Combined Blackbox and AlgebRaic Architecture (CBRA)

Andrew J. Parkes

1 Context

Combinatorial optimisation methods are concerned with an objective function defined over an (exponentially) large search space. Often these are solved with metaheuristics, such as simulated annealing, various evolutionary strategies, tabu search, and many others. However, such metaheuristics are generally rather static creatures with relatively simple behaviours and with most intelligence delegated to the heuristics, the neighbourhoods, selection/acceptance criteria and/or the local search methods that they oversee. This static simpleness has often paid off in terms of simplicity of implementation: Often the metaheuristic is merely tens to hundreds of lines of code, in comparison to maybe tens of thousands of lines to implement complex local search moves (with the associated data-structures needed for them to run quickly). However, the metaheuristics do have some decisions to make, and simple static decisions tend to lead to an over-reliance on careful tuning of parameters. There is a long-standing, but ever-growing, desire that metaheuristics become much more dynamic, self-adaptive, and with inbuilt learning mechanisms, in short, that they become more intelligent. There have been various attempts to add this intelligence. This short abstract cannot hope to do justice to them all, however, some pointers into the recent literature are: Hyper-heuristics (Burke et al 2003; Chakhlevitch and Cowling 2008; Burke et al 2009; Ross 2005; Özcan et al 2008), reactive search Battiti et al (2008), and self-adaptive genetic algorithms. This position paper is an attempt to give one view of how some different approaches are related. (Of course, this is just one view and is not intended to supplant other views.) The starting point is that optimisation methods can often be classified at a very high level according to whether they are blackbox or whitebox:

Blackbox optimisation A blackbox algorithm typically refers to an algorithm that has no insight into the structure of the objective function. Part of the desired intelligence is surely that there is some learning of the structure of the problem. However, in the blackbox case, only inductive learning will be possible. Unfortunately, as known from the ugly duckling theorem (Watanabe 1969) and later the No-Free-Lunch-Theorem (NFLT) (Wolpert. and Macready 1997), when nothing a priori is known about the structure then induction (learning) can be no better than direct enumeration of the search space.

Whitebox optimisation The full structure of the constraints and the objective function is exposed to the solver, and explicitly exploited by it. The archetypal example would be integer programming; the solver would be looking at the internal structure and trying to do deductive learning about new structures, for example, by deriving new cutting planes.

School of Computer Science, University of Nottingham, NG8 1JX, UK
`http://www.cs.nott.ac.uk/∼ajp/`

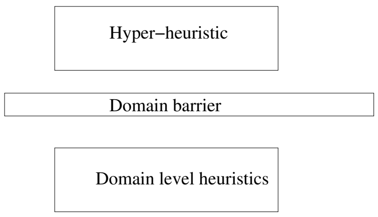

Fig. 1 Hyper-heuristic view of blackbox optimisation. Note that all of the domain is hidden, including all variables.

Such methods are of course not mutually exclusive. The area of constrained blackbox optimisation (cBBO) (e.g. see Wu et al (2006)), roughly speaking, is concerned with solving problems expressed as:

$$\min \quad f^B(x) \tag{1}$$
$$s.t. \; g^W(0) = 0 \tag{2}$$

where the objective function f^B is blackbox, though the constraints g^W are whitebox. In recent work related to timetabling, one can perhaps regard one instantiation of this cBBO framework as the "RAMP" formulation of Bykov[1] in which variables are a set of finite-domain integers, subject to whitebox knapsack and inequality constraints determining feasibility, but also with a blackbox objective function.

The standard selection hyper-heuristic framework is illustrated by the domain barrier of Figure 1 in which the hyper-heuristics are a high-level control system that selects which of various low-level domain-specific heuristics to use, and receives knowledge only of the resulting effect on the objective function. (The term 'hyper-heuristics' also covers other methods, e.g. see Burke et al (2009), but the blackbox selection of heuristics is the most common meaning.) Note that in the classical hyper-heuristic approach the domain is hidden to the extent that the high-level hyper-heuristic controller does not even know of the variables themselves. In contrast, many blackbox methods at least are given a representation of the variables. In genetic algorithm approaches the variables are collected into a chromosome and the blackbox is the objective function.

Hence, overall, there is no unique mix of blackbox and whitebox, which suggests that we should consider more general ways of mixing black and white reasoning. For this, there are 3 general components: the variables, the constraints and the objective(s), and so we discuss each in turn.

