MISTA 2015

Scheduling a conference to minimize attendee preference conflicts

Jeffrey Quesnelle · **Daniel Steffy**

Abstract This paper describes a conference scheduling (or timetabling) problem where, at the time of registration, participants indicate preferences for events within the conference that they would like to attend. Based upon these preferences, an assignment of events to rooms and time slots should be determined that minimizes the number of attendee preference conflicts and satisfies a number of hard constraints. Ideally the schedule should be constructed so that for most, or all, participants the events that they would like to attend are assigned to different time slots. We show that our problem, and several variants of it, are NP-hard. An integer programming model is developed to solve the problem and a computational study of this model is performed on instances generated from real data. Improvements to the model, including a symmetry breaking reformulation and a dualization of some hard constraints, are shown to significantly improve solution times, making the problem tractable for the desired real world application.

1 Introduction

This project was motivated by the problem of scheduling events within PenguiCon [5], a conference organized by the open-source community in Michigan. The conference typically includes approximately 250 events such as lectures, demonstrations and panel discussions, all of which must be scheduled into rooms and time slots. Many of the events involve multiple presenters/panelists, and many presenters participate in more than one event; it is a hard constraint that no speaker can be multi-booked during a given time period. Furthermore, the registration website will give conference participants the ability to indicate preferences for events before the schedule is generated, giving the extra complication of trying to generate the schedule based on these responses that minimizes participant schedule conflicts. Our problem is related to previously studied conference and class scheduling problems but

Jeffrey Quesnelle
University of Michigan-Dearborn
E-mail: jfquesne@umich.edu
This paper was written while the author was an undergraduate at Oakland University

Daniel Steffy
Oakland University
E-mail: steffy@oakland.edu

includes what we believe is a novel and difficult combination of constraints and objectives. The goal of this paper is to model our problem, relate it to previous work, and implement solution methods that can be deployed to schedule the PenguiCon conference in practice.

In Section 2 we discuss some related work. In Section 3 we formally describe our problem and relate it to previous work, we also show that our scheduling problem, and some variants, are all NP-hard. Section 4 describes an integer programming model for the problem. Various improvements to the model are also described and evaluated computationally. Section 5 provides concluding remarks.

2 Background and related work

One of the oldest scheduling problems that has been studied is the *timetable design problem* (TTD). Given a set of time slots, a set of teachers and their available teaching hours, and a matrix describing which courses each teacher is required to teach, the TTD problem is the problem of determining if there exists a schedule that satisfies the constraints. TTD was shown to be NP-Complete in 1976 via reduction from 3-SAT [1]. However it is notable that certain variants of the TTD problem are known to be polynomial time solvable. For example, if each teacher is only available for up to two hours, or each teacher is able to teach any class, then the problem is solvable in polynomial time [2].

The basic TTD model often doesn't map well onto several common problems such as scheduling courses for a university. Specifically, the requirement that a teacher *must* teach certain classes may be relaxed to describing those classes they are *willing* to teach. This is know as the Basic Course Scheduling problem (BCS); it was shown to be solvable in polynomial time by Lovelace [3]. Extensions of the BCS, for example including the requirement that courses are assigned to rooms, results again in an NP-hard problem.

The scheduling problem considered in this paper more closely resembles the TTD problem, before introducing it we will give a precise formulation of the TTD. Here we denote the decision variables as a function f, which gives the assignment of presenters to talks and hours. We henceforth refer to the courses, or events as *talks*. We also assume that all talks have the same length and introduce a set of *hours* which is used to represent the set of time slots in which talks can be scheduled.

TIMETABLE DECISION PROBLEM
INSTANCE:

1. a finite set H of hours and numbers n and m indicating the number of presenters and talks, respectively;
2. a collection $P = \{P_1, P_2, \cdots, P_n\}$, where $P_i \subseteq H$ (there are n presenters and P_i is the set of hours during which the ith presenter is available for presenting);
3. a collection $T = \{T_1, T_2, \cdots, T_m\}$, where $T_j \subseteq H$ (there are m talks and T_j is the set of hours during which the jth talk can be given);
4. an $n \times m$ matrix G of nonnegative integers (G_{ij} is the number of hours (times) which the ith presenter will give the jth talk).

QUESTION: Does there exist a function

$$f(i,j,h) : \{1,\cdots,n\} \times \{1,\cdots,m\} \times H \to \{0,1\}$$

(where $f(i,j,h) = 1$ if and only if presenter i gives talk j during hour h) such that
(a) $f(i,j,h) = 1 \Rightarrow h \in P_i \cap T_j$ (the presenter and talk are both available to be scheduled at hour h);

(b) $\sum_{h \in H} f(i,j,h) = G_{ij}$ for all $1 \leq i \leq n$ and $1 \leq j \leq m$ (the ith presenter was scheduled for the jth talk the required number of times);
(c) $\sum_{i=1}^{n} f(i,j,h) \leq 1$ for all $1 \leq j \leq m$ and $h \in H$ (no talk has more than one presenter at a time);
(d) $\sum_{j=1}^{m} f(i,j,h) \leq 1$ for all $1 \leq i \leq n$ and $h \in H$ (no presenter is giving more than one talk simultaneously).

3 Conference scheduling problem

We now consider extensions and modifications of the TTD problem that incorporate requirements arising from our application.

