Evolutionary Cross-domain Hyper-Heuristics

Patricia Ryser-Welch
University of York
York Road
York
Patricia.Ryser-Welch@York.ac.uk

Julian F. Miller
University of York
York Road
York
Julian.Miller@York.ac.uk

Shahriar Asta
Notthingham University
Wollaton Road
Nottingham
sba@cs.nott.ac.uk

1. INTRODUCTION

Designing effective algorithms to solve computational problems is difficult and time-consuming. The standard methodology for designing such algorithms is "top-down". This process breaks down large problems into more understood components and eventually identifies problem-specific operators that algorithms need to use to solve the given problem. Often, restrictive assumptions have to be made about the use of operators within an algorithm.We argue that it is desirable to automate this process. A wider of range of possible algorithms can be generated automatically and new TSP solvers could be discovered, in a reasonable amount of time and without the restrictions imposed by the human mind.

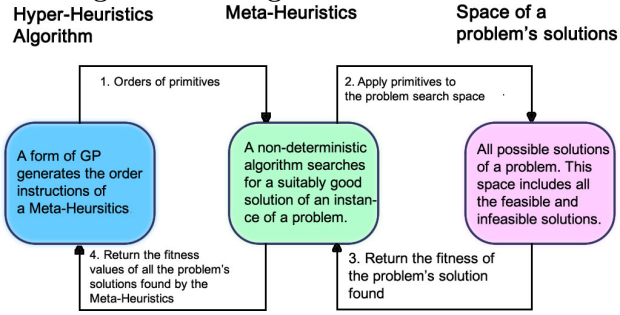
We focus on evolving a fixed sequence of operators inside the loop of a Memetic Algorithm, using an innovative automatic algorithm creation method. We are proposing to extract and hard-code these evolved algorithms in new independent solvers, to find good solutions to a chosen problem.

2. CROSS-DOMAIN HYPER-HEURISTICS

Hyper-heuristics searches the space of heuristics and meta-heuristics, so that it can generate high-quality algorithms for a problem. Algorithms have been constructed iteratively using "templates of operations" based on well-known heuristic and meta-heuristic methods (i.e. Iterated Local Search and Memetic algorithms). Problem-specific heuristics are chosen iteratively during the search to find better solutions in the problem search space.

Cross-Domain Hyper-heuristics guides the constructions of algorithms, using a general-purpose *"template of types of instructions"*. Instead of referring to a specific primitives, the template randomly chooses a problem operator from a specific subset. These subsets can include mutation, crossover or even a Local Search operator. The results of the CHeSCs 2011 competition represent the state-of-the art in the automatic selection of algorithms for optimisation problems. It was made possible by the use of the HyFlex framework [3][1], which includes several test problems together with their specific heuristics. For the TSP, this framework includes 10 benchmarks and 13 low-level heuristics for the Travelling Salesman Problem.

Figure 1: Process used to calculate the fitness value of an algorithm during evolution.



The advantage of *Cross-Domain hyper-heuristics* lies in letting the programmers develop an automatic algorithm creation method, without any extensive knowledge of the problems to solve. These "adaptive algorithms" have solved several well-established combinatorial problems, with a high level of generality. However, the evolved sequences of heuristic operations are often very long, not re-usable and defy human comprehensibility.

3. THE PROPOSED METHOD

We are proposing to use evolution to automatically design a high-quality solver, by letting Hyper-heuristic algorithms assembling and testing part or the whole algorithm (in blue in figure 3). At each iteration of the Hyper-Heuristic search, the fitness of valid sequences are calculated by executing a hybrid meta-heuristic several times (i.e. steps 1,2, 3 of figure 3). All the fitness values of the problem solutions found by the meta-heuristics (see step 4 of figure 3) are returned the Hyper-heuristic algorithm; the average of these values determine the quality of the algorithm. At the end of this process, a generated algorithm can then be extracted to be coded with a programming language. We determine whether the quality of the generated sequences of instructions by analysing their performance and structure, in an independent and subsequent process.

[1]Details of the challenge and the results can be found at http://www.asap. cs.nott.ac.uk/external/chesc2011/ and http://www.hyflex.org/chesc2014/

GECCO'15 Companion, July 11–15, 2015, Madrid, Spain.
ACM 978-1-4503-3488-4/15/07..
http://dx.doi.org/10.1145/2739482.2764644

