Reducing Local Optima in Single-Objective Problems by Multi-objectivization

Joshua D. Knowles[1]*, Richard A. Watson[2], and David W. Corne[1]

[1] School of Computer Science, Cybernetics and Electronic Engineering, University of Reading, Reading RG6 6AY, UK
{j.d.knowles,d.w.corne}@reading.ac.uk
[2] Dynamic and Evolutionary Machine Organization, Volen Center for Complex Systems, MS018, Brandeis University, Waltham, MA 02454, USA
richardw@cs.brandeis.edu

Abstract. One common characterization of how simple hill-climbing optimization methods can fail is that they become trapped in local optima - a state where no small modification of the current best solution will produce a solution that is better. This measure of 'better' depends on the performance of the solution with respect to the single objective being optimized. In contrast, multi-objective optimization (MOO) involves the simultaneous optimization of a number of objectives. Accordingly, the multi-objective notion of 'better' permits consideration of solutions that may be superior in one objective but not in another. Intuitively, we may say that this gives a hill-climber in multi-objective space more freedom to explore and less likelihood of becoming trapped. In this paper, we investigate this intuition by comparing the performance of simple hill-climber-style algorithms on single-objective problems and multi-objective versions of those same problems. Using an abstract building-block problem we illustrate how 'multi-objectivizing' a single-objective optimization (SOO) problem can remove local optima. Then we investigate small instances of the travelling salesman problem where additional objectives are defined using arbitrary sub-tours. Results indicate that multi-objectivization can reduce local optima and facilitate improved optimization in some cases. These results enlighten our intuitions about the nature of search in multi-objective optimization and sources of difficulty in single-objective optimization.

1 Introduction

One of the most general heuristics used in optimization techniques is the idea that the value of solutions is to some extent correlated with how similar the solutions are; crudely, that a good solution is more likely to be found nearby to other good solutions than it is to be found nearby an arbitrary solution. Naturally, 'nearby' or 'similar' needs to be qualified. The simplest notion of similarity of solutions is their proximity as measured in the problem parameters given. But alternatively, we may define proximity in terms of the variation operators used

* http://www.reading.ac.uk/~ssr97jdk/

by the search algorithm [7]. In any case, the simplest way to use this heuristic is a hill-climbing algorithm: start with some random solution, try variations of this solution until a better solution (or at least, non-worse solution) is found, move to this new solution and try variations of this, and so on. But the actual success of a hill-climber requires a stronger assumption to be true: that from any point in the solution space there is a path through neighbouring points to a global optimum that is monotonically increasing in value[1]. If this is true then a hill-climber can find a global optimum - and, although a hill-climber can do better than random guessing on almost all practical problems we encounter, it usually does not find a global optimum. More likely, it gets stuck in a local optimum - a sub-optimal point or plateau that has no superior neighbouring points.

There are several approaches that can be taken to overcome the limitations of a simple hill-climber. Broadly, many approaches can be seen as one of the following: changing the neighbourhood structure of the solution space so that the strong assumption is true; or relaxing the strong assumption and, one way or another, utilizing solutions which are inferior to some extent. Changing the neighbourhood structure can be done by something as simple as increasing the neighbourhood 'radius' by increasing mutation, or by a complete redesign of how solutions are represented and new variants are created, or perhaps by adding redundancy so that a hill-climber can travel along 'neutral networks' [5] to find superior points without having to go through inferior points. Relaxing the strong assumption can be done by, amongst other things, probabilistically accepting inferior solutions, as in simulated annealing, or by the use of multi-point searchers, or multi-restart searchers, where although one searcher may become stuck another, at a different location, may continue.

In this paper, we investigate a different approach, similar to one previously proposed in [10]. Rather than changing the neighbourhood structure so that we can always find a superior point, or accepting search paths through inferior points, we use a different definition of superior and inferior. Specifically, we use a method of comparing two solution that is common in multi-objective optimization (MOO) techniques where more than one measure of a solution is provided. Briefly, under Pareto optimization, a solution $\mathbf{x}$ is superior (said to Pareto dominate) another solution $\mathbf{x}'$ if and only if it is at least as good as $\mathbf{x}'$ in all measures and better in at least one measure. Put another way, if $\mathbf{x}'$ is better than $\mathbf{x}$ in at least one measure then it is not "inferior" to $\mathbf{x}$. Our intuition is this: if we can add other objectives to a problem to make it multi-objective, and use this relaxed notion of inferiority, then we may open up monotonically increasing paths to the global optimum that are not available under the original single-objective optimization (SOO) problem. We call this approach "multi-objectivization".

