Improving metaheuristic performance by evolving a variable fitness function

Improving Metaheuristic Performance by Evolving a Variable Fitness Function*

Keshav Dahal[1], Stephen Remde[1], Peter Cowling[1],and Nic Colledge[1]

[1] MOSAIC Research Group, University of Bradford, Bradford, BD7 1DP, United Kingdom
{s.m.remde, p.i.cowling, k.p.dahal, n.j.colledge}@bradford.ac.uk
http://mosaic.ac/

Abstract. In this paper we study a complex real world workforce scheduling problem. We apply constructive search and variable neighbourhood search (VNS) metaheuristics and enhance these methods by using a variable fitness function. The variable fitness function (VFF) uses an evolutionary approach to evolve weights for each of the (multiple) objectives. The variable fitness function can potentially enhance any search based optimisation heuristic where multiple objectives can be defined through evolutionary changes in the search direction. We show that the VFF significantly improves performance of constructive and VNS approaches on training problems, and "learn" problem features which enhance the performance on unseen test problem instances.

Keywords: Variable Fitness Function, Evolution, Heuristic, Meta-heuristic.

1 Introduction

Search gets stuck when local optima are reached, and there are no better neighboring solutions, but the solution is not globally optimal. While the global fitness function is ideally suited to an approach guaranteed to find an optimal solution, it is not adequate in assessing the fitness of a local move. Many metaheuristics allow escape from these local optima however they may ultimately fail at a higher level because of the nature of the global fitness function.

The variable fitness function seeks to tackle this problem by redefining the fitness function so it may change over the course of the search. The result is that the local fitness function is different from the global fitness function and can be more effective than the global fitness function to assess local moves. [1] shows the variable fitness function's effectiveness at enhancing local search heuristics and in this paper we attempt to show its ability to enhance a metaheuristics and to learn reusable information to guide the search of a difficult optimization problem. The problem we study is a complex real world workforce scheduling problem which contains many scheduling problems from the literature as subproblems. Like many other real world

* This work was funded by EPSRC and @Road Ltd, a Trimble Company under an EPSRC CASE studentship, which was made available through and facilitated by the Smith Institute for Industrial Mathematics and System Engineering

problems it has many features that are hard to understand and model, and objectives that are non-linear in nature. This can make it hard for a person to define a global fitness function, let alone one to describe how to assess the quality of local moves. We aim to show the variable fitness function's ability to enhance the search when we solve this workforce scheduling problem using constructive search and Variable Neighborhood Search (VNS).

In the next section, related work is discussed. In section 3 the variable fitness function is defined and section 4 describes the problem. In section 5 the computational experiments are presented and the results analysed. Finally, section 6 will draw conclusions.

2 Related Work

The variable fitness function (VFF) is a new search enhancement technique that can be used to enhance any search based optimization heuristic provided that a) the problem is multi-objective (or multiple objectives can be defined in some way) and b) we have CPU time available to use this process offline (although the resulting VFF can be used very quickly online). First presented in [1], the variable fitness function provides a simple scheme for encoding a piecewise linear function into a genetic algorithm and a method for evolving these functions. The variable fitness functions are then used to determine the local fitness function at each step in the local search.

Guided Local Search [2] also modifies the fitness function to change the direction when a local optimum has been found. Features of a solution are identified and penalties for solutions exhibiting these features are increased when the solution is stuck in a local optimum. A feature which occurs in a local optimum has its penalty score increased slightly, and these penalties are used to modify the fitness function, attempting to force the search to move in another direction. The primary differences between the VFF and Guided Local Search approaches are in their approaches to modifying the fitness function (evolutionary versus reinforcement learning), the fitness function objects that are being tuned (objectives versus features) and, most importantly, the ability of the Variable Fitness Function to be applied with no CPU time overhead for unseen test instances.

