A Parameter-Free Hyperheuristic for Scheduling a Sales Summit

Peter Cowling Graham Kendall Eric Soubeiga

Automated Scheduling, optimisAtion and Planning (ASAP) Research Group, School of Computer Science and Information Technology
The University of Nottingham - Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, United Kingdom
Email: {pic, gxk, exs}@cs.nott.ac.uk

1 Introduction

Personnel scheduling involves scheduling people to timeslots and possibly locations. This problem remains the subject of much research interest with a survey in every decade since the 1970's [2, 8, 4]. Several models including Constraint Satisfaction Problem (CSP) [6], mathematical programming (linear programming, 0-1 integer programming etc.) [3], and different solution methods (exact [3] or heuristic based [1]) have been proposed to tackle this problem. Also several applications have been reported in the literature including applications for hospital personnel [1], educational institution personnel [7] and others.

However the heuristic methods developed for a particular personnel scheduling problem may not perform well if applied to a different problem. Heuristic and metaheuristic approaches tend to be knowledge rich, requiring substantial expertise in both the problem domain and appropriate heuristic techniques [1]. It is in this context that we proposed a *hyperheuristic* approach [5] as a heuristic that operates at a higher level of abstraction than current metaheuristic approaches. The hyperheuristic has no problem-specific knowledge. It manages a set of simple, knowledge-poor, low-level heuristics. At any given time the hyperheuristic must choose which low-level heuristic to call. The hyperheuristic interacts with the low-level heuristics but is only allowed to communicate non problem-specific information such as CPU time and the change in the evaluation function. Problem-specific domain knowledge is prohibited from passing through the hyperheuristic/low-level heuristic interface. To allow the hyperheuristic to operate, a *choice function* is defined which adaptively ranks the low-level heuristics. The choice function provides guidance to the hyperheuristic by indicating which low-level heuristic should be applied next based upon the historical performance of the heuristics and the region of the search space currently under exploration. In [5] a choice function was defined as a weighted sum of three factors expressing the recent improvement (change in the evaluation function) produced by each of the heuristics, the recent improvement produced by a consecutive pair of heuristics and the time since the heuristic had been called. Although the hyperheuristic produced good results, these results depended on the values of the parameters, and such values were empirically determined by simply choosing the best settings after many trials. Empirically setting the parameters increases the domain-specific knowledge of the hyperheuristic, which is not desirable. This paper is concerned with the development of a mechanism for automatically setting the parameters as well as refining the choice function. To do so we apply the hyperheuristic to a real-world personnel scheduling problem, that of scheduling a sales summit.

The remainder of the paper is structured as follows. We first describe and formulate the sales summit scheduling problem. This is followed by the description of our parameter-free hyperheuristic approach in section 3. Section 4 is devoted to experimental results and section 5 concludes the paper.

2 The Sales Summit Scheduling Problem

The sales summit scheduling problem is that of a commercial company that organises sales summits which involve two groups of company representatives: *suppliers*, who want to sell products or services, and *delegates* who are representatives of companies that are potentially interested in purchasing the products and services. Suppliers pay a registration fee to have a stand at the sales summit and provide a list of the delegates that they would like to meet. A meeting (between one delegate and one supplier) is classified as *Priority* or *Non-Priority* depending on how strongly the supplier would like to meet the delegate. Delegates pay no fee but instead cost money to the commercial company who pays for their travelling and hotel expenses. Besides meetings, seminars are organised where delegates may meet other delegates. Each delegate provides a list of the seminars that he/she would like to attend and (if he/she is invited to the summit) is guaranteed attendance to all seminars. In this instance of the sales summit scheduling problem there are 24 meeting timeslots available for both seminars and meetings, where each seminar lasts three times as long as a supplier/delegate meeting. There are 43 suppliers, 99 potential delegates and 12 seminars. In the first place delegates are assigned to all seminars that they have requested. Then the aim is to schedule meetings, that is, determine (supplier, delegate, timeslot) triples under the constraints: (1) Each delegate must be scheduled for at most 12 meetings; (2) Each delegate must be scheduled for at most one activity (meeting or seminar) within a given timeslot; (3) Each supplier must be scheduled for at most one meeting within a given timeslot; (4) Each supplier should ideally have at least 17 priority meetings; (5)Each supplier should ideally have at least 20 (priority and non-priority) meetings. In practice these two latter constraints are relaxed and the objective is to minimise the number of delegates who will actually attend the sales summit out of the 99 possible delegate attendees and thus minimise cost as well as ensuring that suppliers have sufficient delegate meetings (Priority and Non-Priority). We denote by $\mathbf{S}$ (respectively $\mathbf{D}, \mathbf{T}$) the set of suppliers (respectively delegates, timeslots). Let P_{ij} be 1 if (supplier i, delegate j) is a Priority meeting and 0 otherwise ($i \in \mathbf{S}, j \in \mathbf{D}$). Our decision variables are denoted x_{ijk} ($i \in \mathbf{S}, j \in \mathbf{D}, k \in \mathbf{T}$), where x_{ijk} is 1 if supplier i is to meet delegate j in timeslot k, otherwise x_{ijk} is 0. The formulation is given as follows:

minimise $E(x) = B(x) + 0.05C(x) + 8H(x)$

s.t. $(1)\sum_{i\in\mathbf{S}}\sum_{k\in\mathbf{T}} x_{ijk} \leq 12, \quad (j \in \mathbf{D})$; $(2)\sum_{i\in\mathbf{S}} x_{ijk} \leq 1, \quad (j \in \mathbf{D}, \quad k \in \mathbf{T})$;

$(3)\sum_{j\in\mathbf{D}} x_{ijk} \leq 1, \quad (i \in \mathbf{S}, \quad k \in \mathbf{T})$; $x_{ijk} \in \{0,1\}, \quad (i \in \mathbf{S}, \quad j \in \mathbf{D}, \quad k \in \mathbf{T})$.

$B(x) = \sum_{i\in\mathbf{S}} \left[max\left(0, 17 - \sum_{j\in\mathbf{D}}\sum_{k\in\mathbf{T}} P_{ij}x_{ijk}\right)\right]^2$, $C(x) = \sum_{i\in\mathbf{S}} \left[max\left(0, 20 - \sum_{j\in\mathbf{D}}\sum_{k\in\mathbf{T}} x_{ijk}\right)\right]^2$ and $H(x) = d(x) - 72$ with $d(x) = \sum_{j\in\mathbf{D}} min\left[1, \sum_{i\in\mathbf{S}}\sum_{k\in\mathbf{T}} x_{ijk}\right]$.

Constraint (1) expresses the fact that no delegate must be scheduled for more than 12 meetings. Constraint (2) expresses the fact that no delegate must be scheduled for more than one activity within the same timeslot. Constraint (3) expresses the same idea for suppliers. Both $B(x)$ and $C(x)$ of the objective function are the relaxation of the constraints on suppliers' meeting-satisfaction. More precisely $B(x)$ represents the penalty associated with priority-dissatisfied suppliers (suppliers with less than 17 priority meetings) where constraint (4) is not satisfied, $C(x)$ represents the penalty associated with dissatisfied suppliers (suppliers with less than 20 meetings in total) where constraints (5) is not satisfied. $d(x)$ is the number of delegates who attend the sales summit in the meeting schedule. $H(x)$ represents the penalty associated with the cost of each delegate. The different coefficients in $E(x)$ show the relative importance of each criterion. We penalise only the excess of delegates[1] over 72, to avoid a

[1] In an ideal solution each supplier has the required 20 meetings and there are $43 \times 20 = 860$ meetings. Each delegate can attend at most 12 supplier meetings, therefore at least $\lceil \frac{860}{12} \rceil = 72$ delegates are required.

large constant term in $H(x)$ dominating $B(x)$ and $C(x)$. To simplify the notation we shall assume x and refer to the above quantities as E, B, C, d and H.

The company uses a greedy algorithm that yields a solution of evaluation 444.43 ($B = 226, C = 48.65, d = 99, H = 216$). We developed another greedy algorithm described in [5] which produces a better solution of 225.55 ($B = 52, C = 111, d = 93, H = 168$). This latter solution is used as starting solution in our hyperheuristic algorithm described in the following section.