3.1 Conference TTD Problem

Although the TTD problem is related to our target problem, it does not capture all of the decisions and constraints involved. One requirement is that some talks may involve multiple presenters, each of which may have additional differing scheduling conflicts. In the TTD model, constraint (c) ensures that each talk is scheduled to exactly one presenter. In the case where multiple presenters are allowed we most likely wish to add a different constraint: that for each talk *every* presenter that can be scheduled is scheduled. For example, if Alice is giving talks A, B, and C, and Bob is giving talks B, C, and D, all scheduled instances of B and C should include both Alice and Bob. We call this variant the Conference Timetable Decision problem (CTTD).

CONFERENCE TIMETABLE DECISION PROBLEM
Same as TTD, but with constraint (c) changed to
(c) $(G_{ij} > 0) \wedge (G_{i'j} > 0) \Rightarrow f(i,j,h) = f(i',j,h)$ for all $1 \leq i,i' \leq n$, $1 \leq j \leq m$ and $h \in H$ (all presenters that are required to give a talk must be present at all instances of that talk);

We now show that CTTD is NP-complete via reduction from the Graph k-colorability problem, the decision problem of determining whether or not a graph admits a k-coloring, which is well known to be NP-complete [4]. As we will see in the proof, the CTTD problem is NP-complete even without the inclusion of the availability constraints and even if each entry G_{ij} is equal to zero or one.

Proposition 1 *CTTD is NP-Complete.*

Proof We first note that CTTD is clearly in NP. Given a graph $G = (V,E)$ and a positive integer k, we will show how to construct an instance of CTTD that is feasible if and only if G is k-colorable. For simplicity of presentation we assume that G contains no isolated nodes (coloring of such nodes is trivial). Essentially, a talk is created to correspond to each vertex in G, each of the k colors corresponds to an hour in which talks can be scheduled and speakers are created to correspond to the edges in G. More formally, define $H = \{1,2,\cdots,k\}$ and for each vertex $v_j \in V = \{v_1,v_2,\cdots,v_{|V|}\}$ let $T_j = H$. For each edge $e_l \in E = \{e_1,e_2,\cdots,e_{|E|}\}$ we let $P_l = H$. For each $e_l = (v_i,v_j) \in E$, where $v_i,v_j \in V$, we let $\hat{G}_{li} = \hat{G}_{lj} = 1$, and let $\hat{G}_{lm} =$

0 for each $m \neq i, j$. We now have an instance $(H, P, T, \hat{G})$ of CTTD (whose construction was easily computed in polynomial time).

We now observe that $(H, P, T, \hat{G})$ has a feasible schedule f if and only if G is k-colorable. Given a feasible schedule f, each vertex v_a in G is assigned color h, where h is the timeslot in which talk a is assigned. For any edge $e_l = (v_i, v_j) \in E$ we note that since a speaker l was created to give talks i and j they will not be scheduled in the same time slots, and thus v_i, v_j are assigned different colors, leading to a valid k-coloring of G. Conversely, given a k-coloring of G it is easy to construct a feasible schedule f for $(H, P, T, \hat{G})$ using the same idea. Finally we conclude that CTTD is NP-Complete.

3.2 Basic Conference TTD Problem

Lovelace showed that a relaxed version of TTD (called BCS for "Basic Course Scheduling") can be solved in polynomial time using a network flow model [3]. The principal differences between BCS and TTD are the reduction of many "hard" requirements such as those insisting that presenters give *exactly* a certain number of talks of a certain type to simply saying they may give *at most* the number of talks for which they are *willing* to give. BCS does maintain a hard requirement that all presentations must be scheduled, but offers flexibility in which speaker makes each presentation. We give a formulation of BCS using notation consistent with our description of the Basic Timetable Decision Problem (BTTD).

BASIC TIMETABLE DECISION PROBLEM
INSTANCE:

1. a finite set H of hours and numbers n and m indicating the number of presenters and talks, respectively;
2. a collection $P = \{P_1, P_2, \cdots, P_n\}$, where $P_i \subseteq H$ (there are n presenters and P_i is the set of hours during which the ith presenter is available for presenting);
3. a collection $T = \{T_1, T_2, \cdots, T_m\}$, where $T_j \subseteq H$ (there are m talks and T_j is the set of hours during which the jth talk can be given);
4. a function $L : \mathbb{Z}^+ \to \mathbb{Z}_0^+$, where $L(n)$ is the maximum number of talks that the nth presenter can give;
5. a function $S : \mathbb{Z}^+ \to \mathbb{Z}_0^+$, where $S(m)$ is the desired number of instances of the mth presentation;
6. a function $WTP : \{1, 2, \cdots, n\} \times \{1, 2, \cdots, m\} \to \{0, 1\}$, where $WTP(i, j)$ indicates if the ith presenter is Willing To Present the jth talk.

QUESTION: Does there exist a function

$$f(i, j, h) : \{1, \cdots, n\} \times \{1, \cdots, m\} \times H \to \{0, 1\}$$

(where $f(i, j, h) = 1$ if and only if presenter i gives talk j during hour h) such that

(a) $f(i, j, h) = 1 \Rightarrow h \in P_i \cap T_j$ (the presenter and talk are both available to be scheduled at hour h);

(b) $\sum_{h \in H} f'(j, h) = S(j)$ for all $1 \leq j \leq m$ where $f'(j, h) = 1 \iff \exists i$ with $1 \leq i \leq n$ such that $f(i, j, h) = 1$, and 0 otherwise (the jth talk is given the required number of times);

(c) $\sum_{j=1}^{m} f(i, j, h) \leq 1$ for all $1 \leq i \leq n$ and $h \in H$ (there is no more than one presenter scheduled for each instance of a talk);

(d) $f(i,j,h) = 1 \Rightarrow WTP(i,j) = 1$ (only presenters willing to give a talk are scheduled for it);
(e) $\sum_{j=1}^{n} \sum_{h \in H} f(i,j,h) \leq L(i)$ for all $1 \leq i \leq n$ (the total number of talks that the ith presenter is scheduled for is at most their maximum number of presentations)
(f) $\sum_{j=1}^{m} f(i,j,h) \leq 1$ for all $1 \leq i \leq n$ and $h \in H$ (no presenter is giving more than one talk simultaneously).