The problem we study is based on a mobile workforce scheduling problem presented in [3,4]. It is a complex real workforce scheduling problem identified by @Road Ltd. and shares many complexities found in various other scheduling problems such as the Resource Constrained Project Scheduling Problem (RCPSP) and its variants [5], Job Shop Scheduling Problem (JSSP) [6] and its variants, along with other problems such as Vehicle Routing [7] and the Traveling Salesman Problem [8]. In [3], the problem's multi objective nature was used to show the trade off between diversity and solution quality when using multi objective genetic algorithm compared to a genetic algorithm using weighted sum objective functions. In [4] the problem's complexity was used to show that breaking the problem down into a very large number of smaller parts and then using another method to decide which of these smaller parts to solve, is a very effective way of solving large complex problems.

[4] uses reduced Variable Neighborhood Search (rVNS) [9] amongst other heuristics. rVNS is a faster form of Variable Neighbourhood Search. Variable Neighbourhood Search (VNS) [10] is based on the idea of systematically changing the neighbourhood of a local search algorithm. Variable Neighbourhood Search enhances local search using a variety of neighbourhoods to "kick" the search into a new position after it reaches a local optimum. Several variants of VNS exist as extensions to the VNS framework [9] which have been shown to work well on various optimisation problems. In our experience, VNS has the advantage for complex, real-world problems, of requiring limited additional effort, once a basic local search framework is established.

3 The Variable Fitness Function

The Variable Fitness Function [1] describes how the weights of a weighted sum fitness function change over the iterations of a search process. The variable fitness function is piecewise linear, describing the relative importance of objectives at each iteration. There are two variations: the standard variable fitness function fixes the number of discontinuities and the number of iterations between them and the adaptive variable fitness function allows the points of discontinuity to evolve along with the variable fitness function objective weights (Figure 1). Work so far provides evidence that the adaptive version is more effective than the fixed version, as more complex functions can be evolved [1].

For the adaptive variable fitness function, we define a set of weights $\{W_{b,a}\}$ where a indexes the weight set ($a=1...A$) and b indexes the objective ($b=1...B$). We define I_a, the number of iterations between the weight sets. The variable fitness function is now defined as:

$$f(s,i) = \sum_{b=1}^{B} O_b(s) W_b(i)$$

where s is the solution to be evaluated and i is the iteration, $O_b(s)$ is the value of objective b for solution s and

$$W_b(i) = W_{b,c} + \frac{i-j}{k-j}(W_{b,c+1} - W_{b,c})$$

where iteration i occurs in the range from weight set c (starting at iteration j) to weight set $c+1$ (starting at iteration k) i.e. the linear interpolation of the weight of objective b for iteration i.

Figure 1 shows an example adaptive variable fitness function. This describes how the weights change over the iterations, for example, that the weight of objective 1 (W1) starts off at 2 (hence the objective is to be maximized) and then after iteration 200 its importance starts to decrease and objective 3 (that has weight W3) is to be minimized, and its importance is higher at the start and end of the search process.

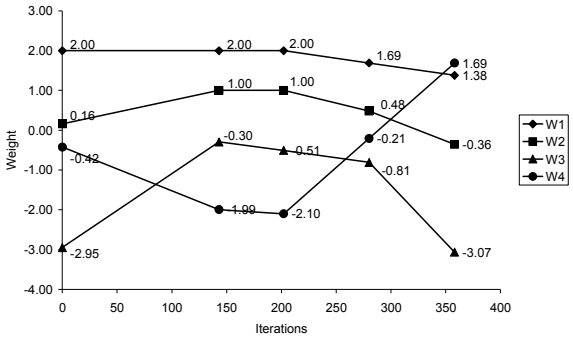

Figure 1. An example adaptive variable fitness function. (the number of iterations between the weight sets and the number of weight sets may vary)

3.1 Evolution

Little work has been done in encoding piecewise linear functions such as these into chromosomes. [11] uses a complex encoding for polynomial expressions. The encoding is used to optimise a curve to fit a function described by a set of data points (and is not an appropriate method for the VFF). The evolution here is similar to work done on tuning of parameters for another algorithm using genetic algorithms [12].