Naturally, the effect of this transformation will depend, in part, on exactly how we 'objectivize' the problem, and the particulars of the algorithm that uses the new multi-objective problem space. To begin with, we illustrate the principle on a test function that has an obvious decomposition. We explain why decomposing this problem naturally leads to the removal of all local optima in the search space, and demonstrate this fact empirically with results showing that a Pareto hillclimber (PAES) can solve the problem much more efficiently than a

[1] 'Efficient' success also requires that this path is not exponentially long [4].

hillclimber. We also compare the performance of PAES with that of simulated annealing on this problem, and show that increasing the freedom of movement for a hill climber, by decomposing a problem into multiple objectives, is more effective than relaxing the strong assumption for a hill-climber, as in simulated annealing. This illustrates an idealized multi-objectivization.

Following this, we take the well-known travelling salesperson problem (TSP) as an exemplar real-world problem with a single objective (to minimize the tour length) and show how it may be decomposed into sub-objectives. We then perform a number of experiments to measure the effectiveness of decomposing the problem, by comparing various single-point and multi-point hill-climbers on the original landscape and on the multi-objective landscape. Some comparison with simulated annealing is also provided. Several instances of the problem are considered, and we attempt to establish the effect the choice of different sub-objectives has on the efficacy of the resultant decomposition.

The remainder of the paper is structured as follows: Section 2 defines the concepts of single and multi-objective optimization, and introduces the technique of multi-objectivization. Section 3 defines the algorithms that we use in our experiments, and Section 4 defines the test problems we use both in their SOO and MOO versions. Section 5 describes the results of the experiments. Section 6 discusses implications and related research, and Section 7 concludes.

2 Single-Objective and Multi-objective Optimization

The general (unconstrained) single-objective combinatorial optimization problem can be expressed as:

$$\begin{array}{rl} \text{maximize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in X \end{array} \tag{1}$$

where $\mathbf{x}$ is a discrete solution vector, and X is a finite set of feasible solutions, and $f(\mathbf{x})$ maps X into $\Re$.

Similarly, the multi-objective combinatorial optimization (MOCO) problem can be expressed as:

$$\begin{array}{rl} \text{“maximize”} & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})) \\ \text{subject to} & \mathbf{x} \in X \end{array} \tag{2}$$

where the vector objective function $\mathbf{f}(\mathbf{x})$ maps X into $\Re^K$, where $K \geq 2$ is the number of objectives. The term ‘maximize’ appears in quotation marks because, in general, there does not exist a single solution that is maximal on all objectives. Instead, one may seek to find a set of solutions $X^* \subseteq X$, called the Pareto optimal set, with the property that:

$$\forall\, \mathbf{x}^* \in X^* \bullet \nexists\, \mathbf{x} \in X \bullet \mathbf{x} \succ \mathbf{x}^* \tag{3}$$

where $\mathbf{x} \succ \mathbf{x}^* \iff ((\forall i \in 1..K \bullet (f_i(\mathbf{x}) \geq f_i(\mathbf{x}^*)) \wedge (\exists\, i \in 1..K \bullet f_i(\mathbf{x}) > f_i(\mathbf{x}^*)))$. The expression $\mathbf{x} \succ \mathbf{x}^*$ is read as $\mathbf{x}$ *dominates* $\mathbf{x}^*$, and solutions in the Pareto optimal set are also known as efficient or admissible solutions. In addition, for two solutions $\mathbf{x}$ and $\mathbf{x}'$, we say $\mathbf{x} \sim \mathbf{x}'$ if and only if $\exists\, i \in 1..K \bullet f_i(\mathbf{x}) > f_i(\mathbf{x}') \wedge \exists\, j \in 1..K \bullet j \neq i \bullet f_j(\mathbf{x}') > f_j(\mathbf{x})$. Such a pair of solutions are said to be *incomparable*, and each is *nondominated* with respect to the other. Since any member of a set of mutually nondominated solutions is not worse (dominated) than any other, a hillclimbing algorithm would be free to move from any one of them to any other if the variation operator allows. This is the notion that is important to allow the increased freedom of a Pareto hill-climber, and which may reduce the problem of local optima.