Remark 1 BTTD $\in$ P [3].

The difference between TTD and CTTD is the ability for presentations to have multiple presenters, and the requirement that all presenters be scheduled for all instances of a talk. Likewise, we can formulate a modified version of BTTD that incorporates this new constraint which we will call the Basic Conference Timetable Decision problem (BCTTD).

BASIC CONFERENCE TIMETABLE DECISION PROBLEM
Same as BTTD, but with constraint (c) changed to
(c) $WTP(i,j) = WTP(i',j) \Rightarrow f(i,j,h) = f(i',j,h)$ for all $1 \leq i,i' \leq n$, $1 \leq j \leq m$ and $h \in H$ (all presenters that are required to give a talk must be present at all instances of that talk);

We observe that after this constraint is introduced, we may apply the same reduction used in Proposition 1 and thus we have the following.

Proposition 2 *BCTTD is NP-Complete.*

3.3 Extended Conference TTD Problem

We now present a modification to CTTD that introduces room assignment decisions and room compatibility constraints. The CTTD problem assigns speakers to talks and time slots but, as in many other applications, we also require that talks are assigned to suitable rooms. Furthermore, rooms also may only be available during certain times or suitable for certain talks and this information must be factored into the problem. This leads us to the Extended Conference Timetable Decision problem (ECTTD).

EXTENDED CONFERENCE TIMETABLE DECISION PROBLEM
INSTANCE: Same as CTTD, but with the additional parameters:
5. a finite set R of rooms;
6. a collection $\{A_1, A_2, \cdots, A_r\}$, where $A_k \subseteq H$ (there are $r = |R|$ rooms and A_k is the set of hours during which the kth room is available);
7. a collection $\{S_1, S_2, \cdots, S_m\}$, where $S_l \subseteq R$ (there are m talks and S_l is the set of rooms that the lth presentation may be given in)

QUESTION: Does there exist a function

$$f(i,j,h,r) : \{1,\cdots,n\} \times \{1,\cdots,m\} \times H \times R \rightarrow \{0,1\}$$

(where $f(i,j,h,r) = 1$ if and only if presenter i gives talk j during hour h in room r) such that
(a) $f(i,j,h,r) = 1 \Rightarrow h \in P_i \cap T_j \cap A_r \wedge r \in S_j$ (the ith presenter, jth presentation and room r are all available to be scheduled at hour h and room r is suitable for the jth presentation);

(b) $\sum_{r\in R}\sum_{h\in H} f(i,j,h,r) = G_{ij}$ for all $1 \le i \le n$ and $1 \le j \le m$ (the ith presenter was scheduled for the jth presentation the required number of times);

(c) $G_{ij} > 0 \wedge G_{i'j} > 0 \Rightarrow f(i,j,h,r) = f(i',j,h,r)$ for all $1 \le i,i' \le n$, $1 \le j \le m$, $h \in H$, and $r \in R$ (all presenters that are required to give a talk must be present at all instances of that talk);

(d) $\sum_{r\in R}\sum_{j=1}^{m} f(i,j,h,r) \le 1$ for all $1 \le i \le n$ and $h \in H$ (no presenter is giving more than one talk simultaneously);

(e) $\sum_{j=1}^{m} f'(j,h,r) \le 1$ for each $h \in H$ and $r \in R$ where $f'(j,h,r) = 1 \iff \exists i$ with $1 \le i \le n$ such that $f(i,j,h,r) = 1$, and 0 otherwise (room r is scheduled for at most one talk at hour h).

We also note that since this is a clear generalization of CTTD, and in the class NP, it is also NP-Complete.

Proposition 3 *ECTTD is NP-Complete.*

3.4 Preference Conference Optimization Problem

We have examined several different variations of scheduling problems as they relate to conferences; we now offer a final variation that will be the subject of study for the rest of the paper. In particular we are interested in not only finding a schedule that is feasible with respect to speaker and room logistics, but one that also minimizes attendee preference conflicts. Formally, an *attendee preference conflict* is a tuple (e, j, j') where e is an attendee, j, j' are two events for which e has indicated an interest to attend, and j, j' are scheduled to occur during the same time slot. Namely, given the set of conference attendees and their preferences for talks they would like to attend, we want to minimize the total number of times that a given attendee has shown preference for a pair of talks that are scheduled in the same time period. We call the resulting optimization problem the Preference Conference Optimization problem (PCO).

PREFERENCE CONFERENCE OPTIMIZATION PROBLEM
INSTANCE: Same as ECTTD, but with the additional parameters:

8. a finite set $E = \{e_1, e_2, \cdots, e_t\}$ of attendees;
9. a $t \times m$ 0-1 matrix W (W_{ej} indicates if the eth attendee would like to attend the jth talk).

