correctly categorized into a track). However, as the profiles of the MAPSP participants tended to contain presentations on similar topics, the resulting sessions were still quite coherent.

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 4 we assumed that the rooms have infinite capacity. The available rooms in our MAPSP case, however, each had a different (and finite) capacity: these capacities equalling 170, 100 and 40 seats. So, with the 3-blocks and 2-blocks known, and hence 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. The session for which this number is minimal goes to the smallest capacitated room. Next, for the two remaining sessions, we sum the votes, and put the session with the largest sum in the largest room. This system of allocation is strictly speaking no hard guarantee that the room capacities are respected. It turned out that, using the system described above, room capacity was not an issue.

## 5.3 Phase 3

Here 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 time-slots. We used an assignment based formulation which features $11 \times 11$ variables (since the total number of 2-blocks and 3-blocks equals 11). The model assigns every block to a timeslot, minimizing speaker availability violations. In total, 13 speakers had availability restrictions (i.e. not being available on certain days). There were 5 presenters who were unavailable to present on Monday, there was 1 presenter unavailable to present on Tuesday, and there were 9 presenters unavailable on Friday. Table 2 gives an overview of how many speakers in every 3- and 2-block were unavailable to present on a certain time.

**Table 2** Number of speakers per block unavailable to present at a certain time.

| | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | 0 |
| | 1 | 1 | 0 | 0 | 3 |
| | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 |
| | 1 | 0 | 0 | 0 | 0 |
| Blocks | 1 | 0 | 0 | 0 | 2 |
| | 0 | 0 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 | 0 |

The solution resulted in a schedule respecting all speaker availabilities, which was then implemented for MAPSP 2015. 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 Conclusions

In this paper, we first argued 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 surveyed the literature, followed by two interesting theoretical results. The first results is that preference based conference scheduling is easy if the conference has two parallel sessions. We then proved the NP-hardness of preference based conference scheduling for conferences with three or more parallel sessions, thereby closing a theoretical gap in the literature. The main motivation for this research however, was a practical application: the scheduling of MAPSP 2015. We discuss this case and how we used a three phase approach to actually schedule MAPSP 2015. In a first phase, based on all preferences we received from registered participants, we maximized the possible attendance using an integer programming model. In the second phase, given the maximum attendance obtained in the first phase, we minimized the number of required session hops in order to obtain that maximum attendance using integer programming, thereby making up the blocks in the MAPSP 2015 schedule. To the best of our knowledge, our approach is the first to deal with session hopping. Finally, in a third phase, we took into account presenter availabilities. The resulting schedule was then implemented for the actual conference.

Although our approach has been developed for a medium-size conference, the question may arise to what extent it scales for much larger conferences, from a practical point of view (e.g. eliciting participants' preferences over a large number of talks) as well as computationally (e.g. solving the integer programming models). Apart from that, this research could be extended in a number of interesting and useful ways. Some directions for future research are taking into account speakers that are presenting multiple times, order relations between presentations (e.g. two presentations that have to follow each other immediately), directly taking into account room capacities, and possibly even weighted preferences by participants (now a person with a high number of preferences had a much bigger influence than a person with a small number of preferences).

**Acknowledgements** We would like to thank the organizers of the MAPSP 2015 conference to give us the opportunity to schedule their conference. We would also like to thank the participants for their cooperation in taking the time to communicate their preferences.

## References

Edis E, Sancar Edis R (2013) An integer programming model for the conference timetabling problem. CBU Journal of Science 9(2):55–62

Eglese R, Rand G (1987) Conference seminar timetabling. Journal of the Operational Research Society 38(7):591–598

Gulati M, Sengupta A (2004) TRACS: Tractable Conference Scheduling. In: Proceedings of the Decision Sciences Institute Annual Meeting (DSI 2004), pp 3161–3166

Ibrahim H, Ramli R, Hassan M (2008) Combinatorial design for a conference: constructing a balanced three-parallel session schedule. Journal of discrete mathematical sciences & cryptography 11(3):305–317

Le Page Y (1996) Optimized schedule for large crystallography meetings. Journal of Applied Crystallography 29(3):291–295

Lovász L, Plummer MD (1986) Matching Theory, Annals of Discrete Mathematics, vol 29. AMS Chelsea Publishing

Martin A (2005) A personalised conference programme advisor. OR Insight 18(2):22–30

Nicholls M (2007) A small-to-medium-sized conference scheduling heuristic incorporating presenter and limited attendee preferences. Journal of the Operational Research Society 58(3):301–308

Potthoff R, Brams S (2007) Scheduling of panels by integer programming: Results for the 2005 and 2006 New Orleans meetings. Public Choice 131(2):465–468

Potthoff R, Munger M (2003) Use of integer programming to optimize the scheduling of panels at annual meetings of the Public Choice Society. Public Choice 117(1):163–175

Quesnelle J, Steffy D (2015) Scheduling a conference to minimize attendee preference conflicts. In: Hanzálek Z, Kendall G, McCollum B, Šucha P (eds) Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA), pp 379–392

van Rooij JMM, van Kooten Niekerk ME, Bodlaender HL (2012) Partition Into Triangles on Bounded Degree Graphs. Theory of Computing Systems 52(4):687–718

Sampson S (2004) Practical Implications of Preference-Based Conference Scheduling. Production and Operations Management 13(3):205–215

Sampson S, Weiss E (1995) Increasing service levels in conference and educational scheduling: a heuristic approach. Management Science 41(11):1816–1825

Sampson S, Weiss E (1996) Designing Conferences to Improve Resource Utilization and Participant Satisfaction. Journal of the Operational Research Society 47(2):297–314

Tanaka M, Mori Y (2002) A Hybrid Grouping Genetic Algorithm for Timetabling of Conference Programs. In: De Causmaecker P, Burke E (eds) Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002), pp 421–440

Tanaka M, Mori Y, Bargiela A (2002) Granulation of Keywords into Sessions for Timetabling Conferences. In: Proceedings of Soft Computing and Intelligent Systems (SCIS 2002), pp 1–5

Thompson G (2002) Improving Conferences through Session Scheduling. Cornell Hotel and Restaurant Administration Quarterly 43(3):71–76

Zulkipli F, Ibrahim H, Benjamin A (2013) Optimization Capacity Planning Problem on Conference Scheduling. In: Proceedings of the Business Engineering and Industrial Applications Colloquium (BEIAC 2013), pp 911–915