"Mine's better than yours" – comparing timetables and timetabling algorithms

Ben Paechter

Automated timetabling papers, along with papers about all types of optimisation methods are full of claims that a particular algorithm, or adaption to an algorithm "out-performs the state-of-the-art". If science is going to move the world forward then we need to be sure what we mean when we make claims such as this and have ways of verifying that the claim is indeed the case. Taking a tutorial approach, this talk begins by looking at how we might decide how good a particular timetable solution might be or how we might, at least, compare one solution with another. Ways of considering the hard and soft constraints are examined, along with methods of dealing with multiple objectives. Three classifications of soft constraints are defined. The need to work with the person using the automated system to fully understand what is "good" about a timetable is underlined. This might include non-obvious criteria, such as the need to reduce the chance that users of the timetable will be able to suggest variations which lead to improvement.

Once we understand how we can compare two solutions to a particular timetabling problem, we can then look at how we might compare two algorithms trying to find good solutions to problems. Factors that might be taken into account are, for example, speed, reliability, closeness to optimum, or the chances of a really super result once in a while. Non-obvious criteria might be for example the extent of the ability for the user to change their mind about what they care about (in terms of either timetable or algorithm quality) in the middle of producing the timetable. Again, in order to be really useful to the world, the emphasis here needs to be on finding out what the users of the algorithm really want from it. The use of standard problem instances is examined along with analysis of the conditions that make this is useful or not. The talk argues that, in order to be useful, problem instances need to be accompanied by one or more standard sets of criteria on which solutions will be measured

Edinburgh Napier University, UK
E-mail: B.Paechter@napier.ac.uk

and, just as importantly, one or more sets of criteria on which algorithms will be measured.

Even when there are clear criteria for judging between algorithms on a specified set of problem instances there can still be problems interpreting the results meaningfully. For example we have to be careful that an algorithm is not just good at solving the particular problem instances, i.e. over-fitted to those instances. One way to try to solve this problem is to have public competitions where algorithms are developed on one set of problem instances, and then compared on another set. Competitions also help to encourage work in a particular area, and work which is genuinely comparative. The talk will discuss the author's experience of running competitions, and what makes a successful competition. The problem of competition entrants using different hardware, compilers, interpreters and operating systems is also discussed. Competitions have largely concentrated on hidden problem instances having the same user criteria. The author will argue that the future may lie in competitions where hidden problems change the user requirements in some way, so as to encourage the development of algorithms which are more generally useful.