Visibility of the variables. The main observation is that work in this area rarely explicitly discusses the black/white nature of the variables or the search space itself. Of course, in any given approach, it is always clear, however, the choices made are often not discussed or modelled explicitly. However, there are a range of possibilities. Here we are driven by some earlier work on the issues of coarse-grained distributed solvers (Parkes 2001) in which the problem is divided into large chunks to be solved on separate solvers. As a means of communication, each chunk is given some partial visibility of the variables of other chunks. Variables within a chunk are hence split into those that are public, x, or private, y. This can also make sense in that encoding many problems results in multiple kinds roles for variables. For example, some are key decision variables, and others are 'mere' auxiliary variables needed to encode some complex expression in the objective.

The primary observation of this paper is that such a public/private or white/black split in the variables should also be explicitly considered as a possibility within general frameworks. In such cases, it seems reasonable that only the public variables, x, are treated as whitebox, and others, y, are hidden inside a blackbox.

The public variables, x, will often simply be a subset of the total set of variables, but in general could be derived variables. So, if there are some underlying total set z of variables then we might express this as:

$$x = p(z)$$

where p is a projection operator that picks out which ones are to be made public. The standard hyper-heuristic corresponds to $p(z) = \{\}$ so that no variables are public. Conversely, genetic algorithms generally

[1] URL http://www.cs.nott.ac.uk/∼yxb/actispec/

take that $p(z) = z$ so that the chromosome is a complete representation. Explicitly introducing such a projection operator could allow an organised study of intermediates between these extremes.

Visibility of the objective(s). There does not seem to be any real, a priori, reason in the cBBO framework, that the entire objective function needs to be completely blackbox, but it could well be a combination of black and white. In this case, by definition, the whitebox portion will only be able to use the public variables, and the objective becomes

$$f^W(x) + f^B(x,y)$$

Visibility of the constraints Generally, there can be whitebox constraints over the public variables, but the blackbox side is also likely to have to account for constraints over all the variables. Note that it could well be that even whitebox constraints are converted to large penalties are then treated in a blackbox fashion (many approaches to cBBO will do this). It is also common that they are treated implicitly within the blackbox portions as hard constraints and so are used to limit the moves in the search space. In the hyper-heuristic approach, it is quite common that the "low-level heuristics" are various neighbourhood moves, but the neighbourhoods are selected so as to preserve feasibility. (For example, in timetabling, the neighbourhood moves might swap times of events only if no new conflicts are created).

Putting all this together the proposed structure is (with $z = (x,y)$):

$$min\ f^W(x) + f^B(z) \tag{3}$$
$$s.t. \quad g^W(x) = 0 \tag{4}$$
$$g^B(z) = 0 \tag{5}$$
$$\text{implicitly with} \qquad x = p(z) \tag{6}$$

Whitebox constraints are often written in algebraic form, so I will refer to as a "Combined Blackbox and algebRaic Architecture" (CBRA) formulation.

However, what would be the point of such a formulation? - Why not just go all black or all white? The main potential advantage is that real-world problems are often a mix of components that most naturally lean towards one or the other. For example, in timetabling the overall time and room choices are a mix of graph colouring and assignment problems and so naturally suggest whitebox methods. On the other hand the pattern constraints (no two events in a row, no more than three in a day, etc) tend to be relatively hard to encode in whitebox style and instead are currently better handled by a good local search method hidden in a blackbox. Note that it is effectively necessary to hide the local search in a blackbox because trying to represent it integer programming or similar is generally awkward and ineffective. In particular, the ability of the framework to exploit only public variables in a whitebox fashion could be useful. (For example, the surface and deep formulations for Course Timetabling in Burke et al (2010) might well have a natural expression in this form.)

Implementing this 'architecture' is future work, but can be mostly expected to be a judicious combination of techniques from cBBO, evolutionary algorithms, hyper-heuristic, and others. The main novelty within this is the public/private variable split, so a key issue is whether this can be handled effectively. Note that the CBRA might well want to fix some or all of the public variables, and so this will need to be passed down to the blackbox objective. In standard hyper-heuristics all variables are blackbox and so this would be handled by an operator requesting the blackbox domain-specific component to create an initial state, but without the top-level getting to know anything about it other than its objective value.

The practical value of this proposal is to help move towards a system giving a good way to handle simultaneously both the extremes of

- whitebox 'algebraic' reasoning about 'simple' constraints over simple public variable
- blackbox handling of complex local moves possibly designed to fix some objectives represented by awkward private variables, and that cannot sensibly be handled with any generic whitebox solver

and also to be able to handle a range of intermediates. Note that this discussion has not really relied at all on any particular choice of algorithms, but is attempting to formalise a representational scheme, and associated overall architecture.