Chosen problem domain: More formally, let $G = (V, E)$ define a graph, where $V = \{1, 2, ..., n\}$ is a vertex set and E the set of edges. Denote, $C = c_{i,j}$ to represent a *"weight"* matrix associated with E, that models the distance from city i to city j. When a tour is represented as a permutation $(i_1, i_2, ..., i_n)$, the *"cost matrix"* becomes an essential element to calculate its overall distance. The quantity to minimise becomes $c_{i_1,i_2} + c_{i_2,i_3} + ... + c_{i_n,i_1}$ [1]. For our purporse, we have chosen to use the following heuristics offered by Hyflex; *Order Based Crossover, Partially-Mapped Crossover, Subtour-Exchange Crossover, Insertion Mutation, Exchange Mutation, Scramble Mutation, Simple Inversion Mutation, 2-opt Local Search, 3-opt Local Search, Best 2-opt Local Search.* Additionally *ReplaceLeastFit, ReplaceRandom, RestartPopulation* alters the population of TSP solutions through generations. We will be using all the TSP problem instances offered by Hyflex. These benchmarks include problems with various numbers of cities ranging from 299 to 18512. The names of these problems are: PR299, PR439, RAT575, U724, RAT783, PCB1173, D1291, USA13509, and D18512. Hyflex parameters were set to to 0.89 for the depth of the local search, increasing the number of maximum number of iterations to 40 for any local search.

Meta-heuristics: Our hybrid Memetic Algorithm (the green component in figure 3) applies the template described in algorithm 1. The body of the loop of a Memetic Algorithm is generated by the evolutionary Hyper-Heuristic algorithm (CGP). This template prevents having an invalid algorithm.

Algorithm 1 : The template of an hybrid MA

```
1: p_0 ← GenerateInitialSolution()
2: p ← Apply a Local Search(p_0)
3: while Not optimum and EvalCount < MaxEvals do
4:    t ← SelectParents(p)
5:    NumEvals = 0
6:    while Not end of evolved sequence of operations do
7:       Apply current operation to t or p
8:       NumEvals = NumEvals + 1
9:    end while
10:   EvalCount = EvalCount + NumEvals
11: end while
```

Hyper-heuristics algorithm : Cartesian Genetic Programming (CGP) is used to automatically generate the sequence of instructions. We use a one-dimensional CGP geometry with 100-nodes. To search the algorithm space, we use a $1+1$ Evolutionary Strategy with a maximum number of allowed iterations set to 1200 and a mutation rate of 5%. The reason why such a simple evolutionary strategy works well is primarily due to the presence of non-coding genes. In our case this means that the applying simple mutations can explore a wide distribution of evolved metaheuristics. This allows continual exploration of the algorithm space even if the algorithm fitness (performance measure) is fixed [2].

4. EXPERIMENTAL RESULTS

CGP has favoured sequences with hill-climbing operators; one sequence applies *Best 2- Opt Local Search, Simple Inversion Mutation, 3-Opt Local Search, Best 2-Opt Local Search and ReplaceLeastFit* . This sequence tend to move towards a minima more efficiently. In the first 1000 generations, this hybrid Memetic Algorithms reduces quickly the length of the tours, then better TSP solutions are found as the search approach nearer and nearer the known minima. The relative error was obtained using the formulae: $\frac{tour\ length - known\ optimum}{known\ optimum}$. The median relative errors goes over marging for instances between 299 and 2152 cities; it was below 4%, with the exception of the benchmark D1291, which increases to 8% approximately. However, with more evaluations it is likely that the known best global optimum could be found and indeed perhaps even be surpassed.

5. CONCLUSIONS

We have presented a new Hyper-heuristics method that not only optimises meta-heuristics, but also allows them to be extracted and analysed. We show that not only can the method can produce human-readable, but also effective new algorithms. The results of our experiments are promising. Close solutions to the actual known optima have been found for the TSP benchmarks. We would have preferred to have established new global optima though. However, this could be just a matter of more evaluations. Nonetheless, we believe evolutionary cross-domain hyper-heuristics needs to be applied to other problems, such the personal scheduling and the vehicle routing problems to generate new hybrid memetic algorithms. Then the full potential of this technique can be fully evaluated. The concept of evolutionary memetic algorithm could then be extended to evolutionary meta-heuristics.

In the future we intend to investigate less restrictive patterns of instructions, in order to learn new possible sequences of operations that humans have not yet though of, but still remain readable and understandable. Perhaps these new algorithms could lower the known minima of these instances. It would also be interesting to use an encoding scheme that is expressive enough to encode more challenging programming structure (i.e. iterations).

6. ACKNOWLEDGMENTS

Thanks to Ender Ozcan and Gabriela Ochoa for kindly answering to all our requests. This work made use of the facilities of N8 HPC provided and funded by the N8 consortium and EPSRC (Grant No.EP/K000225/1). The Centre is co-ordinated by the Universities of Leeds and Manchester.

7. REFERENCES

[1] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[2] J. F. Miller, editor. *Cartesian Genetic Programming*. Natural Computing Series. Springer, 2011.

[3] Gabriela Ochoa, Matthew Hyde, Tim Curtois, Jose A Vazquez-Rodriguez, James Walker, Michel Gendreau, Graham Kendall, Barry McCollum, Andrew J Parkes, Sanja Petrovic, et al. Hyflex: A benchmark framework for cross-domain heuristic search. In *Evolutionary Computation in Combinatorial Optimization*, pages 136–147. Springer, 2012.