

ULRR

Scheduling research grant proposal evaluation meetings

Item Type	Meetings and Proceedings
Authors	Healy, Patrick
Citation	6th International Conference on the Practice and Theory of Automated Timetabling;pp. 432-437
Publisher	PATAT
Download date	2026-04-10 16:13:44
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/2435

Scheduling Research Grant Proposal Evaluation Meetings

Patrick Healy

CS Department, University of Limerick, Limerick, Ireland
`patrick.healy@ul.ie`

Abstract In many funding agencies a model is adopted whereby a fixed panel of evaluators evaluate the set of applications. This is then followed by a general meeting where each proposal is discussed by those evaluators assigned to it with a view to agreeing on a consensus score for that proposal. It is not uncommon for some experts to be unavailable for the entire duration of the meeting; constraints of this nature, and others complicate the search for a solution. We report on a system developed to ensure the smooth running of such meetings.

1 Background

The process of evaluating research grant proposals presents some interesting opportunities for operations research practitioners and researchers. The model we discuss here assumes that a fixed pool of evaluators exists and the set of grant proposals is distributed amongst them subject to the evaluators' stated abilities to evaluate each. (Although this step of the process is outside the current scope it is an interesting assignment problem with many side constraints. For example, some evaluators, such as vice-chairs, may be expected to take a reduced load of evaluations due to other duties; no proposal should have a majority of vice-chairs evaluating it; and, for each proposal, one evaluator should be appointed as proposal reporter amongst the – usually 3, although requests for larger financial sums necessitate more – evaluators assigned to it with this extra duty evenly allocated amongst all evaluators.)

In the present context our interest begins when a general panel¹ meeting brings all of the evaluators together. Each proposal is discussed face to face by the assigned evaluators for the purpose of agreeing a consensus position and category scores, from which the final evaluation report may be written; the consensus meeting runs for fixed length of time. Following the entry of all scores in a database a ranking list is generated which forms the basis for funding decisions by the grant agency. So that the entire panel of evaluators may agree to the ranking list it is desirable that all consensus meetings be completed as quickly as possible allowing time for the inevitable clean-up before the final ranking list acceptance process.

In its most restricted form the problem may be expressed as, given an assignment matrix of proposals to evaluators, where each proposal has been read by some subset of evaluators, generate a schedule of consensus meetings that uses

¹ A panel may be thought of as a general research area, *e.g.* computer science.

the fewest number of time periods where the meetings can be held. A maximum of T time periods exist.

The constraints that must be respected, then, are:

1. A consensus meeting can only take place during one time slot;
2. An expert can only be in one consensus meeting during a time slot;
3. If a consensus meeting for a proposal takes place then all of the experts assigned to it must be present;
4. No more than T time slots may be used

The goal is to minimize the number of time slots used. While the problem bears some resemblance to the teacher-class timetabling problem, the objective function differs, as well as some of the constraints. In the following section we describe our first approach to solve the problem by modelling it as an ILP.

2 An ILP Model

Given $E = e_{ij}$, the assignment matrix that indicates what experts have been assigned to read proposal j , we can view its transpose $E^T = P = p_{ij}$ as the matrix that indicates what proposals have been assigned to expert j .

We introduce binary variables x_{ij} that indicate that the consensus meeting for proposal i will take place in time slot j , and y_{ij} that indicate that expert i is in some meeting during time slot $j, 1 \leq j \leq T$.

Constraint 1 above can be implemented by

$$\sum_j x_{ij} = 1 \quad \forall i$$

Constraint 2 says that if an expert is assigned to a time slot then s/he must be evaluating exactly one proposal from their allocation and, conversely, if an expert is not assigned to a time slot then none of their allocation are being evaluated in this slot.

$$y_{ij} = \sum_k p_{ik}x_{kj} = \sum_k e_{ki}x_{kj} \quad \forall i, j$$

Constraint 3 can be interpreted as meaning “meeting for proposal i happens at time $j \Rightarrow$ all of the experts associated with this proposal are assigned to this slot”. Note that this relation is not \Leftrightarrow since the same three experts may be involved in a different proposal. Also, by virtue of constraint 1 an expert can only attend one meeting in a time slot. Its implementation is

$$w_i x_{ij} \leq \sum_k e_{ik} y_{kj} \quad \forall i, j$$

where $w_i = \sum_j p_{ij}$ is the number of evaluators assigned to proposal i .

The goal is to minimise the number of time slots used. To do this we ask “how many proposals were evaluated in time slot j ?” ($\sum_i x_{ij}$) and charge j for

each of these. This forces as many proposals as possible into “small” time slots for, if a proposal is evaluated in a later time slot j' , instead of j , the cost rises by an amount $j' - j$. The cost z is

$$z = \sum_j (j \sum_i y_{ij})$$

Thus the ILP model we solve is

$$\text{minimize } \sum_j (j \sum_i y_{ij}) \tag{1}$$

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

$$\sum_j x_{ij} = 1 \quad \forall i \tag{3}$$

$$y_{ij} = \sum_k e_{ki} x_{kj} \quad \forall i, j \tag{4}$$

$$w_i x_{ij} \leq \sum_k e_{ik} y_{kj} \quad \forall i, j \tag{5}$$

$$x_{ij}, y_{ij} \quad \text{binary}$$

For a problem instance involving 58 evaluators and 383 proposals using CPLEX 7.0 and running on a Pentium V, the previous model had not terminated after 3 days of running time.