2.1 Multi-objectivization

To perform multi-objectivization we must either replace the original single objective of a problem with a set of new objectives, or add new objectives in addition to the original function. In either case, we want to be sure that the global optimum of the original problem is one of the points that is Pareto optimal in the multi-objective version of the problem. Specifically, the problem must be restated so as to maximize $K \geq 2$ objective functions such that the following relation between solutions in the two formulations holds:

$$\forall \mathbf{x}^{opt} \in X^{opt} \bullet \exists \mathbf{x}^* \in X^* \bullet \mathbf{x}^* = \mathbf{x}^{opt} \tag{4}$$

where $\mathbf{x}^{opt}$ is an optimal solution to the SOO problem, and X^{opt} is the set of all such solutions, and $\mathbf{x}^*$ and X^* relate to the MOO formulation of the problem, and have the meanings attributed above.

Part of the intuition that motivates multi-objectivization comes from notions of problem decomposition - dividing a problem into a number of smaller subproblems and solving each and all of them. Accordingly, it may be appropriate to define an objective as a function similar to the original function but over a subset of the problem parameters. For example, in our TSP example we add objectives corresponding to the length of parts of the tour. Defining functions over subsets of the problem parameters has clear connections with divide and conquer techniques, especially dynamic programming [1]. An alternative approach to multi-objectivization is to define different functions over the same (entire) set of problem parameters. For example, to take a different domain, if we desire a locomotion controller for a legged robot we might suppose that it needs to both lift its body off the floor, and swing it legs forward, both of which may depend on all of the parameters.

The skill of the researcher in either approach (similar functions over subsets, or different functions over the entire set) is to separate out the conflicting aspects of the problem - to find objectives that are as independent as possible. This is not always easy or possible. However, we suppose that in some circumstances the multi-objectivization approach may have useful tolerance of 'lazy' decompositions, where sub-problems are not completely independent. Perhaps this is because a solution that is better in all objectives is preferred if available, but a solution that exploits one objective at the expense of another objective is still valuable in the case that a dominant solution cannot be found.

In the two examples introduced later, the first uses different functions over the entire parameter set, and the second, TSP, uses similar functions over subsets of the solution. The examples serve to illustrate the different approaches to multi-objectivization, the issues involved, and show some cases where multi-objectivization can be successful.

3 Algorithms

In order to test the hypothesis that multi-objectivizing a problem can reduce the number of local optima in a way that is useful for performing local search, we employ a number of simple neighbourhood search algorithms described below. The first pair of algorithms, consisting of a simple hillclimber (SHC), and a multi-objective hillclimber similar in operation to the Pareto archived evolution strategy (PAES) [8, 9], are both single-point hill-climbers. PAES represents the multi-objective analogue of SHC: both algorithms accept a neighbour of the current solution if it is not worse than any solution found so far. In the context of PAES, however, 'worse' means dominated.

The second pair of algorithms, which are a mutation-only genetic algorithm with deterministic crowding [11] (DCGA), and the Pareto envelope-based selection algorithm (PESA) [2], are both multi-point hill-climbers (neither uses recombination here). Once again, they are supposed to be analogues of each other, subject to the differences forced upon them by the different requirements of single and multiple objective optimization. The analogy between them is not as clear as between SHC and PAES because the selection and crowding methods used by them is more complicated, but they have sufficiently similar operation to warrant comparison for our purposes here.

The performance of the two pairs of algorithms above are also compared with a simulated annealing (SA) algorithm. This comparison is intended to place the effect of reducing the number of local optima achieved by multi-objectivization into a context which is familiar to the reader. SA incrementally adjusts the strictness with which it rejects moves to worse solutions, and does so very effectively; it thus serves as a useful comparison to the effect of multi-objectivization.