GOAL: Find a function

$$f(i,j,h,r) : \{1,\cdots,n\} \times \{1,\cdots,m\} \times H \times R \to \{0,1\}$$

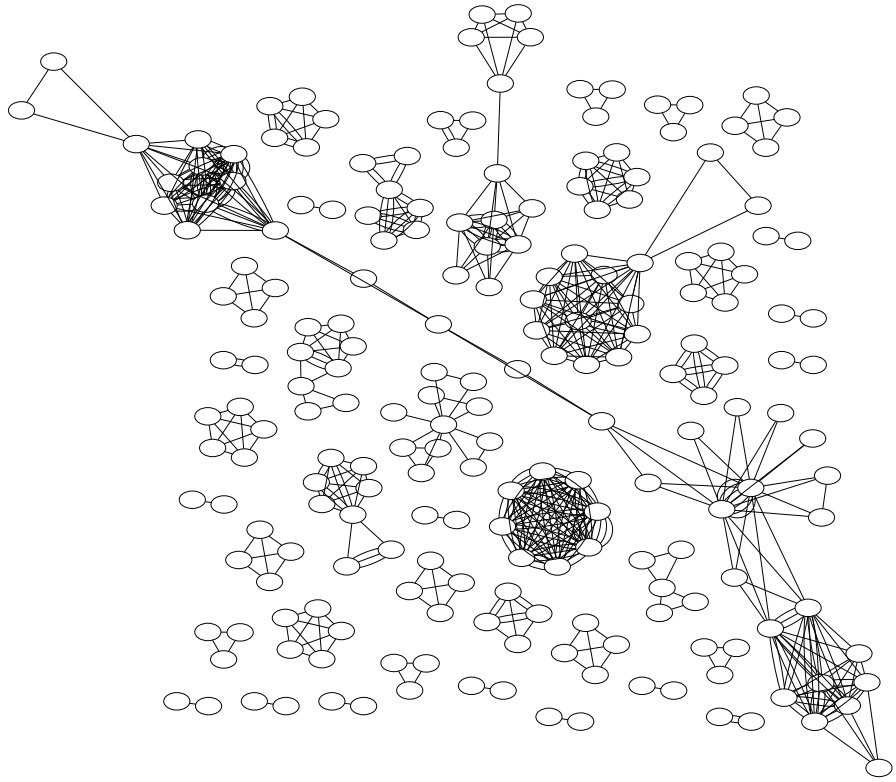
that satisfies all the constraints of the ECTTD problem while minimizing the sum of the attendee preference conflicts where, as described above, an attendee preference conflict is any tuple (e, j, j') such that there exist i, i', h, r and r' such that $f(i,j,h,r) = f(i',j',h,r') = 1$ where $W_{ej} = 1$ and $W_{ej'} = 1$.

4 Integer programming models

We will present integer programming models that can solve PCO and ECTTD. The data used to measure the models comes from a real conference held in 2013, which we shall

refer to as PC2013. PC2013 had 195 presenters giving a total of 253 talks. Figure 1 helps illustrate the data we worked with: each vertex represents a talk and adjacent vertices share a common presenter and cannot be scheduled at the same time. By Proposition 1, solving CTTD is equivalent to asking if this graph admits an h-coloring (where h is the number of time slots available at the conference).

Fig. 1 Presenter conflicts that must be scheduled around in PC2013



4.1 Model for the Extended Conference Timetable Decision Problem

Before describing the model for the PCO problem, we give an integer programming model where feasible solutions determine values of functions f which correspond directly to schedules that satisfy the constraints laid out in the ECTTD decision problem. We note that the size of f, i.e. the number of variables in our model, can be very large when building models corresponding to our application data.

$$\text{size of } f = \#\text{ of presenters } \times \#\text{ of talks } \times \#\text{ of hours } \times \#\text{ of rooms}$$

For PC2013, size of $f = 193 \times 253 \times 37 \times 15 = 27{,}421{,}440$. This number of variables could be problematic computationally, however, we know that many (nearly all) of these variables will be zero based on information we have at formulation time. For example, if a presenter

i doesn't give talk j, then $f(i,j,h,r) = 0$ for all $h \in H, r \in R$. Although integer programming solver may automatically fix such variables to zero in the preprocessing phase, we exclude these variables from the model at the time of construction. We create an index set $\mathscr{F} \subseteq \{1,\cdots,n\} \times \{1,\cdots,m\} \times H \times R$ where $f_{i,j,h,r} \in \mathscr{F}$ only if presenter i gives talk j; the talk j, presenter i and room r are available at hour h; and room r is suitable for the jth talk. In addition to this condition, we restrict the inclusion of variable indices in $\mathscr{F}$ to the intersection of the available hours of all *co-presenters* (pairs of presenters that give the same talk), e.g. if co-presenters i, i' have availability sets $\{h_2, h_3\}$ and $\{h_3, h_4\}$ (assuming room and talk availability is at least $\{h_2, h_3, h_4\}$) then only variables with $h = h_3$ for these co-presenters and talk will be included. In practice, the reduction of our solution space to only $\mathscr{F}$ gives a massive performance gain. For PC2013, this immediately reduced the number of variables down to 91,514 (a reduction of 99.997%).

The variables indexed by $\mathscr{F}$ can be thought of as a sparse representation of the interesting elements of the domain of f. In addition to $\mathscr{F}$ we will use the index set $\mathscr{G}$ to represent tuples (j,h,r) for which talk j can be given by any presenter at hour h in room r, and variables $g_{j,h,r}$ will indicate whether or not this occurs. We will now describe a formulation that implements each of the constraints on f in ECTTD.