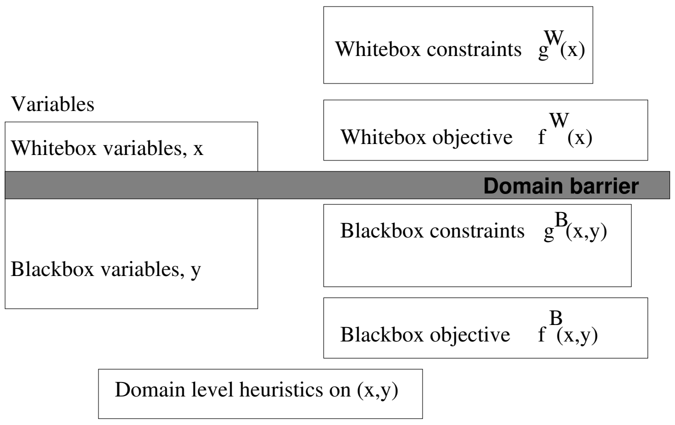
Whitebox constraints $g^W(x)$
Variables
Whitebox variables, x
Whitebox objective $f^W(x)$
Domain barrier
Blackbox constraints $g^B(x,y)$
Blackbox variables, y
Blackbox objective $f^B(x,y)$
Domain level heuristics on (x,y)

Fig. 2 Sketch of CBRA structure.

2 Conclusions

A framework is proposed that includes many existing variants of solvers from fully blackbox to fully whitebox, and most specifically proposes that black/white and mixtures considerations be explicitly applied to variables as well as to objective functions and constraints. In a sense, this paper can be taken as the suggestion that theoretical developments in this direction might well take as much account of the partial hiding of variables as much as they do of the (partial) hiding of the details of objectives.

Two final comments:
Changing black to white? In some cases, in principle, it might well be possible to convert blackbox components to white again by combining promises about the structure of the objective with analysis of results of calls to obtain its value. For example, if the whitebox reasoner where promised that the blackbox just encodes a TSP, then maybe with sufficient acumen it could actually deduce the edge lengths that were hidden and so would then be able to reason directly. This is not so likely to be a practical issue, but could well place limitations on what can be formally proven about such systems: the extra promise that is is an encoding of some particular domain might sometimes lead to breakage of the domain barrier.
Adding "solution features"? In a machine learning hyper-heuristic framework we might well want some features of the solution other than just its objective. For the whitebox components CBRA can use whatever features it decides are useful, but it might well need to also use features of the hidden variables. Presumably these can simply be returned along with the objective function.

References

Battiti R, Brunato M, Mascia F (2008) Reactive Search and Intelligent Optimization, Operations Research/Computer Science Interfaces Series, vol 45. Springer

Burke EK, Hart E, Kendall G, Newall J, Ross P, Schulenburg S (2003) Handbook of Meta-Heuristics, Kluwer, chap Hyper-Heuristics: An Emerging Direction in Modern Search Technology, pp 457–474. URL http://www.asap.cs.nott.ac.uk/publications/pdf/hhchapv002.pdf

Burke EK, Hyde MR, Kendall G, Ochoa G, Özcan E, Woodward J (2009) A classification of hyper-heuristic approaches. Tech. Rep. NOTTCS-TR-SUB-0907061259-5808, University of Nottingham, School of Computer Science.

Burke EK, Mareček J, Parkes AJ, Rudová H (2010) Decomposition, reformulation, and diving in university course timetabling. Comput Oper Res 37(1):582–597, DOI http://dx.doi.org/10.1016/j.cor.2009.02.023

Chakhlevitch K, Cowling P (2008) Hyperheuristics: Recent developments. Studies in Computational Intelligence 136:3–29, in Cotta C, Sevaux M, Sorensen K (eds) Adaptive and Multilevel Metaheuristics

Özcan E, Bilgin B, Korkmaz EE (2008) A comprehensive survey of hyperheuristics. Intelligent Data Analysis 12(1):3–23

Parkes AJ (2001) Exploiting solution clusters for coarse-grained distributed search. In: IJCAI01 Workshop on Distributed Constraint Reasoning, Seattle, Washington, USA

Ross P (2005) Hyper-heuristics. In: Burke EK, Kendall G (eds) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer, chap 17, pp 529–556

Watanabe S (1969) Knowing and Guessing: A Quantitative Study of Inference and Information. New York: Wiley

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1:67

Wu Y, Ozdamar L, Kumar A (2006) Nonconvex Optimization and Its Applications, vol 85, chap Parallel Triangulated Partitioning for Black Box Optimization, pp 487 – 506