An adaptive neuroevolution-based hyperheuristic

Etor Arza
BCAM - Basque Center for Applied Mathematics
Bilbao, Spain - Basque Country
earza@bcamath.org

Josu Ceberio
University of the Basque Country (UPV/EHU)
Donostia-San Sebastian, Spain
josu.ceberio@ehu.es

Aritz Pérez
BCAM - Basque Center for Applied Mathematics
Bilbao, Spain - Basque Country
aperez@bcamath.org

Ekhiñe Irurozki
BCAM - Basque Center for Applied Mathematics
Bilbao, Spain - Basque Country
eirurozki@bcamath.org

ABSTRACT

According to the No-Free-Lunch theorem, an algorithm that performs efficiently on any type of problem does not exist. In this sense, algorithms that exploit problem-specific knowledge usually outperform more generic approaches, at the cost of a more complex design and parameter tuning process. Trying to combine the best of both worlds, the field of hyperheuristics investigates the automatized generation and hybridization of heuristic algorithms. In this paper, we propose a neuroevolution-based hyperheuristic approach. Particularly, we develop a population-based hyperheuristic algorithm that first trains a neural network on an instance of a problem and then uses the trained neural network to control how and which low-level operators are applied to each of the solutions when optimizing different problem instances. The trained neural network maps the state of the optimization process to the operations to be applied to the solutions in the population at each generation.

CCS CONCEPTS

• **Mathematics of computing** → Combinatorial optimization; • **Computing methodologies** → **Reinforcement learning**; *Neural networks*; *Transfer learning*;

KEYWORDS

neuroevolution, transfer learning, hyperheuristic, optimization

ACM Reference Format:
Etor Arza, Josu Ceberio, Aritz Pérez, and Ekhiñe Irurozki. 2020. An adaptive neuroevolution-based hyperheuristic. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3389937

1 INTRODUCTION

Due to the challenge that many optimization problems represent, a wide range of heuristics have been proposed. These are usually designed by hand, which requires not only problem-specific knowledge but also time to finely tune the parameters to the problem at hand. Much like the design of heuristics, hybridizing different heuristics by trying different heuristic combinations can also be very time consuming if done by hand. As an alternative, some parameter control approaches have been able to effectively select suitable parameters of an algorithm, obtaining a better performance than with the default parameters [2].

In the same trend, hyperheuristics were proposed as a step further in the development of automatized algorithm design. The field of hyperheuristics [1] studies the automatized generation and hybridization of heuristics. Recently, a new research area has emerged in this framework: neural combinatorial optimization (NCO) [3]. NCO studies the use of fixed topology neural networks (NN) designed for sequence-to-sequence learning to define a hyperheuristic for combinatorial optimization problems. Specifically, solutions are constructed from the ground up by iteratively selecting solution components with a previously trained NN heuristic. However, NCO has its own limitations, such as the requirement of a NN structure to be defined by the user before training.

In this paper, we introduce NNs to the field of population-based hyperheuristics. Particularly, we propose a two-stage, population-based, hyperheuristic algorithm. In the first stage, a NN is trained with neuroevolution, which will later guide the hyperheuristic. Then, in the second stage, the hyperheuristic uses the trained NN to select which and how operators are applied to the solutions during the optimization of an instance of a problem. It is worth noting that the operators are chosen on a per solution basis. Once the NN is trained, the hyperheuristic can be applied to other instances without any additional training.

To validate the proposed idea, some preliminary experiments were carried out on a set of Quadratic Assignment Problem [4] (QAP) instances. Specifically, we have focused on analyzing the transferability of the hyperheuristic by showing that the proposed approach adapts to the properties of the instances. The results point out that the NN adapts the behavior of the hyperheuristic algorithm to take advantage of the properties of the problem instances.

2 A NEUROEVOLUTION-BASED HYPERHEURISTIC

The hyperheuristic presented in this paper is a population-based optimization algorithm that uses a NN (called controller) to select the operators to be applied to the solutions in the population. This

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7127-8/20/07.
https://doi.org/10.1145/3377929.3389937

network, based on the state of the optimization, decides which and how operators are applied to each solution during the optimization of a problem instance. Prior to using the hyperheuristic, the NN needs to be trained. In this context, training a NN involves running a neuroevolution algorithm, with the average performance of the hyperheuristic with that NN as the only reward. More specifically, neuroevolution of augmenting topologies (NEAT) has been used[1]. Introduced by Stanley et al., NEAT is a widely used neuroevolution technique able to evolve not only the parameters (weights) of a neural network but also its structure [5].