Of equal concern was the inadequacy of the implemented model. It often arises that an evaluator cannot be present for the entire duration or may arrive late and thus, not all slots are equally suitable. Further, some evaluators (*e.g.* vice-chairs) have other duties and it is desirable that their consensus meetings be scheduled as early as possible. (One further constraint that the system was required to deal with was that, on occasion, the entire panel meeting is actually a coalition of smaller panels, with evaluators involved in some or all of these smaller panels. It was desirable that smaller panels were completed as soon as possible, allowing the data entry and ranking list generation to take place for these smaller panels.)

An alternative solution strategy is described below.

3 A Refined Model

In addition to constraints 1 - 4, the following refinements are now also considered to address the deficiencies described above.

- R1 No consensus meeting may be scheduled at a time when not all assigned evaluators are present;
- R2 Evaluators with other duties should be scheduled to finish their consensus meeting duties as early as possible;

R3 Some proposals (for example, those associated with a smaller sub-panel) should be scheduled as early as possible

Graph coloring is long associated with timetabling [2–4] and we adopt this approach here. For the model presented in Section 2 we construct a graph $G = (V, E)$, where the vertices represent proposals and an edge exists between two vertices if the corresponding proposals have one or more evaluators in common. In any legal vertex colouring, vertices of the same colour may be scheduled together since they are guaranteed to be non-adjacent. In the absence of limits on evaluators availabilities P_i , the proposals of colour $i = 1, \dots, C$, may be scheduled, respectively, in time slots $S_i, i = 1, \dots, C$.

Using a Tabu search-based vertex colouring algorithm gives quite satisfactory results on problem instances commensurate with that described earlier.

3.1 Restricted Evaluator Availabilities

Restricted availability of one or more evaluators can be accommodated by solving a *maximum cardinality bipartite matching* instance [1]. We construct the bipartite graph $B = (U, V, E)$, where $U = \{i | 1 \leq i \leq C\}$ is the set of colourings and $V = \{j | 1 \leq j \leq T\}$. Vertices u and v are connected by an edge if it is possible to schedule *all* evaluators involved with proposals P_u during time period v . The neighbourhood of u is the set of time slots in which all proposals P_u may be feasibly scheduled.

However, two proposals assigned to the same colour class may require evaluators who cannot be present simultaneously and this will result in vertex u having 0 neighbours, and thus unschedulable. Therefore, it is necessary to add to G , prior to colouring, an edge between every pair of proposals having evaluators not present simultaneously. (A separate but related *feasibility* check ensures that if two evaluators work together on k proposals then there are at least k slots when both are available.

Soft Constraints Constraints R2 and R3 are treated differently since they do not affect feasibility. We build the bipartite graph as previously described but we now add weights to edges. Initially every edge (u, v) has weight 1 but under certain circumstances these weights may be augmented by the following process: for a colour class u and its neighbourhood, $N(u) = \{v_{i_1}, v_{i_2}, \dots, v_{i_m}\}, 1 \leq i_j \leq T, |N(u)| = m$, a weight or bias $b^{i_j}, 0 < b < 1$ is added to each such edge. Different biases may be used for R2 and R3.

On this weighted bipartite graph we call a *maximum weighted bipartite matching* algorithm, which has the effect of choosing earlier time slots for a coloured set of proposals

In the case of constraint R2 each vice-chair is considered in turn, and the previous process is applied to the colourings in which their proposals appear. Likewise, in order to accommodate constraint R3, if a proposal is marked as requiring early completion then it can be thus biased. Funding agency officials have the ability to specify different weightings depending on their priorities.

The system is implemented in Perl and, when run on a Pentium V using problem instance data of the magnitude discussed earlier, returns a schedule (or indicates that there is an infeasibility) in a few seconds. We have also solved problems along the scale of 110 evaluators and 1,000 proposals in approximately 30 seconds.

4 Discussion

Prior to the introduction of this system consensus meetings took place in a haphazard, ad-hoc fashion, with evaluators wasting much effort searching out their associates in order to discuss a proposal. According to one official, for the scale of problem instance we have discussed in Section 2 the system has resulted in panel meetings being completed a day sooner than heretofore.

The problem has been decomposed into a graph colouring subproblem and a matching problem. While the latter is an exact solution, the former finds a heuristically generated colouring. Further, by separating the problem in this manner and ignoring the soft constraints initially we may lose opportunities for finding solutions that are more satisfactory with respect to the soft constraints.

Clearly there is an interaction between the two constraints and the choice of b for each type of soft constraint and this is an area which can be investigated further.

References

1. Helmut Alt, Norbert Blum, Kurt Mehlhorn, and Markus Paul. Computing a maximum cardinality matching in a bipartite graph in time $o(n^{1.5}\sqrt{m/\log n})$. *Inf. Process. Lett.*, 37(4):237–240, 1991.
2. E.K. Burke, K.S. Jackson, J.H. Kingston, and R.F. Weare. Automated timetabling: The state of the art. *The Computer Journal*, 40(9):565–571, 1997.
3. C. Friden, A. Hertz, and D. de Werra. STABULUS: A technique for finding stable sets in large graphs with tabu search. *Computing*, 42:35–44, 1989.
4. D. C. Wood. A technique for coloring a graph applicable to large-scale timetabling problems. *Computer Journal*, 12:317 – 322, 1969.