A Template for Designing Single-Solution Hybrid Metaheuristics

Manuel López-Ibáñez
IRIDIA, CoDE,
Université Libre de Bruxelles
Brussels, Belgium
manuel.lopez-ibanez@ulb.ac.be

Franco Mascia
IRIDIA, CoDE,
Université Libre de Bruxelles
Brussels, Belgium
fmascia@ulb.ac.be

Marie-Éléonore Marmion
LIFL, Université Lille 1
Lille, France
marie-eleonore.marmion@inria.fr

Thomas Stützle
IRIDIA, CoDE,
Université Libre de Bruxelles
Brussels, Belgium
stuetzle@ulb.ac.be

ABSTRACT

Single-solution metaheuristics are among the earliest and most successful metaheuristics, with many variants appearing in the literature. Even among the most popular variants, there is a large degree of overlap in terms of actual behavior. Moreover, in the case of hybrids of different metaheuristics, traditional names do not actually reflect how the hybrids are composed. In this paper, we discuss a template for single-solution hybrid metaheuristics. Our template builds upon the Paradiseo-MO framework, but restricts itself to a predefined structure based on iterated local search (ILS). The flexibility is given by generalizing the components of ILS (perturbation, local search and acceptance criterion) in order to incorporate components from other metaheuristics. We give precise definitions of these components within the context of our proposed template. The template proposed is flexible enough to reproduce many classical single-solution metaheuristics and hybrids thereof, while at the same time being sufficiently concrete to generate code from a grammar description in order to support automatic design of algorithms. We give examples of three IG-VNS hybrids that can be instantiated from the proposed template.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2881-4/14/07 ...$15.00.
http://dx.doi.org/10.1145/2598394.2609846 .

Keywords

Automatic Algorithm Configuration, Automatic Algorithm Design, Local Search, Metaheuristics

1. PROBLEM STATEMENT

Single-solution metaheuristics are those metaheuristics that consider a single solution at each step instead of a population of solutions [3]. These include classical local search algorithms, such as iterated local search (ILS) [7], simulated annealing (SA) [6], variable neighborhood search (VNS) [2], among others; and also classical evolution strategies such as (1+1)-ES and (1,1)-ES [1].

Given the large number of single-solution metaheuristics proposed in the literature, it is not surprising that very similar proposals are discussed under different names. For example, *iterated greedy* (IG) [10] is fundamentally equivalent to *ruin-and-recreate* (RR) [11] and *large-neighborhood search* (LNS) [9]. This duplication is problematic for two reasons: (1) it artificially divides the research concerning these algorithms leading to a lack knowledge sharing and a duplication of efforts, and (2) it makes more difficult for practitioners to understand the state-of-the-art and identify the best algorithm for their particular problem.

A related but different problem arises when well-known algorithms are slightly modified or hybridized with other metaheuristics. For instance, ILS could be augmented with the Metropolis acceptance criterion typical of SA. In such a case, depending on the point of view, it could be considered either a hybrid of ILS and SA or an SA with a large neighborhood and a local search intensification. Both views basically describe the same algorithm.

We believe that previous approaches suffer from an excessive flexibility [4, 14]. They attempt to instantiate almost any possible algorithm, which leads to a high degree of flexibility, but allows defining the same algorithm in multiple equivalent ways. Nonetheless, our proposal is not a radical departure from existing approaches, since our goal is to synthesize the accumulated knowledge about which single-solution metaheuristics, and which of their components, have been shown to work well on various problems.

In particular, we build directly on top of the Paradiseo-MO C++ framework [4] by reusing many of its underlying components, although other underlying frameworks and programming languages may be used to implement the template described here.

2. SOLUTION

Our proposal sacrifices some of the flexibility of Paradiseo-MO for simplicity by pre-defining an algorithmic structure based on ILS (Fig. 1). A classical ILS algorithm is composed of a perturbation, a local search, an acceptance criterion and a termination criterion. However, in our template, these components are much more general than those found in classical ILS algorithms. Moreover, the components may influence each other by updating a common state, which may represent a strength parameter in the case of perturbation, a temperature in the case of the Metropolis acceptance criterion, a history of previous solutions, a tabu list, etc.[1]

We define a perturbation as a stochastic move within a particular neighborhood. A perturbation is often defined in terms of a strength parameter, which may be dynamically adjusted. The neighborhood of the perturbation may also be the solutions that can be reached in a certain number of consecutive moves. In that case, the strength parameter defines the number of moves for each perturbation. The class of perturbations includes not only classical random swap moves, but also the complex destruction-reconstruction moves typical of IG (and by extension, RR and LNS). The key feature of a perturbation is that it is stochastic, that is, applying the same perturbation twice with the same strength to the same solution is likely to produce different solutions. Moreover, perturbations may react to a new solution being accepted as the current one by adjusting the strength.

An acceptance criterion is defined as a procedure that takes the current solution and a new solution and returns either one of them or a previously found one. The solution returned then becomes the current solution. Examples of acceptance criteria are: accept always the new solution, which is typical of restart mechanism; accept only if improving, which is typical of ILS and VNS, and Metropolis acceptance, which is the acceptance criterion used in the classical SA. Some of these acceptance criteria, such as the Metropolis one, have internal parameters.