When optimizing the weights of the variable fitness function, each weight in the variable fitness function appears as a gene in a GA chromosome. When the adaptive variable fitness function is used, the iterations between the weight sets are also included. Figure 2 shows how the weight sets are mapped to the genes of a chromosome.

$W_{1,1}$	…	$W_{1,B}$	I_1	$W_{1,1}$	…	$W_{1,B}$	I_1	…	$W_{A,1}$	…	$W_{A,B}$

Figure 2. Mapping the weights to a chromosome for an adaptive variable fitness function.

A modified version of 1 point crossover [13] will be used. It works the same way as normal 1-point crossover but the crossover point may only be on a weight set boundary. This method will keep mutually compatible weight sets together. The thick lines in Figure 2 show these crossover points. Each gene will have a chance to be mutated with a probability of p_{mut}, the mutation rate. Mutation will simply mutate the value of the gene by a random variable normally distributed around 0 and with a standard deviation as predefined for each weight. Hence $W_{a,b}$ is mutated by a value from the normal distribution $N(0, V_b)$ with probability p_{mut}. Where V_b is the standard deviation of mutation associated with objective b and p_{mut} is the probability of mutation which is the same for all alleles. This is similar to work done on mutation of artificial neural network weights evolved using GAs [14] where the network weight is mutated by a random number selected from a normal distribution.

The initial population of variable fitness functions is generated randomly where $W_{a,b}$ is picked uniformly at random out of the interval $[L_b, U_b]$ for objective b. There is a p_{adapt} probability that the chromosome will change length. If a chromosome is to change length there is an equal probability it will either shrink or grow by one weight set. If it is to shrink, a random weight set is chosen and removed from the chromosome. If it is to grow, a new weight set is inserted between two randomly chosen adjacent weight sets. The inserted weight set does not change the shape of the variable fitness function as it is inserted exactly half way between the two adjacent weight sets and has weight values that are the mean of the bordering weight sets. The new weight set is then mutated. Lastly, the I_a genes also have a p_{mut} probability of being mutated using the same method as the $W_{a,b}$ genes. This gives the chromosomes a chance to get more and less complex and to also expand to more or less iterations.

In our experiments, L_b and U_b will be -1 and 1 respectively (since objective values are normalized), $V_b = 0.05(U_b - L_b)$, and $p_{mut} = p_{adapt} = 0.05$ for all objectives b. These are known good values from previous work [1] and this previous work also shows that the sensitivity to parameters is low.

4 The Case Study Problem and Solution Heuristics

The problem we study is based on the workforce scheduling problem in [3]. The problem consists of assigning resources and time slots to geographically dispersed tasks. Tasks require various skills and resources possess these skills at different competencies. Resources are mobile and must travel between tasks and to and from their "home" location at the end and start of the day. Tasks have a priority associated with them which indicates their urgency or the reward for completing the task.

In the problem instances we study, 10 resources possess between 1 and 5 skills of which there are various bottlenecks in the availability. The resources travel at varying speeds. There are 300 tasks requiring between 0.5 and 1 hours to complete to be scheduled over 3 days and each has a 4 hour time window in which it must be completed. A task requires a resource to possess a certain skill, of which some skills are in more demand than others. Tasks are to be completed as early in the 4 hour time window as possible. Tasks have precedence constraints such that some task may not be started before another has completed. A chain of tasks is a subgraph of the precedence digraph of maximum indegree 1 where the indegree (outdegree) of a task in the subgraph is one if the indegree (outdegree) in the precedence digraph is greater than zero.

For this is a real world complex problem there are many objectives. Table 1 lists some of the principal objectives we have identified. The global fitness function is defined as *f = 5 (Scheduled High) + 2 (Scheduled Low) +(Complete Chains) – 0.1 (Overrun)*, following reflection with our industrial collaborator. We use a constructive heuristic, *CON*, to build an initial schedule then an improvement metaheuristic, *IMP*, to improve it.

Table 1. Objectives used for the workforce scheduling problem.

