Evolving Bin Packing Heuristics with Genetic Programming

E.K. Burke, M.R. Hyde, and G. Kendall

School of Computer Science and Information Technology
The University of Nottingham
Nottingham, NG8 1BB
United Kingdom (UK)
mvh@cs.nott.ac.uk
http://cs.nott.ac.uk/∼mvh

Abstract. The bin-packing problem is a well known NP-Hard optimisation problem, and, over the years, many heuristics have been developed to generate good quality solutions. This paper outlines a genetic programming system which evolves a heuristic that decides whether to put a piece in a bin when presented with the sum of the pieces already in the bin and the size of the piece that is about to be packed. This heuristic operates in a fixed framework that iterates through the open bins, applying the heuristic to each one, before deciding which bin to use. The best evolved programs emulate the functionality of the human designed 'first-fit' heuristic. Thus, the contribution of this paper is to demonstrate that genetic programming can be employed to automatically evolve bin packing heuristics which are the same as high quality heuristics which have been designed by humans.

1 Introduction

The aim of this work is to explore how a genetic programming system might evolve good heuristics that can pack bins. The goal is to automatically generate good heuristics which can operate over a range of instances and, perhaps more importantly, do not require human intervention or parameter tuning. The aim is therefore to show that a genetic programming system is capable of generating heuristics which have the functionality of human-designed heuristics. In this work we use a basic genetic programming system which does not incorporate any of the more advanced techniques such as re-usable code, loops, or memory, all of which are explained in [1] and [2] for the interested reader.

1.1 Genetic Programming

Genetic programming [3] evolves a population of computer programs which are represented as tree structures. After each program (or individual) has been executed, its performance is assessed and a fitness rating is given to it. The genetic operators of crossover and reproduction are applied to individuals in the population in proportion to their relative fitness.

T.P. Runarsson et al. (Eds.): PPSN IX, LNCS 4193, pp. 860–869, 2006.

1.2 Hyper-heuristics

One of the aims of a hyper-heuristic [4,5] is to "raise the level of generality at which optimisation systems can operate" [5]. To this end, hyper-heuristics are heuristics which choose "between a set of low-level heuristics, using some learning mechanism" [6].

Of course, the No Free Lunch theorem [7,8] shows that all search algorithms have the same average performance on all problems defined on a given finite search space. However, it is important to recognise that this theorem is *not* saying that it is not possible to build search methodologies which are *more* general than is possible today. Indeed, research into hyper-heuristics is motivated by the assertion that in many real world problem solving environments, there are users who are interested in *"good-enough soon-enough cheap-enough"* solutions to their optimisation problems [5].

Some examples of hyper-heuristic methods are briefly explained here. Two hyper-heuristic methods have been tested on the one-dimensional bin-packing problem, a learning classifier system [9] and a genetic algorithm [10]. In [11], an ant algorithm hyper-heuristic chooses a sequence of low-level heuristics for the project presentation scheduling problem. A tabu search hyper-heuristic is applied in [12] to a nurse scheduling problem and a university course timetabling problem. A choice function has also been employed as a hyper-heuristic, to rank the low-level heuristics and choose the best [13]. A multi-objective hyper-heuristic is applied to space allocation and timetabling in [14]. A graph based hyper-heuristic is used for timetabling in [15]. Case based heuristic selection is applied on a timetabling problem in [16]. Finally, in [17], a simulated annealing hyper-heuristic is used to determine shipper sizes.

1.3 Using Genetic Programming as a Hyper-heuristic

This paper invesigates the role of genetic programming as a hyper-heuristic. The genetic programming system chooses between a set of low level building blocks to construct a heuristic which performs well in the environment given to it. In this case the building blocks are the function and terminal set shown in table 2, and the environment is the data set given to the system.

In previous work (e.g. [12]), hyper-heuristics have had their low-level heuristics given to them by the human programmer, and so the number and quality of heuristics that the hyper-heuristic has available is limited to those which a human can provide. The aim of this research is to show that even more of the decision process can be automated. For example, the human programmer would normally choose the low-level heuristics, from the space of all heuristics. This paper aims to show that by using a genetic programming system as a hyper-heuristic, *any* heuristic can be chosen from the space of all heuristics that can be constructed with the function and terminal set. The human programmer therefore need only supply the *potential* components of a heuristic.