Do it yourself (DIY) optimisation approach to practical timetabling

Yuri Bykov, Sanja Petrovic, Christos Braziotis

Nottingham University Business School

Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK

yuri.bykov, sanja.petrovic, christos.braziotis@nottingham.ac.uk

This abstract describes our work in progress towards facilitating a greater uptake of metaheuristic optimisation algorithms in practice. Many researchers and practitioners have recognised that there is still a considerable gap between theory and practice in metaheuristic optimisation. Although this gap exists in many application areas, the educational timetabling is the field where it was clearly formulated (see McCollum 2007).

One of the causes of this gap is the inflexibility of the existent timetabling applications, i.e. they cannot enfold the high variety of real-world requirements and restrictions necessary for a good timetable. A survey by Burke et al. (1996) revealed that students and administrative preferences vary greatly among universities. This means that a computer-aided timetabling system developed in one educational institution is unsuitable for another one. Therefore, a common solution here is to order made-to-measure timetabling systems separately tailored for each particular university. This becomes very expensive, time-consuming and inflexible approach, especially when some alterations have to be included into existing systems. As an alternative to that, some universities develop in-house timetabling systems. However, this solution requires a capacity for programming skills by timetabling department staff.

In this study we propose a third variant, which constitutes a middle ground between the above-mentioned extremes. It can be viewed as DIY (do it yourself) optimisation approach because it is more flexible than the first approach but requires fewer user’s skills than the second one. The main idea of the approach is based on the two following observations.

Our first observation is that almost in every timetabling (as well as scheduling, rostering, etc.) research paper we can find a formal definition of a problem as a set of *formulas*. For example, the well-known Carter's formulation of the Exam Timetabling Problem where objective function refers to the proximity of exams (see Carter et al. 1996) is expressed in a mathematical model as follows (taken from Burke et al. 2004):

$$\text{minimize: } \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} \times proximity(t_i t_j)$$

$$\text{where: } proximity(t_i t_j) = \begin{cases} 2^{5-|t_i - t_j|} & if\ 1 \le |t_i - t_j| \le 5 \\ 0 & otherwise \end{cases}$$

$$\text{subject to clash-free requirement: } \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} \times clash(t_i t_j) = 0$$

$$\text{where: } clash(t_i t_j) = \begin{cases} 1 & if\ t_i = t_j \\ 0 & otherwise \end{cases}$$

In these formulas N is the number of exams, c_{ij} are the elements of NxN conflict matrix, which indicate the numbers of students sitting exams i and j together and t_i is the timeslot of ith exam. However, in most cases such formulas are given for reference only and are not used *explicitly* in the supplementary software (experimental or end-user ones). They just formally describe the rules, which are implemented in the form of an *algorithm*, which de-facto represents a more complex procedure than just few formulas. Our second observation is that in other areas there exists software that explicitly operates with formulas for different purposes, such as Matlab for mathematical operations, CPLEX for integer programming or Microsoft Excel for tabbed calculations. By combining these two observations we came up with an idea of the explicit use of mathematical formulas in metaheuristic optimisation as well. It should be noted that the generic idea of embedding a CPLEX-like functionality for increasing the flexibility of metaheuristics is rather straightforward and was circulated in private communications throughout the research community. Therefore, the novelty of our particular contribution to that idea is its effective practical implementation.

To implement this idea in practice a compiler-like intelligent engine was developed, which recognises mathematical notations for the cost function and constraints, verifies them and then prepares them for the use within metaheuristic search algorithm. The formulas can be entered in a quite transparent machine-readable form, which follows common rules used in existing systems (e.g. MS Excel). The detailed description of the rules can be downloaded together with our software. For example, the formulas for the cost function presented above can be entered as follows:

```
cost : sum[i,1,n-1,sum(j,i+1,n,c(i,j)*proximity)]
proximity : when[(1<=absdt) and (absdt<=5),pow(2,5-absdt),0]
absdt : abs[t(i)-t(j)]
```

The corresponding constraints can have the following form:

```
constraint : sum[i,1,n-1,sum(j,i+1,n,c(i,j)*clash)]
clash : when[t(i)=t(j),1,0]
```

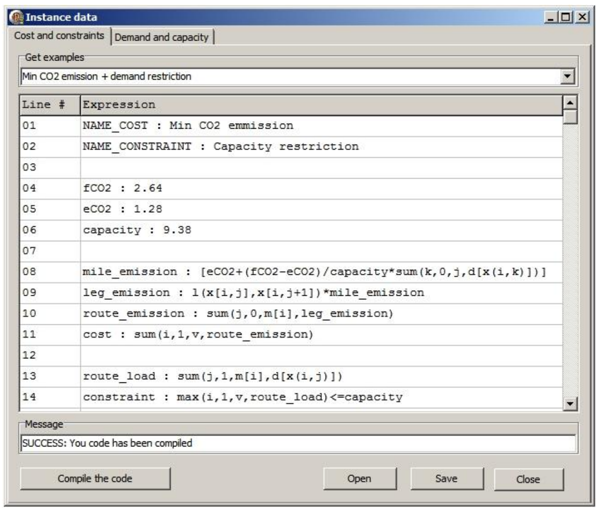
As a result, the user needs just to write five formulas to solve a timetabling problem with Carter’s cost/constraint. Moreover, the user is free to change them or enter completely new ones according to his/her own requirements. If, for example, additionally to the clash-free requirement we need to schedule exam #1 before exam #2 and the number of timeslots should not be more than 15, then the constraint definition could be re-written as:

```
constraint : clash_free and [t(1)<t(2)] and (max[i,1,n,t(i)]<=15)
clash_free : sum[i,1,n-1,sum(j,i+1,n,c(i,j)*clash)]
clash : when[t(i)=t(j),1,0]
```

Our engine recognises virtually any type of cost and constraints, which can be expressed by formulas. This way of cost/constraints specification is much simpler than in-house programming, while at the same time is much more flexible than the ordering of made-to-measure software.