Objective	Description
Scheduled High	The number of high priority tasks scheduled.
Scheduled Low	The number of low priority tasks scheduled.
Complete Chains	The number of task chains that have been completed.
Travel Distance	The total distance traveled by the resources.
Travel Time	The total time spent traveling by the resources
Overrun	The total number of hours the task have overrun.

For the improvement heuristic we have decided to use Variable Neighbourhood Search (VNS). VNS is relatively simple to implement and we have seen that this kind of method can work well for scheduling problems [1]. We use a local search heuristic where we define the neighborhood as schedules which result from optimally reinserting a task, i.e. placing a task optimally in the schedule in terms of the current (variable) fitness function. If the task is not yet scheduled, this means allocating the resources and time to it that yield the best improvement in fitness. If the task is already scheduled, this may mean moving the task in time, allocating new resources or a combination of the two (Figure 3). At each iteration of the local search, the entire neighbourhood is sampled and the best solution accepted. When the local search of the VNS reaches a local optimum, the search is kicked into a new area of the search space. We define these kicks as removing between 1 and 4 tasks and all dependent tasks. We remove dependent tasks so that precedence constraints are not broken.

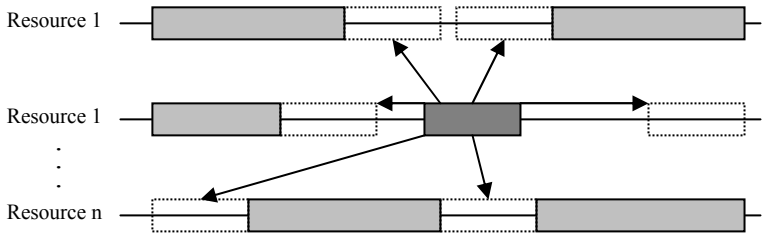

Figure 4. Task reinsertion. The task is moved to the resource and time in the schedule which provides the best change in fitness according to the variable fitness function. The light grey boxes represent other tasks, the dark grey is the task being optimized and the dotted boxed are the positions being considered.

The construction method, *CON*, uses the local search operator of the VNS and terminates when a local optimum is found. The improvement metaheuristic, *IMP*, has a stopping criterion of 10,000 iterations. Table 2 lists the methods we will try. We can see the first two are normal heuristics and the last three are enhanced using VFF.

Table 2. Various methods to be used and their VFF enhance versions.

Heuristics	Description
CON	Construction heuristic using the global fitness function.
CON + IMP	Construction heuristic and improvement heuristic using the global fitness function.
VFF(CON)	Construction heuristic using a variable fitness function.
VFF(CON + IMP)	Construction heuristic and improvement heuristic using a variable fitness function.
CON + VFF(IMP)	Construction heuristic using the global fitness function and improvement heuristic using a variable fitness function.

5 Computational Experiments

The five methods are used ten times on each of the five training instances and averages taken to reduce the effect of randomness. These five training instances are chosen to contain variations that a workforce would see on a day to day basis. By using multiple problem instances to evolve variable fitness functions we are trying to ensure that variable fitness functions learn characteristics of the problem through learning specific problem instances. For the methods enhanced with the variable fitness function, a test set of five problems instances will be solved using VFFs which were evolved using the training instances. Good performance on the test data will imply that a lot of CPU time could be used to train a "general purpose" variable fitness function, then that variable fitness function could be used very quickly in "real time".

The methods requiring the evolution of a variable fitness function will be given 50 generations with a population size of 10 (equivalent to 500 evaluations). Methods without a variable fitness function will also be given 500 evaluations and the best one taken to give them the same amount of CPU time. The *CON* heuristic takes approximately 30 seconds to construct the five schedules and the *IMP* heuristic takes approximately 25 minutes on a 3.0 GHz PC. As these experiments will take over 260 CPU days to complete they will be run in parallel on approximately 95 computers.