ECTTD formulation

$$\text{minimize: } 0 \tag{1}$$

$$\text{subject to:} \tag{2}$$

$$\sum_{h,r:(i,j,h,r)\in\mathscr{F}} f_{i,j,h,r} = G_{ij} \qquad \text{for every presenter } i \text{ and talk } j \tag{3}$$

$$f_{i,j,h,r} - f_{i',j,h,r} = 0 \qquad \text{for every talk } j \text{ with co-presenters } i,\ i' \tag{4}$$

$$\sum_{j,r:(i,j,h,r)\in\mathscr{F}} f_{i,j,h,r} \leq 1 \qquad \text{for every presenter } i \text{ and hour } h \tag{5}$$

$$\left(\sum_{i:(i,j,h,r)\in\mathscr{F}} f_{i,j,h,r}\right) - U \times g_{j,h,r} \leq 0 \qquad \text{for each } g_{j,h,r} \in \mathscr{G} \tag{6}$$

$$\sum_{j:(j,h,r)\in\mathscr{G}} g_{j,h,r} \leq 1 \qquad \text{for each hour } h \text{ and room } r \tag{7}$$

$$\text{binary: } f_{i,j,h,r}, g_{j,h,r} \tag{8}$$

The first requirement (a) of ECTTD merely enforces all availability and suitability sets. We implicitly enforce this in our model by considering only the variables indexed over $\mathscr{F}$. As such, no specific constraints are needed in our model.

The second requirement (b) ensures that every presenter is scheduled for all of their talks, which we receive as parameter G to ECTTD where G_{ij} is the number of times presenter i should give talk j. For each presenter i and talk j, the sum of the times they are scheduled (over all hours and rooms) should be G_{ij}; this is constraint (3).

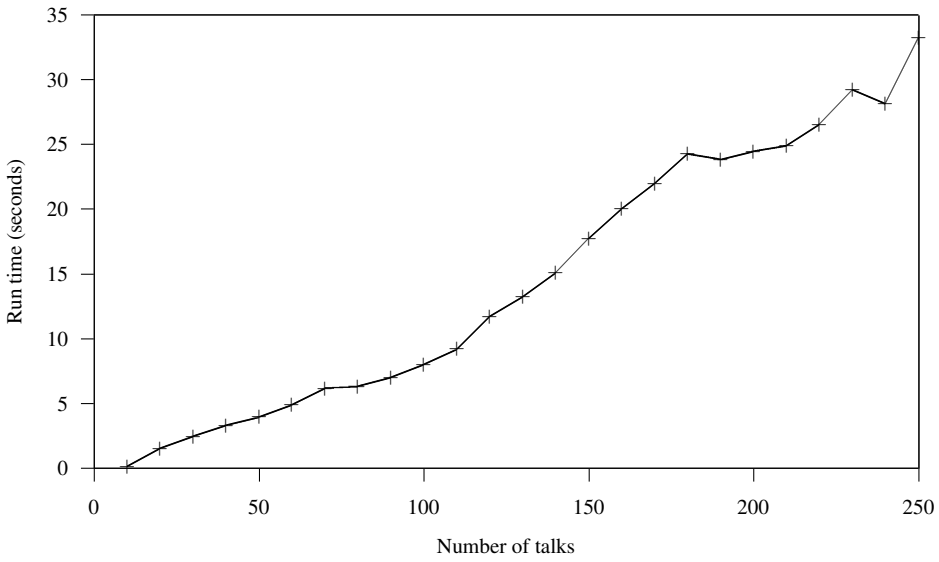
To ensure requirement (c) we force that all co-presenters have the same schedule for their shared talk (constraint (4)).

Requirements (a)-(c) guarantee that all presenters are scheduled for their talks and that co-presenters are scheduled together. Requirement (d) ensures that if a presenter has multiple talks, then these talks are scheduled during different hours. For each presenter i and hour h, the sum of their schedule variables for their talks in all rooms must be less than or equal to one (constraint (5)).

The final requirement (e) ensures that room scheduling is exclusive. We would like to simply iterate over $\mathscr{F}$ for a particular pair of hour h and room r, summing all of these together. If we didn't allow co-presenters (like TTD) then we could simply make this sum less than or equal to one. But, for talks with co-presenters this sum varies. To overcome this we create indicator variables $g_{j,h,r}$ where $g_{j,h,r} = 1$ whenever talk j is scheduled at hour h in room r; this is modeled in constraint (6), where U represents a sufficiently large number. Finally, constraint (6) ensures that no room is multi-booked by checking the sum of $g_{j,h,r}$ for each pair h, r.

Solving this feasibility problem proved tractable. We solved this formulation with the open source IP solver CBC on PC2013 with varying numbers of talks pruned out to see how the model scales. The results are given in Figure 2.

Fig. 2 Run time of ECTTD model



4.2 Model for the Preference Conference Optimization Problem

We now turn our attention to the Preference Conference Optimization (PCO) problem. PCO adds an additional layer of complexity to ECTTD by including a matrix of preferences for attendees with the goal of minimizing the number of conflicts caused by concurrent talks. Through experimentation we have found that considering these preferences significantly increases the difficulty of solving our conference scheduling problem. We now present an integer programming model for PCO.