A local search, in our template, is defined as any classical iterative improvement plus the ILS itself. Iterative improvement algorithms are not based on stochastic perturbations as defined above, but on a systematic search within a neighborhood. Therefore, they are not part of the algorithms that can be instantiated from our ILS template. Moreover, iterative improvement algorithms are often problem-specific in order to make use of particular speed-ups when systematically exploring a neighborhood. Since the ILS contains a local search, and the local search can be an ILS, this recursive definition allows hybridizing different types of single-solution metaheuristics within a single algorithm. The idea of defining hybrid ILS algorithms by means of a recursive definition has already been explored in the literature in the context of the traditional ILS definition [5]. More generally, the idea that a template for local search should have a recursive definition is even older [14]. Recently, the same ideas have been applied to hyperheuristics templates [12, 13].

[1]In the case of Paradiseo-MO, this state is not passed explicitly between functions as in Fig. 1, but implemented in terms of relationships between different objects.

```
ILS Algorithm
 1: s_0 := Initialization()
 2: s* := ILS(s_0)
 3: return s*
```

```
Function ILS(s_0)
Require: Perturbation, LS,
    Acceptance, Termination
 1: s* := LS(s_0, State)
 2: repeat
 3:     s' := Perturbation(s*, State)
 4:     s' := LS(s', State)
 5:     s* := Acceptance(s', s*, State)
 6: until Termination(State)
 7: return s*
```

Figure 1: The iterated local search (ILS) template.

Finally, the termination criterion is defined as a function that returns true when the main loop of ILS should stop. It can be a time limit, a number of iterations or another condition. In the case of an ILS embedded within another ILS, the termination condition of the ILS at the lower level is often a function, such as a fraction of time, of the termination condition of the upper level.

3. CONSEQUENCES

Despite being more restricted than the general framework given by Paradiseo-MO [4], the above template can instantiate a large number of classical single-solution metaheuristics, including SA, ILS, IG, VNS, etc. Moreover, it can combine components of them in a flexible (but not entirely arbitrary) way. We will discuss some examples below.

One of the consequences of such a framework is to move away from describing algorithms in terms of similarities with other algorithms, which is actually a subjective view that depends on our own background, towards describing algorithms in terms of the components that they use. That is, a SA-VNS hybrid could be implemented in various ways, but an algorithm that uses a perturbation with a dynamically increasing strength and a Metropolis acceptance criterion has always the same structure in the template proposed above.

Another consequence is that, except for a few problem-specific components, generating a hybrid single-solution metaheuristic is a matter of choosing how many levels of ILS we wish to have and which components should appear at each level. This property allowed us to generate code in a procedural manner from a grammar description of the above template and to find the best instantiation using an off-the-shelf automatic algorithm configuration tool [8].

4. EXAMPLES

As an example, let us consider a hybrid IG-VNS. Such a name on its own is not very helpful, since what is exactly a hybrid IG-VNS depends on the actual components and how they are combined. In terms of our ILS template, we have at least the following three possibilities:

IG that restarts from solutions increasingly distant from the best-so-far. In this case, there is an IG algorithm that, after not improving the best-so-far solution for a while, applies a perturbation to it and restarts the IG algorithm from the perturbed solution. Every time such a restart occurs without improving the best-so-far solution, the perturbation applied is stronger. This corresponds to a two level ILS, where the perturbation of the inner ILS is of the type destruct-reconstruct, whereas the perturbation of

the outer ILS is a partial reinitialization of the best-so-far solution with a strength that increases as long as no new best-so-far solution is found.

IG with increasingly stronger destruction-reconstruction. This corresponds to a single level ILS, where the perturbation is of type destruct-reconstruct, but the strength of the perturbation increases as long as no new best-so-far solution is found.

IG with a VNS as subsidiary local search. This corresponds to a two level ILS, where the perturbation of the outer level is of the type destruct-reconstruct and the inner level is a classical VNS algorithm.

The above examples show that complex hybrid algorithms can be described in terms of very few components based on the template proposed above.

5. REFERENCES

[1] H.-G. Beyer and H.-P. Schwefel. Evolution stratagies: a comprehensive introduction. *Natural Computing*, 1: 3–52, 2002.

[2] P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[3] H. H. Hoos and T. Stützle. *Stochastic Local Search—Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[4] J. Humeau, A. Liefooghe, E.-G. Talbi, and S. Verel. ParadisEO-MO: From fitness landscape analysis to efficient local search algorithms. *Journal of Heuristics*, 19(6):881–915, June 2013.

[5] M. S. Hussin and T. Stützle. Hierarchical iterated local search for the quadratic assignment problem. In M. J. Blesa et al., editors, *Hybrid Metaheuristics*, volume 5818 of *LNCS*, pages 115–129. Springer, 2009.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220: 671–680, 1983.

[7] H. R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 321–353. Kluwer Academic Publishers, Norwell, MA, 2002.

[8] M.-E. Marmion, F. Mascia, M. López-Ibáñez, and T. Stützle. Automatic design of hybrid stochastic local search algorithms. In M. J. Blesa et al., editors, *Hybrid Metaheuristics*, volume 7919 of *LNCS*, pages 144–158. Springer, 2013.

[9] D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer, New York, NY, 2 edition, 2010.

[10] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049, 2007.

[11] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139 – 171, 2000.

[12] J. Swan, E. Özcan, and G. Kendall. Hyperion - a recursive hyper-heuristic framework. In C. A. Coello Coello, editor, *Learning and Intelligent Optimization, 5th International Conference, LION 5*, volume 6683 of *LNCS*, pages 616–630. Springer, 2011.

[13] J. Swan, J. Woodward, E. Özcan, G. Kendall, and E. Burke. Searching the hyper-heuristic design space. *Cognitive Computation*, 6(1):66–73, Mar. 2014.

[14] R. J. M. Vaessens, E. H. L. Aarts, and J. K. Lenstra. A local search template. *Computers & Operations Research*, 25(11):969–979, 1998.