Table 3 shows the results of the individual methods and their standard deviations and Figure 5 graphs these with 90% confidence intervals. From the results we can see that the variable fitness functions were indeed able to enhance the standard methods significantly in all cases. We see very large variations in fitness when the variable fitness function is used on the constructive part of the search (*VFF(CON)* and *VFF(CON + IMP)*). Further investigation leads us to believe that this is because when the variable fitness function affects the constructive part of the search, it has the possibility to move a great distance in the search space from the "normal" constructive algorithm. The best variable fitness function enhancement for the *CON + IMP* method was to just enhance the improvement part. The variation of *CON + IMP* is extremely low and the solutions that it has found are far from optimal. This may be because the kick method of the VNS we have chosen was not sufficiently disruptive.

Table 3. Average fitness and standard deviation of ten runs of each method assessed using the global fitness function.

Method	Average Fitness	Standard Deviation
CON	3189.6774	N/A
VFF(CON)	3308.0253	150.3236
CON + IMP	3617.4517	8.4949
VFF(CON + IMP)	3689.9001	111.2555
CON + VFF(IMP)	3770.2434	35.4282

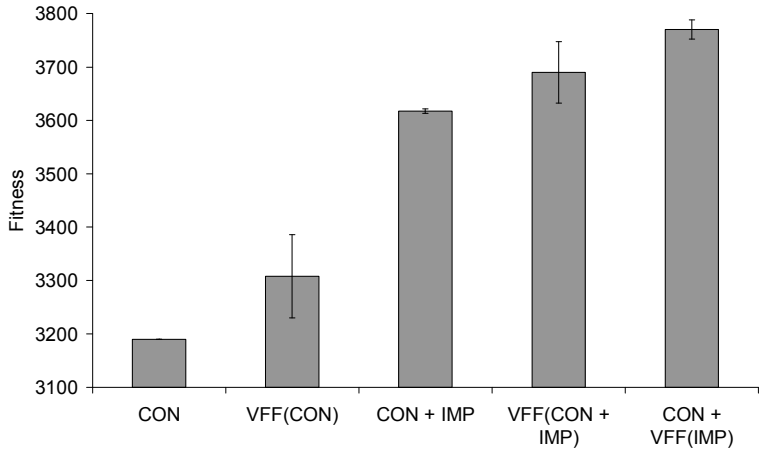

Figure 4. Graph of the results in Table 3 with 90% confidence intervals.

Figure 5 shows the breakdown of the individual objectives and Table 4 shows the difference in the average objective measures the enhanced methods have produced. This chart and table show that the VFF approach has found a way to obtain improvements to high priority objectives at the expense of low priority ones.

In all of the cases where the method is enhanced by the variable fitness function, the number of scheduled tasks, both low priority and high priority, has increased. This intuitively makes sense as these are the highest weighted objectives in the global fitness function. Travel time was decreased in both the cases where the variable fitness function was used to enhance the metaheuristic. The increase in travel time and other penalty objectives for the CON approach is not surprising as CON has no way to optimize these objectives by reinserting. Travel time is not included in the global fitness function, however, it would appear that when task reinsertion is permitted, the VFFs have learnt that less time spent traveling means more time can be spent doing tasks. In all cases, overrun increased, indicating that tasks were not scheduled as close to their start time as possible. This may be because tasks were shifted later in time so other tasks could be completed before them, enabling preceding tasks to also be completed.

Table 4. Average change in objectives as a result of variable fitness function enhancement.

Base Method	CON	CON + IMP	
Improvement Using	VFF(CON)	VFF(CON + IMP)	CON + VFF(IMP)
Scheduled High (max)	20.20	4.30	17.10
Scheduled Low (max)	40.50	37.10	43.20
Travel Distance (min)	13.79	-24.03	-67.69
Travel Time (min)	14.91	-24.85	-64.34
Overrun (min)	603.52	220.51	203.08
Complete Chains (max)	-3.30	-1.20	1.20