3.1 Single-point hill-climbers

Initialization:	$B \leftarrow \emptyset$
	$\mathbf{x} \in X \leftarrow \mathsf{Init}(\mathbf{x})$
	$B \leftarrow B \cup \mathbf{x}$
Main Loop:	$\mathbf{x}' \in X \leftarrow \mathsf{Mutate}(\mathbf{x})$
	`if` $(\mathsf{Inferior}(\mathbf{x}', B) \neq \mathsf{TRUE})\,\{$
	$\quad \mathbf{x} \leftarrow \mathbf{x}'$
	$\quad B \leftarrow \mathsf{Reduce}(\mathbf{x}' \cup B)\,\}$
Termination:	`return` $\mathsf{Best}(B)$

Fig. 1. Generic pseudocode for hill-climbing algorithms

Pseudocode for a generic version of a hill-climbing algorithm that may be single or multi-objective is given in Figure 1. The current solution vector is denoted by $\mathbf{x}$, and B is the minimal representation of the best solutions encountered so far. The function $\mathsf{Mutate}(\mathbf{x})$ returns a new solution $\mathbf{x}'$ made by variation of $\mathbf{x}$. For the simple, single-objective hillclimber, the functions $\mathsf{Inferior}()$, $\mathsf{Reduce}()$ and $\mathsf{Best}()$ are very simple. The function $\mathsf{Inferior}(\mathbf{x}', \mathsf{B})$ returns true iff there is any element of the set B whose evaluation is greater than $\mathbf{x}'$, $\mathsf{Reduce}(B)$ returns the set of equally maximum value elements of the set B, and $\mathsf{Best}(B)$ returns any member of B, because they are equally good. For the PAES-based multi-objective hill-climber, these functions have Pareto versions that follow the same semantics: $\mathsf{Inferior}(\mathbf{x}', B)$ returns true iff there is any element of the set B that dominates $\mathbf{x}'$, $\mathsf{Reduce}(B)$ returns the set of elements from B that are not dominated by any other member of B, and $\mathsf{Best}(B)$ returns the element of B that is maximal in the original single objective. In our experiments here, both SHC and PAES are terminated when the number of function evaluations reaches a predetermined number, num_evals.

3.2 Multi-point hill-climbers

Initialization:	$P \leftarrow \emptyset$
	$\mathsf{Init\_pop}(P)$
Main Loop:	$\mathbf{x_1} \leftarrow \mathsf{Rand\_mem}(P),\ \mathbf{x_2} \leftarrow \mathsf{Rand\_mem}(P)$
	$\mathbf{x_1'} \leftarrow \mathsf{Mutate}(\mathbf{x_1}),\ \mathbf{x_2'} \leftarrow \mathsf{Mutate}(\mathbf{x_2})$
	`if` $(\mathsf{H}(\mathbf{x_1}, \mathbf{x_1'}) + \mathsf{H}(\mathbf{x_2}, \mathbf{x_2'}) > \mathsf{H}(\mathbf{x_1}, \mathbf{x_2'}) + \mathsf{H}(\mathbf{x_2}, \mathbf{x_1'}))$
	$\mathsf{Swap}(\mathbf{x_1'}, \mathbf{x_2'})$
	`if` $(f(\mathbf{x_1'}) > f(\mathbf{x_1}))$
	$P \leftarrow P \cup \mathbf{x_1'} \setminus \mathbf{x_1}$
	`if` $(f(\mathbf{x_2'}) > f(\mathbf{x_2}))$
	$P \leftarrow P \cup \mathbf{x_2'} \setminus \mathbf{x_2}$
Termination:	`return` $\mathsf{Best}(P)$

Fig. 2. A simple form of a genetic algorithm using deterministic crowding (DCGA) as used in our experiments

Pseudocode for the mutation-only, deterministic crowding genetic algorithm (DCGA) is given in Figure 2. The set P is the population of candidate solutions initialized randomly using $\mathsf{Init\_pop}(P)$. The function $\mathsf{H}(\mathbf{x}, \mathbf{x}')$ measures the genotypic Hamming distance between two solution vectors. At each iteration, two parents are selected from the population at random and two offspring are generated by mutation, one from each parent. Parents and offspring are then paired up so as to minimize the sum of the genotypic Hamming distance between them. Each offspring then replaces the parent it is paired with if it non-worse than that parent.

