Contribution ID: **102** Type: **Poster (sponsored)**

A Multi-Space Search Algorithm for Discrete Optimisation and Prediction Problems

Many of the machine learning techniques that are used today involve some form of search within an abstract space. For example, genetic algorithms search within a space consisting of solutions to the problem at hand, whereas genetic programming searches within a program space which contains programs that output a solution to the problem when executed. In addition to searching within a particular space, different techniques search in different ways. For example, evolutionary algorithms search by taking an analogy from the theory of evolution, whereas ant colony optimisation searches by taking an analogy from how ants in a colony behave.

One field which is gaining in popularity is that of so-called hyper-heuristics, an extension to the idea of heuristics. Although no precise definition of a heuristic exists, a heuristic can generally be thought of as a ”rule of thumb” which employs domain knowledge in order to guide the solution process towards finding a quality solution [1]. The idea behind hyper-heuristics is to search within a space of heuristics, in order to find a method of solving the problem, as opposed to searching through solutions to the problem. As stated in Burke at al. [2], there exist two distinct search spaces in hyper-heuristic methodologies, namely the heuristic and solution spaces. Each point in the heuristic space can be mapped to a corresponding point within the solution space, however, the same is not true in reverse (ie: not every point in the solution space has a corresponding point in the heuristic space) [2]. In addition, it is not guaranteed that neighbouring points within the heuristic space will be neighbours within the solution space [2]. The advantage of this is that the hyper-heuristic methodology is able to effectively explore a larger area of the solution space within a shorter time (when compared to methods which search the solution space directly). However, it is plausible to have cases where the optimal solutions do not have a corresponding point within the heuristic space. In such cases, a method which searches solely in the heuristic space will never find an optimal solution.

In this research we propose a methodology which simultaneously searches within both of the two spaces, thereby combining the benefits associated with searching in each space. Burke et al. [2] have implemented a similar approach for the exam and course timetabling problems, where a greedy local search is performed on complete solutions which were created by heuristic combinations. We extend this idea by optimising when the algorithm searches in each space. This multi-space search methodology involves a large amount of computational effort as it requires search in two spaces and hence both spaces will be populated during the search process. Additionally, our multi-space search methodology was implemented in the form of a genetic algorithm (GA), which inherently have high runtimes. The combination of these factors meant that it was necessary to make extensive use of high performance computing.

The implemented GA, which used the generational control model and tournament selection, was created with the aid of the
EvoHyp
toolkit [3]. The distribution of the algorithm was done by identifying
key points in the algorithm at which ”bottle-necks” occur, and attempting to perform multiple tasks concurrently at these points. Two such ”bottle-necks” identified were the population generation and the fitness evaluation of the members of the population. The distribution was achieved by creating, and evaluating, sub-populations on each of a specified number of processing cores. Additionally, due to the stochastic nature of the GA it is necessary to average over multiple runs in order to minimise the stochastic noise in the results. Our implementation made further use of high performance computing resources by distributing these runs across multiple processing cores. The data was also distributed. Hence, our simulations made extensive use of high performance computing by distributing the data across multiple processing cores, where for each dataset the required runs were further distributed across

cores. Furthermore, for each run executed, the algorithm was also distributed. For example, with a single dataset, requiring 3 runs and distributing the algorithm across 5 cores, a simulation would use 15 processing cores as opposed to just 1.
We tested our multi-space search framework using two problem domains, namely the one-dimensional bin packing problem (BPP) and the weather prediction problem. The BPP involves packing a list of items of a given size into bins of a predetermined capacity, such that a minimum number of bins is used. This problem is of particular interest to industry as it relates to tasks such as reducing materials waste (when cutting components from a larger sheet), improving packing efficiency of stock, etc. The weather prediction problem involves using various past environmental conditions (such as temperature, humidity, etc.) to predict their future values as well as future weather conditions (eg: thunderstorm, hurricane, etc.). For both problem domains, the results from the optimised multi-space search are compared with those from an un-optimised multi-space search (as implemented in Burke et al. [2]), as well as with those from a standard Evolutionary Algorithm Hyper-Heuristic (EAHH). The Scholl [4] benchmark datasets were used for the BPP. Preliminary results suggest that performing search across multiple spaces is more effective than searching within a single space, and optimising when each space is searched is the most effective. For future work we intend to extend our algorithm to work across more than two search spaces.

References
[1] E.K. Burke and G. Kendall.
Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques
. SpringerLink : B¨ucher. Springer US, 2013.
[2] Edmund K. Burke, Barry McCollum, Amnon Meisels, Sanja Petrovic, and Rong Qu. A graph-based hyper-heuristic for educational timetabling problems.
European Journal of Operational Research
,
176(1):177–192, jan 2007.
[3] N. Pillay and D. Beckedahl. EvoHyp - a Java toolkit for evolutionary algorithm hyper-heuristics. In 2017 IEEE Congress on Evolutionary Computation (CEC)
, pages 2706–2713, June 2017.
[4] Armin Scholl, Robert Klein, and Christian J¨urgens. Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem.
Computers & Operations Research
, 24(7):627 – 645, 1997.

Presenter Biography

Primary author: BECKEDAHL, Derrick (University of Pretoria)

Co-author: PILLAY, Nelishia (University of Pretoria)

Presenter: BECKEDAHL, Derrick (University of Pretoria)

Session Classification: Poster session

Track Classification: Cognitive Computing & Machine Learning