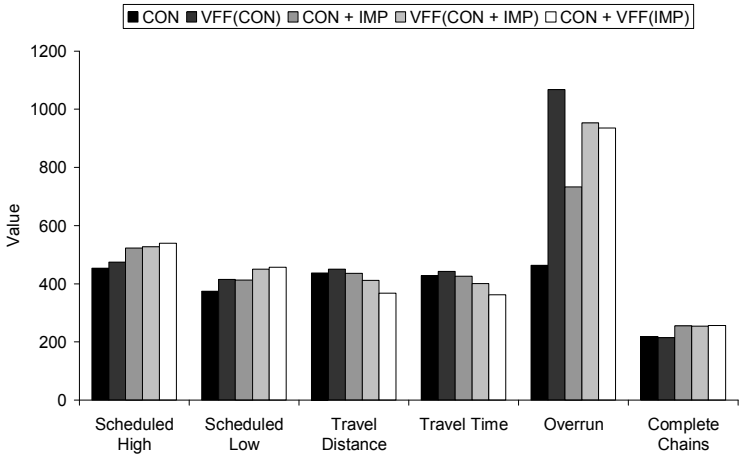

Figure 5. Individual objective break down for each method.

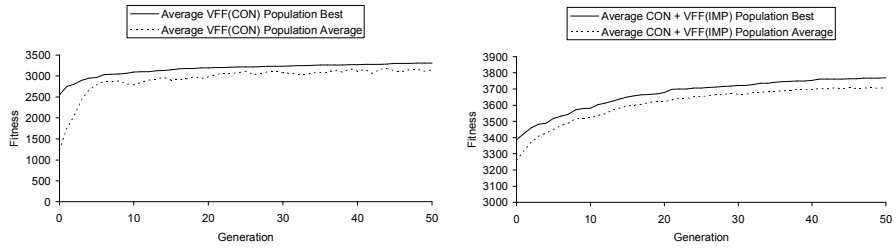

Figure 6. Average population fitness and best of the population's fitness at each generation showing the evolution for *VFF (CON)* and *CON + VFF(IMP)* methods.

Figure 6 shows the evolution process in action. Not only do these graphs show that the evolution process is working, and that the populations are evolving, but it shows the difference between randomly generated variable fitness functions (those in the initial population at generation 0) and evolved ones (those in the final population at generation 50). The plot showing the evolution of *VFF(CON)* method shows a greater

increase in fitness from random variable fitness functions to evolved variable fitness functions than that of *CON + VFF(IMP)* (note the difference in "fitness" scale between the graphs). This is because the *CON + IMP* is a better method than *CON*, and hence there is less room for improvement.

