A Research Agenda for Metaheuristic Standardization

Jerry Swan, Steven Adriaensen, Mohamed Bishr, Edmund K. Burke, John A. Clark, Patrick De Causmaecker, Juanjo Durillo, Kevin Hammond, Emma Hart, Colin G. Johnson, Zoltan A. Kocsis, Ben Kovitz, Krzysztof Krawiec, Simon Martin, J. J. Merelo, Leandro L. Minku, Ender Özcan, Gisele L. Pappa, Erwin Pesch, Pablo García-Sánchez, Andrea Schaerf, Kevin Sim, Jim E. Smith, Thomas Stützle, Stefan Voß, Stefan Wagner, Xin Yao.

1 Introduction

We propose that the development of standardized, explicit, machine-readable descriptions of metaheuristics will greatly advance scientific progress in the field. In particular, we advocate a purely functional description of metaheuristics — separate from any metaphors that inspire them and with no hidden mechanisms. A recent policy statement in the *Journal of Heuristics*[1] highlights the need for improved research practice for metaheuristics via increased transparency of implementation and understanding of the contribution of their component parts. We describe here how addressing these issues via explicit descriptions can also offer further benefits. Standardization and pure-functional descriptions promote a higher standard of rigor for both communication and reproducibility of results. The modularity of our proposed approach opens up opportunities to compose heuristics in novel ways, along with better support for parallel processing. Most significantly, it is the basis for a greater degree of mechanized reasoning: we discuss how this might support large-scale collaborative research activity, leading to automated discovery, mining and assembly of metaheuristics.

2 A pure-functional description of metaheuristics

The Generalized Local Search (GLS) framework [5] gives a broad definition of the search process in terms of an automaton with an external memory. This memory represents 'environmental state', which can include whatever information is required by the search process, e.g. memoized information derived from the search trajectory.

The general approach to casting this in pure-functional terms takes the environment as an additional argument/return value for each metaheuristic (e.g. Genetic Algorithm) and its component heuristics (acceptance, perturbation, etc.). For example, perturbation is expressed as:

$$perturb_{State,Env} : State \times Env \to State \times Env$$

where *State* is a generic placeholder for the solution representation (bitstrings, permutations, or populations / Pareto-fronts thereof, etc.), and *Env* is a generic placeholder for any additional information needed by *Perturb* or its invoking metaheuristic.

So, each metaheuristic is parameterized by its own environment and the environments of the components it invokes. We therefore reap the well-known benefits of functional descriptions: the abstractions are simple and compose easily, state is represented explicitly rather than implicitly, and results are deterministic, making each component's behavior easy to reproduce consistently even when used in a variety of metaheuristics.

Pure functional approaches have previously been applied to metaheuristics (e.g. [15, 12]) but have been typically been confined to a 'proof-of-concept' implementations in a single programming language. What we propose here differs in that a) the explicit parametization by environment allows heuristics to be composed in a modular fashion b) the functional approach is rather the basis of a wider research agenda towards standardization, with the associated referential transparency being a cornerstone for reasoning about heuristics by both human and machine. In the following section, we outline this agenda and the associated benefits in more detail.

[1] `http://goo.gl/61USme`

3 The benefits

Communicability and reproducibility Metaphors often inspire new metaheuristics, but without mathematical rigor, it can be hard to tell if a new metaheuristic is really distinct from a familiar one [14]. For example, mathematically, 'Harmony search' turned out to be a simple variant of 'Evolution Strategies' even though the metaphors that inspired them were quite different [17]. Formally describing state, representation, and operators allows genuine novelty to be distinguished from minor variation. The formality of purely functional descriptions promotes principled decomposition of frameworks [8] and a degree of clarity that enables reproducibility.

Interoperability Because all environmental requirements are made explicit, a greater re-use of frameworks and components is possible. In particular, this explicit description allows the invocation of remotely-hosted (meta-)heuristics, offered as a Service Oriented Architecture (SOA) [4] through stateless web services. This also allows the creation of generic (i.e. metaheuristic-independent) tools for such tasks as parameter-tuning and visualization.

Automated assembly of metaheuristics Functional descriptions make inputs and outputs explicit, and allow no side-effects. This greatly facilitates automated assembly of metaheuristics. Software can search for effective ways to combine metaheuristics 'bottom-up', avoiding the human bias inherent in choosing specific metaheuristics *a priori* [9]. In particular, the modularity of the proposed approach permits credit assignment and facilitates the (automatic) detection and elimination of accidental complexity (as in [1]), making components easier to understand and reuse in different contexts.

Knowledge engineering Many aspects of the search process can be described formally in a domain-independent manner. A trivial example is the notion of the inverse of some operator; more sophisticated examples are given in [7, 10, 11]. The environment parameter offers a way to inject rich information (from either or both of the problem- and solution- domains) directly into the metaheuristic search process. An algorithm can exploit such knowledge in two broad ways:

- Offline—that is, from knowledge arising from the problem *class*. For example, the list of cities in the TSP form a permutation, so the algebra of group theory can be exploited to devise new operators.
- Online—that is, from knowledge arising from the observed behavior of a metaheuristic operator as it is applied to a specific problem *instance*.