3 A Parameter-Free Hyperheuristic Method

This work is motivated by the desire to provide a more effective hyperheuristic which does not require parameters to be predetermined. The choice function is a key element in the definition of our hyperheuristic, allowing us to adaptively determine, at each point of the search, how good each low-level-heuristic is likely to be. To define the choice function we regard E in a three-dimensional space with three components B, C and H and $E = B+0.05C+8H$. Firstly the choice function is defined with respect to each criterion $l \in \mathbf{L}$ where $\mathbf{L} = \{B, C, H\}$ is the set of criteria B, C and H. Secondly for each criterion l the corresponding choice function f_l is defined as the combination of three factors reflecting the historical individual performance of each low-level heuristic in terms of change in E, the historical joint performance of each pair of low-level heuristics in terms of change in E and the amount of time elapsed since each low-level heuristic was last called. The first two factors are there for the purpose of intensifying the search using a methodology akin to the exponential smoothing forecast. The third factor constitutes an element of diversification by favouring those low-level heuristics which have not been called for a long time. Thirdly each factor is weighted by a parameter which indicates the relative importance of the factor within the choice function. The choice function is then a weighted sum of three factors. The parameters values are changed throughout the search depending upon whether the hyperheuristic decides that it is appropriate to intensify or diversify the search. This is done by a mechanism wich increases the value of the parameters of the intensification factors when a low-level heuristic proposed by the choice function has produced a solution better than the current one. The reward is proportional to the magnitude of improvement of the solution. Conversely there is a decrease of the parameter value in the case of non-improvement. The penalty is also proportional to the magnitude of worsening of the solution. The intensification factor for which the value of the corresponding parameter is changed is the one that has the largest value among the three factors (the biggest contributor in the choice function). The value of the parameter of the diversification factor is increased when there has not been any improvement after a certain number of iterations corresponding to the number of low-level heuristics. It is decreased to a value that is small enough to allow the selection of low-level heuristics which are effectively improving on the solution.

Overall the search for a good solution can then be regarded (decomposed) as a search for a solution that is good with respect to B, C and H. So when searching for a good solution regarding a criterion $l \in \mathbf{L}$, we shall use the corresponding choice function f_l and the neighbourhood for which f_l is maximum will be applied.

4 Experiments

Our hyperheuristic was implemented in Microsoft Visual C++ and all experiments were run on a PC Pentium II 225MHz with 128MB RAM running under Windows NT Version 4.0. In all experiments the stopping condition was 30 minutes of CPU time. All experiment results are averaged over 10 runs. There are $n = 10$ neighbourhood functions.[2] We used the following four simple hyperheuristic

[2](1)Remove one delegate, (2)Remove and add one meeting to delegate, (3)Add one delegate, (4)Remove and add one meeting to delegate Variant 2, (5)Add one meeting to dissatisfied supplier, (6)Add one meeting to priority-dissatisfied supplier, (7)Cut meeting surplus of supplier, (8)Remove and add one meeting to supplier, (9)Remove and add one meeting

algorithms: *SimpleRandom* chooses one neighbourhood at random and applies it once (single call). *RandomDescent* chooses one neighbourhood at random and applies it until no further improvement is possible (descent call). *RandomPerm* chooses a random sequence of the n neighbourhoods and applies each neighbourhood once in the sequence order. It cycles round from the n^{th} neighbourhood to the first neighbourhood of the sequence. *RandomPermDescent* does the same as *RandomPerm* but each neighbourhood is applied in a steepest descent fashion. Also for each algorithm we distinguished the case where all moves (AM) are accepted and the case where only improving moves (OI) are accepted. We have noticed that algorithms applying neighbourhoods in a descent fashion produce the best results. This was the case of $RandomPermDescent - AM$ which produce the best result of 60.42 as shown in Table 1. Therefore, we decided to use a descent-type call in our new hyperheuristic in addition to the single call. More precisely, neighbourhoods are applied in a single-call fashion in the case of diversification and in a descent-call fashion otherwise. An early version of the hyperheuristic conducted the search according to the following cycle: search for an improvement (a better solution) with respect to H, search for an improvement with respect to B, search for an improvement with respect to C and return to search for an improvement with respect to H and so forth. The hyperheuristic started by "warming up" the choice function by initially running *RandomDescent*. Results obtained were disappointing with the evaluation function value ranging between 62.15 and 89.5. Closer analysis of this hyperheuristic's search pattern revealed that the hyperheuristic had a (sometimes subtle) cycling behaviour, due to the fact that minimising the number of delegates (H) tends to pull in the opposite direction to maximising the quality of the meeting schedule (B, C). To overcome this problem in the final hyperheuristic, for which we report results in Table 1, we decided to choose the search criterion based on a probability distribution which assigns the probability with which a criterion is chosen depending on the relative importance of that criterion in the evaluation function. Since the search criteria are chosen at random, there is little chance of falling into cycles of conflicting neighbourhoods. Furthermore the search effort is now distributed according to the relative importance of each criterion. Other issues were raised during our experimental study. We distinguished between the *absolute improvement* and the *relative improvement*. The absolute improvement is the difference in terms of E between the current solution and the best solution encountered so far. The relative improvement is the difference in terms of E between the current solution and the previous solution. An immediate consequence is that the hyperheuristic is now capable of recognising a key feature of conflicting cycles as sequences of neighbourhood calls during which the absolute improvement is non-positive. We also decided to skip the warm-up period which precedes the hyperheuristic search. Hence the hyperheuristic gets extra time for its search and the problem of determining the proportion of time to devote to the warm-up is eliminated. The resulting hyperheuristic was implemented and results averaged over 10 runs in both AM and OI variants. The results are presented in Table 1. The hyperheuristic in its AM variant, called $Hyperheuristic - AM$ in Table 1, performs significantly better than all the simple algorithms. $Hyperheuristic - AM$ is better than all the simple algorithms in terms of best solution found and worst solution found. We observed that the choice function parameters were evolved to final values consistent with the region of the search currently under exploration. $Hyperheuristic - OI$ gave a worse result than $Hyperheuristic - AM$ because of its more restricted view on the solution space but gave the second best result overall.

