Hyper-heuristics learning a varying set of low-level heuristics

M.Mısır*♮, K.Verbeeck*♮, G.Vanden Berghe*♮
*CODeS, KaHo Sint-Lieven
{mustafa.misir,katja.verbeeck,greet.vandenberghe}@kahosl.be

P. De Causmaecker♮
♮CODeS, Department of Computer Science,
K.U.Leuven Campus Kortrijk
patrick.decausmaecker@kuleuven-kortrijk.be

The main motivation behind using hyper-heuristics is related to providing generality for solving different combinatorial optimisation problems. Hyper-heuristics perform on a higher level than traditional search and optimisation strategies. They operate on a set of solution approaches (i.e. low-level heuristics) rather than on the set of solutions directly. The performance of heuristics can vary from a problem(-instance) to a problem(-instance). They may even behave differently in various search regions of one problem. Therefore, using a management mechanism on top of a number of search algorithms can help to find the most appropriate heuristics to apply. This kind of management can be handled by *choice hyper-heuristics*. A simple choice hyper-heuristic consists of a *heuristic selection* mechanism and a *move acceptance* mechanism. While a heuristic selection mechanism chooses heuristics for applying to a solution(s) at hand, a move acceptance mechanism concludes whether the new solution(s) is good enough to accept.

In this study, we focus on the heuristic selection part. We propose a method for determining the set of best performing heuristics that should be used during different phases of a search. This subset is formed based on the relative improvement per execution time of each heuristic. At the end of each phase, heuristics are ranked according to some quality related values (quality index). That is, the quality index of the best performing heuristic is set to n and for the worst heuristic, this value is set to 1. The index value of heuristics that do not provide any improvement are automatically set to 1. All heuristics which have a quality index value lower than average are excluded from the search for the next phase. In addition, when the set of best performing heuristics becomes too small, all the excluded heuristics are entered in the heuristic set again, and the algorithm repeats itself.

We apply our new hyper-heuristic approach to a set of home care scheduling problem instances.