The PCO model we present builds on our previous ECTTD model. In addition to the G variables which collapse the four dimensional f function down to three dimensions (talk $\times$ hour $\times$ room), all the while considering only those variables for which a feasible schedule is even possible given the different availability constraints, we will introduce three new classes

of variables for PCO. The first is z which will collapse g to two dimensions (talk $\times$ hour). Next, we will expand z to c which will have a three dimensional range from talk $\times$ talk $\times$ hour; it will indicate if two talks j, j' are given concurrently at hour h. As with previous models, the corresponding script (e.g. $\mathscr{Z}$ for z) will represent the index set of variables that are possible given availability constraints. The parameter w is constructed from W as a talk $\times$ talk $\times$ hour matrix where each entry is the number of attendees who wish to attend both talks j, j' for all h when j, j' can be given based on talk, presenter, and room availability. Formally, $w_{j,j',h} = |\{e_k \in E : e_k \text{ has } W_{kj} = 1 \text{ and } W_{kj'} = 1\}|$ for each h where $(h,j) \cup (h,j') \subseteq \mathscr{Z}$. The objective function that is minimized in the model is the sum of elements in c.

PCO formulation

$$\text{minimize:} \sum_{(j,j',h)\in\mathscr{C}} w_{j,j',h} \times c_{j,j',h} \tag{9}$$

subject to: (10)

$$\sum_{h,r:(i,j,h,r)\in\mathscr{F}} f_{i,j,h,r} = G_{ij} \qquad \text{for every presenter } i \text{ and talk } j \tag{11}$$

$$f_{i,j,h,r} - f_{i',j,h,r} = 0 \qquad \text{for every talk } j \text{ with co-presenters } i,\ i' \tag{12}$$

$$\sum_{j,r:(i,j,h,r)\in\mathscr{F}} f_{i,j,h,r} \leq 1 \qquad \text{for every presenter } i \text{ and hour } h \tag{13}$$

$$\left(\sum_{i:(i,j,h,r)\in\mathscr{F}} f_{i,j,h,r}\right) - U \times g_{j,h,r} \leq 0 \qquad \text{for each } (j,h,r) \in \mathscr{G} \tag{14}$$

$$\sum_{j:(j,h,r)\in\mathscr{G}} g_{j,h,r} \leq 1 \qquad \text{for each hour } h \text{ and room } r \tag{15}$$

$$\left(\sum_{j,h:(j,h,r)\in\mathscr{G}} g_{j,h,r}\right) - U \times z_{j,h} \leq 0 \qquad \text{for each room } r \tag{16}$$

$$\sum_{h:(j,h)\in\mathscr{Z}} z_{j,h} = G_{ij} \qquad \text{for each talk } j \text{ and some presenter } i \tag{17}$$

$$z_{j,h} + z_{j',h} - c_{j,j',h} \leq 1 \qquad \text{for each } (j,j',h) \in \mathscr{C} \tag{18}$$

$$\text{binary: } f_{i,j,h,r},\ g_{j,h,r},\ z_{j,h},\ c_{j,j',h} \tag{19}$$

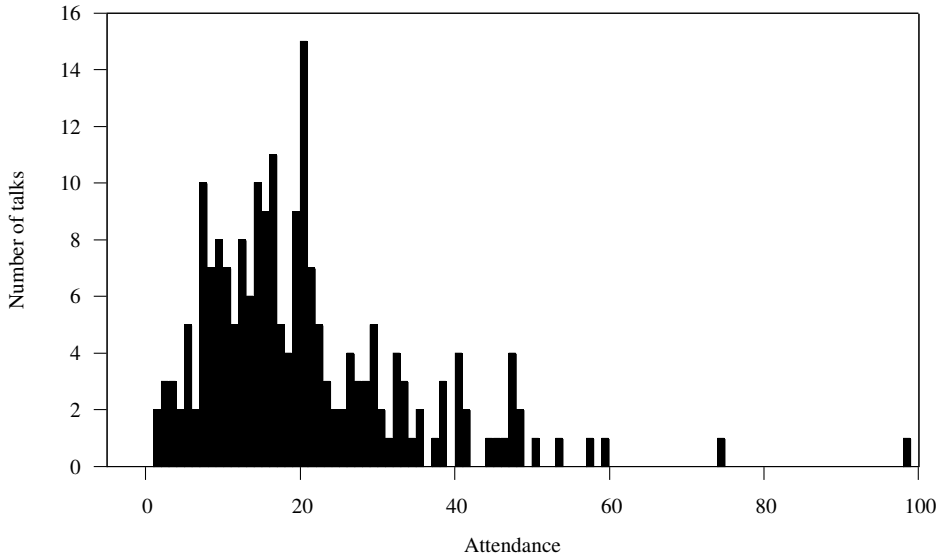
Constraints (11) - (15) are the same as in our model for ECTTD. Constraint (16) begins to build the z variables which will be 0-1 indicators of talk j being given at hour h via the same boolean cast mechanism described previously by collapsing the room entries for j,h in g. To ensure that only the correct number of zs are set to one, constraint (17) sums all hours h for each talk j and sets it equal to the number of times that talk j was set to be given in the problem instance (the matrix G). It is of note that we pick *any* presenter i's entry in G for talk j; although it is possible that a co-presenter i' may have a different value for $G_{i'j}$ requirement (c) of PCO explicitly forbids this since it would be impossible for all co-presenters to be at all instances of a talk if they had different entries for their shared talk j, thus we can pick any presenter i.

The z variables will now be used to generate c, which indicates if a pair of talks j, j' are being given concurrently at hour h. Specifically, constraint (18) , enforces that $z_{j,h} = 1$ and $z_{j',h} = 1 \implies c_{j,j',h} = 1$. The objective (9) of the model is to minimize the sum of attendee preference conflicts. Each variable $c_{j,j',h}$ appears with coefficient of $w_{j,j',h}$, which is the number of attendee preference conflicts generated by talks j and j' being scheduled in the

same time period. Note that if the coefficient $w_{j,j',h}$ is nonzero, then the minimization nature of the problem will force the corresponding $c_{j,j',h}$ to take zero value whenever possible.