5 Conclusion

We have discussed some of the issues involved in the development of a more effective and efficient hyperheuristic approach. These issues are raised during the definition of the choice function which plays a role of primary importance in the choice of which neighbourhood to apply next. Particularly we have observed a considerable advantage in decomposing the choice function into the criteria of the solution. There were problems of conflicting neighbourhoods when the search followed a criterion-based cyclical pattern, which were eliminated by using randomisation. Another important issue was the settings of the choice function parameters throughout the search. We have incorporated a mechanism that allows a readjustment of each parameter depending on the context (diversification, intensification)

to priority-dissatisfied supplier, (10)Remove and add one meeting to priority-dissatisfied supplier Variant 2.

Algorithms	B	C	d	H	**E**	m
Original Greedy Heuristic	226.00	48.65	99.00	27.00	**444.43**	823.00
Initial Greedy	52.00	111.00	93.00	21.00	**225.55**	811.00
SimpleRandom-AM	42.60	86.40	85.20	13.20	**152.52**	819.40
SimpleRandom-OI	30.00	80.40	76.40	4.40	**69.22**	816.60
RandomDescent-AM	31.40	85.20	75.80	3.80	**66.06**	814.60
RandomDescent-OI	29.20	67.20	76.60	4.60	**69.36**	820.20
RandomPerm-AM	12.40	18.20	96.00	24.00	**205.31**	848.60
RandomPerm-OI	29.00	72.40	75.80	3.80	**63.02**	815.20
RandomPermDescent-AM	31.00	92.40	75.10	3.10	**60.42**	811.40
RandomPermDescent-OI	28.20	80.60	76.20	4.20	**66.23**	817.00
Hyperheuristic-AM	34.90	127.00	72.80	0.80	**47.65**	804.40
Hyperheuristic-OI	25.60	88.00	75.40	3.40	**57.20**	813.20

Table 1: m is the number of meetings scheduled

and the resulting change on the evaluation function (improvement) incurred by the application of the neighbourhood. The resulting hyperheuristic is not only autonomous with respect to all parameters but also shows a definite superiority over simple application of multiple neighbourhood techniques in terms of average result, best result and worst result. We intend to demonstrate the effectiveness of our hyperheuristic for other personnel scheduling problems.

References

[1] U. Aickelin and K. A. Dowsland. Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling*, 3:139–153, 2000.

[2] K. Baker. Workforce allocation in cyclical scheduling problems: A survey. *Operational Research Quarterly*, 27(1):155–167, 1976.

[3] A. Billionnet. Integer programming to schedule a hierarchical workforce with variable demands. *European Journal of Operational Research*, 114:105–114, 1999.

[4] D. J. Bradley and J. B. Martin. Continuous personnel scheduling algorithms: a literature review. *Journal Of The Society For Health Systems*, 2(2):8–23, 1990.

[5] P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In E. Burke and W. Erben, editors, *Selected Paper of the Third International Conference on the Practice And Theory of Automated Timetabling PATAT'2000*, Lecture Notes in Computer Science, 2001. To be published in 2001.

[6] A. Meisels and N. Lusternik. Experiments on networks of employee timetabling problems. In E. Burke and M. Carter, editors, *Selected Paper of the Second International Conference on the Practice And Theory of Automated Timetabling PATAT'1997*, Lecture Notes in Computer Science, 1997.

[7] A. Schaerf. Local search techniques for large high school timetabling problems. *IEEE Transactions on Systems, Man and Cybernetics Part A:systems and Human*, 29/4:368–77, 1999.

[8] J. M. Tien and A. Kamiyama. On manpower scheduling algorithms. *SIAM Review*, 24(3):275–287, July 1982.