Once the NN has been trained, the hyperheuristic can be applied to any other problem instance without additional training. Given a trained NN and a problem instance, in the following, we explain how the hyperheuristic optimizes a problem instance. First, a population of q solutions[2] is initialized at random and their objective function values are computed. Next, for each solution σ_i in the population, the information regarding this solution and the state of the optimization process, X_i, is collected by the encoder. Among other relevant features, X_i includes information about the relative objective value of σ_i with respect to the rest of the solutions, the variability in the population, the distance of σ_i to the close solutions and whether σ_i is a local optimum or not (for further details we refer the reader to the online repository). This information X_i is a real valued vector with values in the interval $(-1, 1)$. Once X_i has been obtained, it is fed into the NN that processes this input into another real-valued vector Y_i in the interval $(-1, 1)$. Then, the decoder interprets Y_i by applying the operators specified by Y_i to solution σ_i. It is noteworthy that Y_i also contains information about how to apply the specified operators. Some of the information contained in Y_i includes whether to apply a local search procedure or not, moving towards or away from neighbour solutions or the best solution in the population and the choice of neighborhood system. Once all the solutions σ_i for $i \in \{1, ..., q\}$ have been modified, the objective values of the new solutions are updated. This process is repeated until a stopping criterion is reached, and, upon termination, the best-found solution is reported.

3 EXPERIMENTAL STUDY

The goal of this experimentation is to validate the proposed approach by analyzing the adaptability of the hyperheuristics algorithm to the properties of the instances. Specifically, a study on the transferability[3] of the hyperheuristic is presented. To that end, we trained and tested the hyperheuristic approach in some instances of the Quadratic Assignment Problem (QAP) [4] with different properties. Nine QAP [4] instances of same size ($n = 60$), from three source benchmarks (*Taixxa*, *Taixxb* and *Sko*) where chosen (called A, B and C in this paper).

A sample of the results of this experiment are shown in Figure 1. A lighter color indicates a higher normalized objective function value and, thus, better performance. NNs trained on instances of type A perform well on other instances of the same type, but are unable to perform well on other kinds of instances. In addition, NNs that perform best in each instance class are the ones that were trained in that instance class. The results point out that the best performance is obtained when NNs are trained and tested in instances with the same properties. This suggests that the hyperheuristic is adapting to the properties of the instances to obtain a better performance.

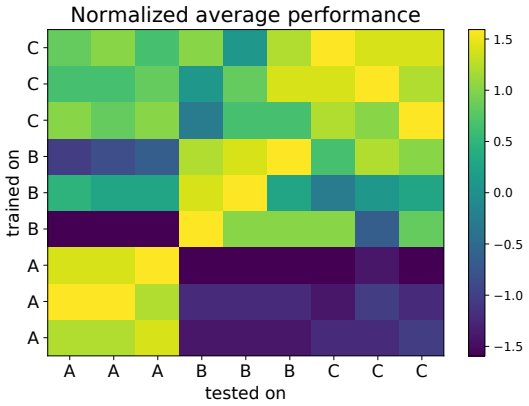

Figure 1: Normalized average objective value for NN trained and tested in QAP instances. Lighter is better.

4 CONCLUSION AND FUTURE WORK

In this paper, a NN-based hyperheuristic is proposed. Specifically, a population-based hyperheuristic is proposed that uses a NN to guide the optimization process. The NN allows the hyperheuristic to adapt the use of operators to the particularities of the problem instances during training. Using the QAP as a case study, the experimentation revealed that this new hyperheuristic trains optimization strategies tailored to the properties of the training instances. Future work will be focused on i) extending the number of combinatorial optimization problems, ii) adaptation to continuous problems and iii) studying how differently the hyperheuristic behaves depending on the type of instance that was used to train it.

Acknowledgments

This work is partially supported by the Basque Government through the BERC 2014-2017 and the ELKARTEK programs, by the Spanish Ministry of Economy and Competitiveness, through BCAM Severo Ochoa excellence accreditation SEV-2017-0718, and through the project TIN2016-78365R and by AEI/FEDER UE, through TIN2017-82626-R, and by the Research Groups 2013-2018 (IT-609-13).

REFERENCES

[1] Edmund K Burke and Michel et al. Gendreau. 2013. Hyper-Heuristics: A Survey of the State of the Art. *Journal of the Operational Research Society* (2013).
[2] Benjamin Doerr and Carola Doerr. 2020. Theory of Parameter Control for Discrete Black-Box Optimization: Provable Performance Gains Through Dynamic Parameter Choices. In *Theory of Evolutionary Computation*. Springer International.
[3] Bello et al. 2016. Neural Combinatorial Optimization with Reinforcement Learning. *arXiv preprint* (2016).
[4] Tjalling C. Koopmans and Martin Beckmann. 1957. Assignment Problems and the Location of Economic Activities. *Econometrica* 1 (1957).
[5] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* (2002).

[1] Our implementation, available in our GitHub repo github.com/EtorArza/GECCO2020, is based on accneat. Accneat is a fork of Stanley et al.'s implementation [5] with some improvements, such as delete mutations and speed improvements, available at github.com/sean-dougherty/accneat.
[2] In this work, the QAP is used as a case study, and thus the solutions are permutations.
[3] We refer to "transferability" as the difference in the performance of a hyperheuristic trained in a certain type of instance and tested in another type of instance.