To measure the efficiency of our model we tested it on our sample set PC2013. This data set included information about talks, speakers and attendance, but did not include attendee preferences (as they were not solicited that year), however we may use this data to generate reasonable instances by taking the historical attendance data as a basis for generating hypothetical attendee preferences. Attendance figures we recorded for each talk given at PC2013; a distribution of the attendance per talk is shown in Figure 3. The sum of all atten-

Fig. 3 Distribution of attendance at PC2013



dance counts was 4101 [6] for around 1000 unique attendees. For the purposes of testing our model created a W such that $\sum_{i=0}^{n} W_{ij}$ was equal to the attendance count for that talk j, i.e. we created an indicated attendance preference for each individual talk attendance at PC2013. Since individual attendee attendance wasn't tracked (only totals were) we took some liberties in distributing the preference responses across the attendees in our model. We first randomly spread the preference responses over the attendees using a uniform distribution; that is, if talk j had an attendance of 24 in PC2013 then 24 attendees were randomly chosen to express a preference for attending this talk. We solved our model with commercial solver Gurobi which returned a solution with an objective value of 0 after 64 seconds, i.e. a schedule with absolutely no attendee conflicts. All computations were run on a machine with 4 12-core Intel Xeon E5-2695 CPUs running at 2.4 GHz with 96 GB of RAM

After finding a non-conflicting schedule for uniformly distributed random attendees that followed the attendance counts in PC2013 we turned our attention to how the solver would react when the random attendees were not distributed uniformly. Our intuition was that, like the actual attendance figures, the distribution of attendance per attendee would not be evenly spaced out; there would be some attendees who went to many talks, and some who went to

only a few. We chose a normal distribution with $\mu = 500, \sigma = 100$. For each attendance in the PC2013 distribution (Figure 3) a random integer from the normal distribution was chosen. Since $\mu = 500$ those attendees with index around 500 were much more likely to be chosen to indicate a preference to the talk then those with indices 50 or 950, which implemented our intuition that a small percentage of attendees will indicate attendance preferences for many talks while the bulk of those remaining will indicate preferences for relatively few. In addition, we added a check to ensure that no attendee was selected to attend more talks than were hours available; such a scenario would automatically preclude a zero objective value. Even with an extremely powerful computer to run our model on and a state-of-the-art commercial solver we were unable to solve this instance after 24 hours. To help understand this phenomenon we created a half-sized problem (half as many talks and hours) and ran the model with decreasing values of σ and found that the solving time exploded exponentially as σ decreased; the average running time for $\sigma = 200$ was 22 seconds but increased to 22,683 seconds for $\sigma = 100$.

4.2.1 Performance considerations

After including attendee preferences in our test models, the integer programming models became significantly more difficult to solve. One possible cause of this is symmetry present in the models, a property that often leads to increased solution time [7]. An integer programming model is said to be *symmetric* if some of its variables can be permuted (nontrivially) without changing the structure of the problem [8]. Our model exhibits high amounts of symmetry in relation to the scheduling of talks in rooms. If two rooms have the same availability and suitability set then permuting talk assignments among them in each hour produces no discernible change to the objective. It may be, however, that the solver will choose to branch early on in its branch-and-bound tree on these room assignments, leading to lots of unnecessary computation. In general it is difficult for the solver to detect that such variables "really" represent the same thing, although there are several mechanisms for determining and avoiding symmetry in solvers [9]. However, it is easy for us to identify this symmetry and avoid it.

Two rooms will be said to be *symmetric* if they have the same suitability and availability sets, i.e. for rooms R_α and R_β we have that R_α and R_β are symmetric if and only if

$$A_\alpha = A_\beta \quad \text{and}$$
$$\{i \mid R_\alpha \in S_i \text{ for all } 1 \leq i \leq m\} = \{i \mid R_\beta \in S_i \text{ for all } 1 \leq i \leq m\}.$$

To break the symmetry we create room *classes* which will represent several rooms with the same attributes. First, we make a new room set $R' = \{r' = \{r_1, r_2, \cdots, r_p\} \subseteq R \mid \text{all } r \in r' \text{ are symmetric with each other}\}$. The corresponding new availability set A' has simply the common availability set for each new $r' \in R'$. For the new suitability set S' we replace each instance of r with the room class r' that r is a member of. When we solve our model talks will be booked to room classes, avoiding the symmetry that arises by having to consider two essentially "equal" rooms separately. When our model is solved we will have bookings in room classes, and we can arbitrarily assign the talk to any room in that class. We must make only one adjustment to our model: Constraint (15) in the PCO model ensures that each room has only one talk booked in it per hour. For our room classes we wish to relax this, requiring only that the number of bookings be at most the number of rooms in the class; this way, when we assign actual rooms from the solved model we can match talks to rooms in a

one-to-one way. Formally, we will change constraint (15) to

$$\sum_{j \in \mathscr{G}_{h,r}} g_{j,h,r} \leq |r|. \tag{20}$$

It is easy to see that this model degenerates to our regular PCO model when no rooms are symmetric; in this case each room class would contain only one room. In practice the removal of the room symmetries increased performance by roughly a factor of 5x for our solver on PC2013, which contained only three room classes but had fourteen rooms (see Figure 4).