The PESA algorithm used here has been described in detail in [2], and pseudocode for it is given in Figure 3. It has an internal population IP of size P_I, and an external population of nondominated solutions EP. Here it is used without crossover so that each generation consists of selecting P_I parents from EP and mutating them to produce P_I new offspring. Then, the nondominated members of IP are incorporated into EP. The selection operator, $\mathsf{select}()$ is based on crowding in objective space.

The PESA and DCGA algorithms are quite different in some ways, but do represent analogues of each other at a high level: both algorithms have a popula-

Initialization:	$\mathrm{IP} \leftarrow \emptyset,\ \mathrm{EP} \leftarrow \emptyset$
	Init_pop(IP)
	`foreach` ($\mathbf{x} \in \mathrm{IP}$)
	$\mathrm{EP} \leftarrow$ Reduce($\mathrm{EP} \cup \mathbf{x}$)
Main Loop:	$\mathrm{IP} \leftarrow \emptyset$
	`while` ($\lvert\mathrm{IP}\rvert < P_I$) {
	$\mathbf{x} \leftarrow$ Select(EP)
	$\mathbf{x}' \leftarrow$ Mutate($\mathbf{x}$)
	$\mathrm{IP} \leftarrow \mathrm{IP} \cup \mathbf{x}$ }
	`foreach` ($\mathbf{x} \in \mathrm{IP}$)
	$\mathrm{EP} \leftarrow$ Reduce($\mathrm{EP} \cup \mathbf{x}$)
Termination:	`return` Best(EP)

Fig. 3. The Pareto envelope-based selection algorithm (PESA)

tion from which parents are selected and used to produce offspring via mutation, and both have a mechanism for maintaining diversity in the population. However, in PESA, the diversity is maintained in the objective space, while with DCGA it is in the genotype space. With PESA, the initial size of the pool of random solutions is P_I, and P_I solutions are generated at each step. The prevailing size of EP is the size of the pool from which solutions can be selected. In DCGA, the population size P determines the initial pool and the size of the pool from which solutions are selected, and 2 solutions are generated at each step. These differences in detail, however, should not affect the substance of the judgments we make about the performance of these algorithms in our experiments.

3.3 Simulated annealing

The simulated annealing (SA) algorithm used is identical to the SHC, except for changes to the acceptance function to allow it to accept moves to worse solutions. The acceptance function we employ for accepting degrading moves is the standard exponential function:

$$p\,(\text{accept } \mathbf{x}') = \frac{\exp\left(f(\mathbf{x}) - f(\mathbf{x}')\right)}{T} \tag{5}$$

where $f(\mathbf{x})$ is the evaluation of the current solution $\mathbf{x}$, $f(\mathbf{x}')$ is the evaluation of the neighbouring solution, $\mathbf{x}'$, and $p\,(\text{accept } \mathbf{x}')$ denotes the probability of accepting $\mathbf{x}'$.

We choose to use a simple form of simulated annealing, employing an inhomogeneous, geometric cooling schedule where T is updated after every iteration using:

$$T \leftarrow \alpha T \tag{6}$$

Following general procedures outlined in [14], we attempt to set the starting temperature T_0 so that between 40% and 60% of moves are accepted. The final temperature is set so that between 0.1% and 2% of moves are accepted. We can then calculate α using:

$$\alpha = \exp\left(\frac{\ln(T_f/T_0)}{num\_evals}\right) \tag{7}$$

given that we know the total number of function evaluations num_evals required.

4 Problems

4.1 H-IFF/MH-IFF: An abstract illustration

The Hierarchical-if-and-only-if function, H-IFF, is a genetic algorithm test problem designed to model a problem with a building-block structure where the sub-problem that each block represents has strong interdependencies with other blocks. That is, unlike many existing building-block problems, the optimal solution to any block in H-IFF is strongly dependent on how other building-blocks have been solved [16, 17].