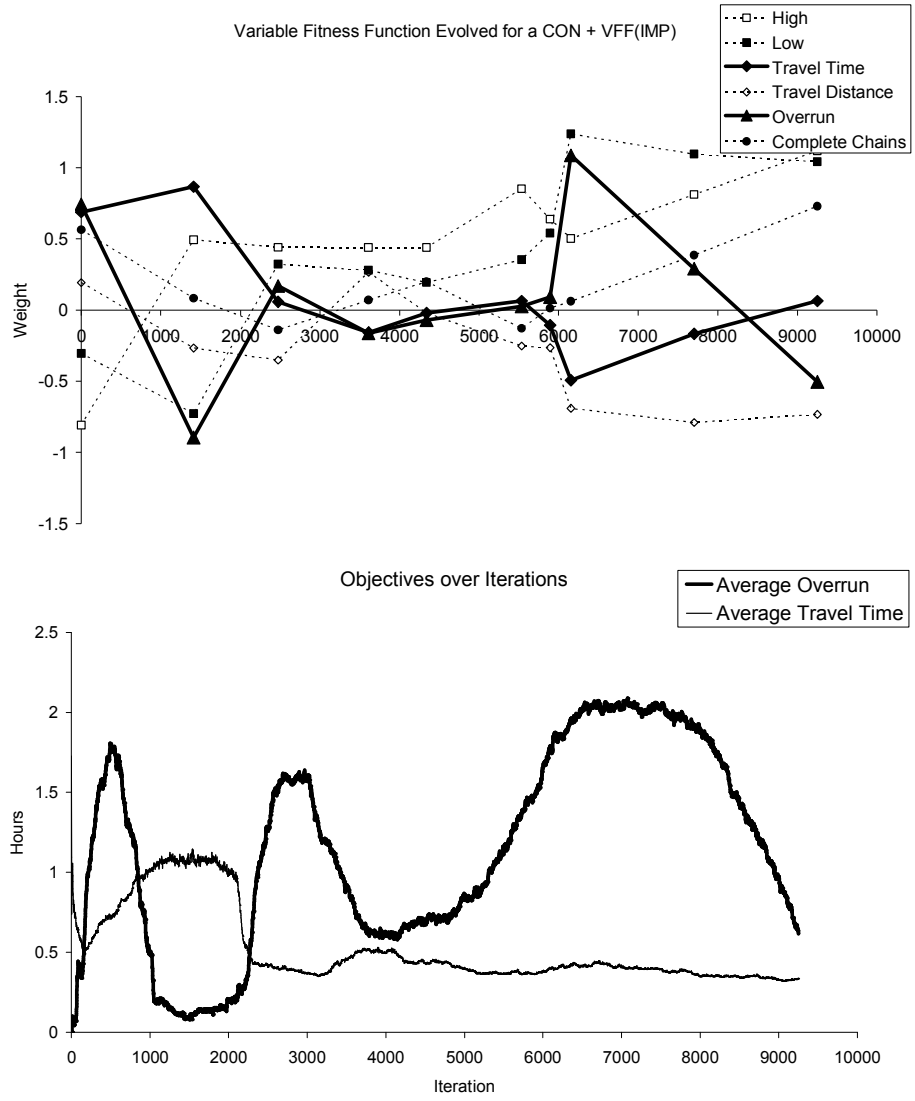

Figure 7. A selected evolved VFF shown above and a plot below showing how two selected objective measures change over the course of a search.

Figure 7 shows an example of how the variable fitness function is working. The top plot shows a typical evolved variable fitness function from the population and how the objective weights change over the iterations. Highlighted are the Travel Time

and Overrun objective weights. Plotted below are the objective values obtained at each iteration from a single run using this variable fitness function. A quite obvious correlation can be seen between the weight of overrun and the average overrun observed. When the weight is positive, overrun increases and when the weight is negative it decreases. This variable fitness function has in fact learnt a type of right-left shift heuristic [15], which is frequently used in schedule repair.

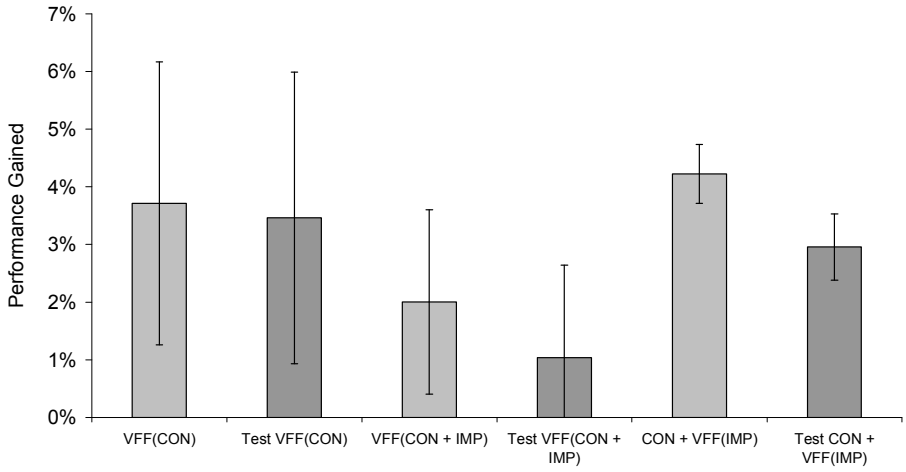

Figure 7. Average method performance gained using variable fitness function on test data compared to training data.