In our particular implementation the engine represents a run-time library, which can be embedded into any optimisation algorithm, not necessary only for timetabling problems. In each case, the engine should be adjusted to the particular problem and solution representations. In the above example (uncapacitated exam timetabling problem), the compiler recognises variables **n** and **c** as the elements of the problem statement and variable **t** as the element of solution. However, if one would like to solve the capacitated problem, the engine should be adjusted in order to recognise the room-related variables.

In order to demonstrate the simplicity and flexibility of the proposed approach and to prove that it is workable we have embedded the prototype cost/constraints compiler into our Vehicle Routing Problem (VRP) solver (the choice of a problem was motivated by a good visual characteristics of VRP solutions, but we anticipate the variant for Exam Timetabling to be ready soon). Figure 1 illustrates a built-in formula editor where the user can enter or change his/her formulas. After pressing the button "Compile the code" the system produces error checks and if no errors are found, incorporates these formulas to the search procedure. In this example, the cost formula represents an amount of CO_2 emissions, but certainly, the user can enter here an unlimited number of possible cost functions.



Line #	Expression
01	NAME_COST : Min CO2 emmission
02	NAME_CONSTRAINT : Capacity restriction
03	
04	fCO2 : 2.64
05	eCO2 : 1.28
06	capacity : 9.38
07	
08	mile_emission : [eCO2+(fCO2-eCO2)/capacity*sum(k,0,j,d[x(i,k)])]
09	leg_emission : l(x[i,j],x[i,j+1])*mile_emission
10	route_emission : sum(j,0,m[i],leg_emission)
11	cost : sum(i,1,v,route_emission)
12	
13	route_load : sum(j,1,m[i],d[x(i,j)])
14	constraint : max(i,1,v,route_load)<=capacity

Message
SUCCESS: You code has been compiled

Fig 1. The cost/constraints entered as formulas

We expect that the variation of the cost/constraints definition could affect the performance of a core optimization technique: therefore the solver offers to choose the preferable one among 6 available metaheuristics: Hill-Climbing, Simulated Annealing, Tabu Search, Great Deluge Algorithm, Late Acceptance Hill-Climbing and Step Counting Hill Climbing (the details of the last two ones can be found in (Burke and Bykov 2008) and (Bykov and Petrovic 2013)). Also the user can select appropriate algorithmic parameters, initialization, restarting/reheating strategies as well as the type of moves (neighborhood operators).

The prototype solver can be downloaded from: http://www.yuribykov.com/MHsolver/. It works in MS Windows with a minimum hardware configuration and does not require an installation procedure.

In order to test the practical effectiveness of the proposed approach, we used our solver as a part of the coursework for the undergraduate module Management Science for Business Decisions at the Nottingham University Business School. This served as a pilot study to assess the usability of our approach in solving the given VRP problems and it involved 10 students without programming skills and with different mathematical background. Among other tasks the students were asked to solve the variants with known and unknown formulas for cost/constraints. The results revealed that all students were able to transform the known formulas to the machine-readable form and enter them into the system. However, some students were not able to solve a more complex task where the formulas are not given but cost/constraints are specified in a textual description. Here 6 out of the 10 students were able to derive a correct formula. These preliminary results suggest that a particular level of mathematical skill/experience is required for the successful use of our approach. Thus, we see the development of a proper training methodology as an important direction of increasing the practical value of our approach.

References:

Burke, E.K., D. Elliman, P. Ford, R. Weare. 1996. Examination timetabling in British universities: a survey. Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science 1153, 76-90.

Burke, E.K., Y. Bykov, J. Newall, S. Petrovic. 2004. A time-predefined local search approach to exam timetabling problems. IIE Transactions 36, 509-528.

Burke, E.K., Y. Bykov. 2008. A late acceptance strategy in hill-climbing for exam timetabling problems. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling PATAT 2008, Montreal, Canada, August 2008.

Bykov, Y., S. Petrovic. 2013. An initial study of a novel Step Counting Hill Climbing heuristic applied to timetabling problems. Proceedings of the 6th Multidisciplinary International Scheduling Conference MISTA 2013, Gent, Belgium, August 2013.

Carter, M.W., G. Laporte, S. Lee. 1996. Examination timetabling: algorithmic strategies and applications. Journal of the Operational Research Society 47, 373-383.

McCollum, B. 2007. A perspective on bridging the gap between theory and practice in university timetabling. Practice and Theory of Automated Timetabling VI, Springer Lecture Notes in Computer Science 3867, 3-23.