The fitness of a string using H-IFF can be defined using the recursive function given below. This function interprets a string as a binary tree and recursively decomposes the string into left and right halves. Each resultant sub-string constitutes a building-block and confers a fitness contribution equal to its size if all the bits in the block have the same allele value - either all ones or all zeros. The fitness of the whole string is the sum of these fitness contributions for all blocks at all levels.

$$f(\mathrm{B}) = \begin{cases} 1, & \text{if } |\mathrm{B}| = 1, \text{ else} \\ |\mathrm{B}| + f(\mathrm{B_L}) + f(\mathrm{B_R}), & \text{if } (\forall i\{b_i = 0\} \text{ or } \forall i\{b_i = 1\}), \\ f(\mathrm{B_L}) + f(\mathrm{B_R}), & \text{otherwise,} \end{cases} \tag{8}$$

where B is a block of bits, $\{b_1, b_2, \ldots, b_n\}$, $|\mathrm{B}|$ is the size of the block $= n$, b_i is the ith element of B, and $\mathrm{B_L}$ and $\mathrm{B_R}$ are the left and right halves of B (i.e. $\mathrm{B_L} = \{b_1, b_2, \ldots, b_{n/2}\}$ and $\mathrm{B_R} = \{b_{n/2+1}, \ldots, b_n\}$). The length of the string evaluated, n, must equal 2^p where p is an integer (the number of hierarchical levels).

Each of the two competing solutions to each block (all-ones and all-zeros) give equal fitness contributions on average. But only when neighbouring blocks match do they confer a bonus fitness contribution by forming a correct block at the next level in the hierarchy. These competing solutions and their interdependencies create strong epistatic linkage in H-IFF and many local optima (see Figure 4 left). These local optima prevent any kind of mutation based hill climber from reliably reaching one of the two global optima in H-IFF (all-ones or all-zeros) in time less than exponential in N, the number of bits in the problem [17].

Figure 4 (left) shows a particular section through the H-IFF landscape. This is the section through a 64-bit landscape starting from all zeros on the left and ending with all ones. Specifically, it shows the fitness of the strings "000...0", "100...0", "110...0", ..., "111...1". This indicates the local optima in H-IFF and the two global optima at opposite corners of the space.

$$f_k(\mathrm{B}) = \begin{cases} 0, & \text{if } |\mathrm{B}| = 1 \text{ and } b_1 \neq k, \text{ else} \\ 1, & \text{if } |\mathrm{B}| = 1 \text{ and } b_1 = k, \text{ else} \\ |\mathrm{B}| + f_k(\mathrm{B_L}) + f_k(\mathrm{B_R}), & \text{if } (\forall i\{b_i = k\}), \\ f_k(\mathrm{B_L}) + f_k(\mathrm{B_R}), & \text{otherwise.} \end{cases} \tag{9}$$

where $f_0(\mathbf{x})$ is the first objective and $f_1(\mathbf{x})$ is the second.

This particular decomposition of H-IFF results in a two-objective problem in which there are no local optima for a multi-objective hill-climber. Figure 4 (right)

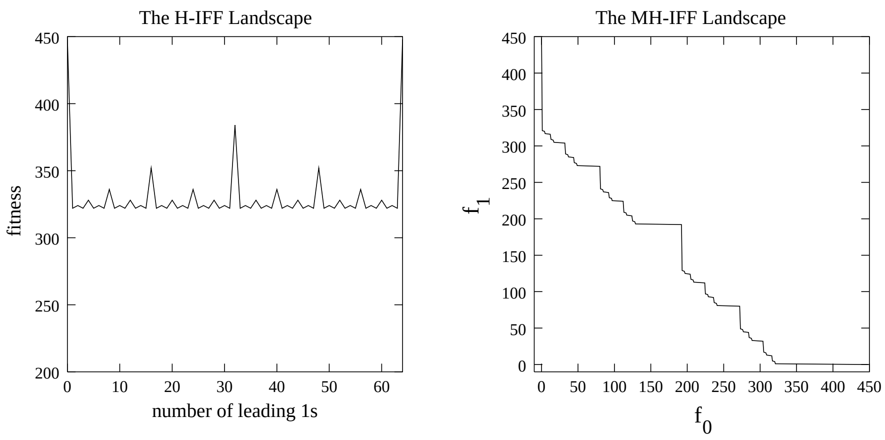

Fig. 4. H-IFF and MH-IFF

is a section through the two-objective landscape, MH-IFF, using the same strings as used in the section through H-IFF. We see that if neighbouring points on this section are worse in respect to one dimension, they are better in the other. This has transformed the problem into one that is now completely bit-climbable. That is, single bit changes can move a hill-climber from one end of the Pareto front to the other, including every point in between, without ever moving to a dominated point. (In fact, every Pareto optimal solution can be reached by bit-climbing from any start point.) This transformation is possible because we have access to the structure of the function and can thereby separate the conflicting sub-goals inherent in the SOO version. This serves to illustrate the mechanisms of multi-objectivization - but naturally, we will not have this luxury in practical applications, like the TSP.