Figure 7 shows the improvement gained in the global fitness function from using the variable fitness function enhanced methods over the standard methods, for both the training data and the test data. Note that for test data instances, the amount of CPU time for the VFF and standard approaches are the same. As seen in the chart, the variable fitness functions enhanced methods are still significantly better than their standard versions on the test data (with the exception of the *VFF(CON + IMP)* method whose 90% confidence interval takes it below 0%). This is a good indication that variable fitness functions trained for the *VFF(CON)* and *CON + VFF(IMP*) could be reused on different problem instances with good performance, and that they have "learned" generalisable information about the problem as well as specific information about the training instances.

5 Conclusions

In this paper we have demonstrated the application of an evolutionary variable fitness function to a constructive heuristic and a metaheuristic for a complex, real-world workforce scheduling problem. We have shown that statistically significant increases in heuristic and metaheuristic performance can be gained by using the variable fitness

function. We have also seen that evolution plays a key role in getting these gains. To show the reusability of the evolved variable fitness functions they were used on another set of problem instances and showed gains of nearly equal magnitude. This is a strong indicator that a variable fitness function could be evolved offline and then the evolved variable fitness function be used in a real time situation. Arguably, the variable fitness function can be used for any optimization problem where multiple objectives can be defined. In future work we will investigate further how and why the VFF approach works its potential as a general problem-solving approach.

References

[1] Remde, S., Cowling, P., Dahal, K. and Colledge, N.: Evolution of Fitness Functions to Improve Heuristic Performance. Proceedings of Learning and Intelligent Optimization (LION) II, LNCS, Springer (2008),

[2] Tsang, E. and Voudouris, C.: Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. Operations Research Letters 20 (3): 119-127 (1997)

[3] Cowling, P., Colledge, N., Dahal, K. and Remde, S.: The Trade Off between Diversity and Quality for Multi-objective Workforce Scheduling. Evolutionary Computation in Combinatorial Optimization, Proc. Lecture Notes in Comp. Science 3906: 13-24 (2006)

[4] Remde, S., Cowling, P., Dahal, K. and Colledge, N.: Exact/Heuristic Hybrids using rVNS and Hyperheuristics for Workforce Scheduling. Evolutionary Computation in Combinatorial Optimization Proc., LNCS, Springer (2007)

[5] Hartmann, S.: Project Scheduling under Limited Resources: Model, methods and applications. Springer-Verlag, Berlin Heidelberg, New York (1999)

[6] Vanlaarhoven, P.J.M, Aarts, E.H.L and Lenstra, J.K.: Job Shop Scheduling by Simulated Annealing. Operations Research 40(1), pp 113-125 (1992)

[7] Toth, P. and Vigo, D. (Eds.): The Vehicle-Routing Problem. Siam, (2001)

[8] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B. (Eds.): The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley, (1991)

[9] Hansen, P. and Mladenovic, N.: Variable neighborhood search: Principles and applications. European Journal of Oper. Res. 130 (3): 449-467 May 1 (2001)

[10] Mladenovic, N. and Hansen, P.: Variable neighborhood search. Computers & Operational Research 24 (11): 1097-1100 Nov (1997)

[11] Potgieter, G. and Engelbrecht, A. P.: Genetic Algorithms for the Structure Optimisation of learned Polynomial Expressions. Applied Mathematics and Computation 186 (2): 1441-1466 Mar (2007)

[12] Shimojika, K., Fukuda, T., Hasehawa Y.: Self-Tuning Fuzzy Modeling with adaptive membership function, rules, and hierarchical structure-based on Genetic Algorithm. Fuzzy Sets And Systems 71 (3): 295-309 (1995)

[13] Reeves, C.R.: Genetic Algorithms and Combinatorial Optimization. In V.J. Rayward-Smith (Ed.), Applications of Modern Heuristic Methods, Alfred Waller, Henley-on-Thames, 111–125. (1995)

[14] Yao, X.: Evolving artificial neural networks.: Proceedings Of The IEEE 87 (9): 1423-1447 (1999)

[15] Valls, V., Ballestin, F. and Quintanilla, S.: Justification and RCPSP: A technique that pays. European Journal of Operational Research, Volume 165, Issue 2, (2005)