The next step we took to improve performance was to *dualize* one class of constraints. In this technique, these requirements are moved from being hard constraints in the model, to appearing in the objective function with a sufficiently large penalty to ensure their satisfaction. We chose to dualize constraint (d), namely that no presenter is scheduled for more than one talk per hour. Intuitively, this seemed like a promising adjustment because these constraints are similar in structure to the attendee preference conflicts that are minimized in the objective function. We first created 0-1 indicator variables $d_{i,h} = 1 \iff$ presenter i is doubly (or more) booked at hour h by changing (13) (which enforced (d)) to

$$\left(\sum_{j,r \in \mathscr{F}_{i,h}} f_{i,j,h,r} \right) - U \times d_{i,h} \leq 1. \tag{21}$$

Where U is a sufficiently large number. The left hand side of (21) is the number of times that presenter i is scheduled at hour h, and $d_{i,h}$ may be 0 or 1 if this sum is less than 2, but must be 1 if the sum is 2 or greater. We then changed the objective (9) to

$$\text{minimize: } \sum_{(j,j',h) \in \mathscr{C}} w_{j,j',h} \times c_{j,j',h} + \sum_{(i,h) \in \mathscr{D}} U \times d_{i,h}. \tag{22}$$

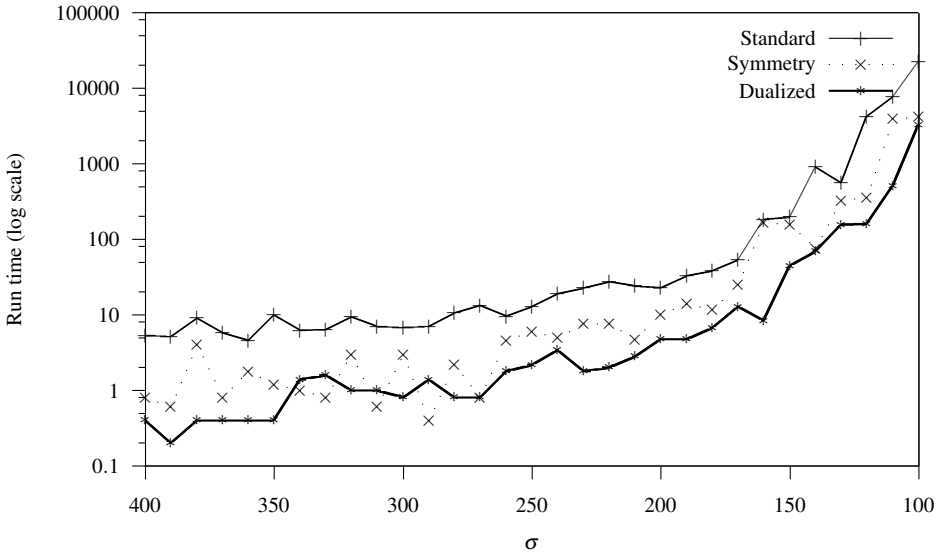
Our new objective places a penalty of U on presenters being multiply booked. We should choose U sufficiently large so d is identically zero, otherwise the model can be resolved with a larger value of U. In our experiments the dualized constraints were always satisfied after solving the model. In practice, dualizing PCO led to moderate performance increases of roughly 75% faster.

For a summary of solution times comparing the original model with the improved models discussed in this subsection see Figure 4. The three models compared are: the Standard model, which corresponds to the PCO formulation given by (9)-(15); the Symmetry model, which incorporates the symmetry breaking reformulation described above; and finally the Dualized model, which incorporates both the symmetry breaking reformulation, and the dualization of constraint (d) as described above. The graph plots the average running time for solving 10 randomly generated instances for values of σ between 100 and 400 with increments of 10. The table shows the same information, only listing times for instances where σ is a multiple of 50. All times are listed in seconds. We also note that it turned out that for the generated instances solved in these experiments, the optimal solutions had an objective value of zero.

5 Conclusion

Conference scheduling represents an important class of timetabling problems. This paper studies a conference scheduling problem where attendee preference conflicts are minimized,

Fig. 4 Run time of PCO model with decreasing σ



(σ)	Standard	Symmetry	Dualized
400	5.4	0.8	0.4
350	10.0	1.2	0.4
300	6.8	3.0	0.8
250	12.8	6.0	2.2
200	22.8	10.2	2.0
150	196.2	156.8	45.2
100	22683.7	4179.2	3282.6

subject to a collection of hard constraints. We have demonstrated integer programming to be an effective solution technique, especially after incorporating symmetry breaking and other improvements.

References

1. Even, S., A. Itai, and A. Shamir. 1976. On the complexity of timetable and multicommodity flow problems. SIAM Journal on Computing 5, (4) (12): 691-13
2. Garey, M., and D. Johnson. 1976. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W. H. Freeman
3. Lovelace, A. 2010. On the complexity of scheduling university classes. M.S. in Computer Science Thesis. California Polytechnic State University: U.S.A.
4. Garey, M., D. Johnson, and L. Stockmeyer. 1976. Some simplified NP-Complete graph problems. Theoretical Computer Science 1: 237-267
5. Penguicon Conference, `http://www.penguicon.org/`, Accessed: January, 2015.
6. Penguicon Programming Ops. `http://penguicon.info/doku.php/programmingops?s=attendance`. Accessed: January, 2015.
7. Sherali, H.D., and J.C. Smith. 2001. Improving Discrete Model Representations via Symmetry Considerations. Managements Science 47: 1396-1407.
8. Margot, F. 2009. Symmetry in Integer Linear Programming. 2010. 50 Years of Integer Programming 1958-2008, Chapter 16: 647-681. Springer.
9. Ostrowski, J. 2008. Symmetry in Integer Programming. Ph.D. Thesis. Lehigh University: U.S.A.