Machine-readable representation of domain knowledge enables a shift in metaheuristic design practice from software engineering to knowledge engineering: creating *declarative* descriptions of a problem and letting software apply that knowledge, rather than embedding knowledge in procedural program code. The goal is to allow mainstream metaheuristics to make use of the kind of 'domain-independent domain knowledge' common in formal methods and constraint programming, leading to the development of reusable libraries of problem domains, solution representations and operators, in declarative form.

Efficiency Many popular metaheuristics are extremely well-suited to being split into subtasks that can be performed simultaneously on separate CPU cores. The purely-functional approach advocated here avoids race conditions and facilitates the automated application of parallelism. In particular, heuristics can be chained using a functional-programming mechanism known as a *monad*. Monads allow the composition of functions in ways that follow well-defined mathematical laws [16]. Monads make explicit the interactions between variables that accompany flow of control (e.g. the inner loop of iterated perturbation). This simplifies the propagation of environmental state and provides convenient placeholders for the injection of parallelism.

Scientific advancement Combining the data from the great volume of metaheuristic research into a general theory that explains their results is nearly impossible without a common, formal vocabulary for describing those results. Purely functional descriptions will enable researchers to better combine, exchange, mine, and reason about metaheuristics and their components, greatly increasing the impact of approaches such as [3, 13].

4 Research questions

The pursuit of our proposed agenda will raise some important new research questions, namely:

i). How can semantic/whitebox descriptions of problem- and solution domains best bring declaratively-driven approaches, such as those common in the scheduling and planning communities, into main-stream metaheuristics?

ii). What kinds of declarative descriptions might be required for 'domain-driven' algorithm generation—including automated problem reformulation [2] and formal approaches to program synthesis [6]?

5 Conclusion

We have advocated a research program leading to greater automation of metaheuristic practice. As a minimum, the need to build upon existing work means that practitioners should be able to assemble (perhaps remotely-hosted) collections of heuristics to form a solver, with this choice being informed by a collective 'performance repository', relating problem features and algorithm behavior across a range of problem instances. A community initiative toward standard machine-readable descriptions of frame-works, components, and experimental results will do much to accelerate scientific progress.

References

[1] Steven Adriaensen, Tim Brys, and Ann Nowé. Fair-share ILS: a simple state-of-the-art iterated local search hyperheuristic. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 1303–1310. ACM, 2014.

[2] Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. Decomposition, reformulation, and diving in university course timetabling. *Comput. Oper. Res.*, 37(3), March 2010.

[3] Pietro Consoli, Leandro L. Minku, and Xin Yao. Dynamic selection of evolutionary algorithm operators based on online learning and fitness landscape metrics. In *10th International Conference on Simulated Evolution And Learning*, volume 8886 of *LNCS*. Springer, 2014.

[4] Pablo García-Sánchez, J. González, Pedro A. Castillo, Maribel García Arenas, and Juan Julián Merelo Guervós. Service oriented evolutionary algorithms. *Soft Comput.*, 17(6):1059–1075, 2013.

[5] Holger Hoos and Thomas Stützle. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann, 2005.

[6] Emanuel Kitzelmann. Inductive programming: A survey of program synthesis techniques. In Ute Schmid, Emanuel Kitzelmann, and Rinus Plasmeijer, editors, *Approaches and Applications of Inductive Programming*, volume 5812 of *LNCS*, pages 50–73. Springer Berlin Heidelberg, 2010.

[7] Krzysztof Krawiec and Jerry Swan. Pattern-guided Genetic Programming. In *GECCO '13 Companion*, 2013.

[8] Manuel López-Ibáñez, Franco Mascia, Marie-Éléonore Marmion, and Thomas Stützle. A template for designing single-solution hybrid metaheuristics. In *GECCO Comp '14*, pages 1423–1426, New York, USA, 2014.

[9] Matthew A. Martin and Daniel R. Tauritz. Evolving black-box search algorithms employing genetic programming. In *GECCO Comp '13*, NY, USA, 2013.

[10] J. C. Ortiz-Bayliss, H. Terashima-Marín, S. E. Conant-Pablos, Ender Özcan, and A. J. Parkes. Improving the performance of vector hyper-heuristics through local search. In *GECCO '12*, NY, USA, 2012. ACM.

[11] Andrew J. Parkes. Combined Blackbox and AlgebRaic Architecture (CBRA). In *PATAT 2010*, 2010.

[12] Richard Senington and David Duke. Decomposing metaheuristic operations. In Ralf Hinze, editor, *Implementation and Application of Functional Languages*, LNCS. Springer Berlin Heidelberg, 2013.

[13] Kate Smith-Miles, Davaatseren Baatar, Brendan Wreford, and Rhyd Lewis. Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45, 2014.

[14] Kenneth Sörensen. Metaheuristics - the metaphor exposed. *International Transactions in Operational Research*, 22(1), 2015.

[15] Pascal Van Hentenryck, Laurent Michel, and Liyuan Liu. Constraint-based combinators for local search. In Mark Wallace, editor, *Principles and Practice of Constraint Programming CP 2004*, volume 3258 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004.

[16] Philip Wadler. The essence of functional programming. In *Proceedings of POPL '92*, New York, USA, 1992.

[17] Dennis Weyland. A rigorous analysis of the Harmony Search algorithm: How the research community can be misled by a "novel" methodology. *Int. J. Appl. Metaheuristic Comput.*, 1(2), 2010.