4.2 TSP/MTSP: decomposition via multi-objectivization

The travelling salesperson problem (TSP) is the most well-known of all $\mathcal{N}P$-hard optimization problems. For a comprehensive review and comparison of methods used to solve it see [6], where the problem is stated as follows: We are given a set $\mathrm{C} = \{c_1, c_2, \ldots, c_N\}$ of *cities* and for each pair $\{c_i, c_j\}$ of distinct cities there is a *distance* $d(c_i, c_j)$. Our goal is to find an ordering π of the cities that minimizes the quantity

$$\sum_{i=1}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(N)}, c_{\pi(1)}). \tag{10}$$

In order to multi-objectivize the TSP, we need to identify sub-problems to be solved. Of course, the TSP is $\mathcal{NP}$-hard for the very reason that there is no good decomposition of the problem, i.e. dependencies between components of the problem exist in most instances. However, an obvious decomposition, although by no means perfect, is simply to divide the problem into two (or more) sub-tours, each to be minimized. This can be done in a number of ways, and some may be preferable to others depending on how much is known about the problem instance, but here we use a method that is general for the TSP class. Specifically, to make a two-objective formulation of the TSP, we define two distinct sub-tours to be minimized. The sub-tours are defined by two cities, and the problem becomes:

$$\begin{aligned} \text{“minimize”}\ \mathbf{f}(\pi, a, b) &= (f_1(\pi, a, b), f_2(\pi, a, b)) \\ \text{where}\ f_1(\pi, a, b) &= \textstyle\sum_{i=\pi^{-1}(a)}^{\pi^{-1}(b)-1} d(c_{\pi(i)}, c_{\pi(i+1)}) \\ \text{and}\ f_2(\pi, a, b) &= \textstyle\sum_{i=\pi^{-1}(b)}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + \\ &\quad \textstyle\sum_{i=1}^{\pi^{-1}(a)-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(N)}, c_{\pi(1)}) \end{aligned} \tag{11}$$

where a and b are the two cities specified *a priori*, and if $\pi(a) < \pi(b)$ they are swapped. It is intended that a and b are chosen arbitrarily. Notice that the sum of the two objectives is the same as the quantity to be minimized in (10). This ensures that the optimum of (10) is coincident with at least one of the Pareto optima of (11), as required by our definition of multi-objectivizing.

5 Results

5.1 H-IFF/MH-IFF

In this section, we present a comparison of hillclimbing algorithms on the H-IFF problem, and its multi-objectivized formulation, MH-IFF. All algorithms were run on a 64-bit version of the problem, for the same number of function evaluations, num_evals = 500000. Two different mutation rates $p_m = 1/n, 2/n, n = 64$ were used in each algorithm to investigate whether mutation rate choice substantially affected any of the algorithms. The choice of archive size for PAES was 100. The DCGA and PESA algorithms require population sizes to be set. For DCGA, $P = 100$, and for PESA, $P_I = 10$. For the simulated annealing algorithm, preliminary runs were performed and T_0 and T_f were adjusted until the acceptance probabilities fell into the required ranges described in Section 3.

Table 1 shows the full set of results collected on the 64-bit H-IFF problem. In each case, the results were gathered from 30 independent runs of the algorithm. For all algorithms, all best genotypically different solutions were stored off-line, and the tabulated results relate to these off-line results. The table clearly indicates that on H-IFF, removing the local optima through multi-objectivization transforms the problem from one that is very difficult for a neighbourhood searcher, to one that is much easier. The performance of the SOO search algorithms improves with increased mutation rate p_m, confirming that they require larger steps to escape the local optima in the H-IFF problem. The multi-objective algorithms are almost unaffected by the choice of p_m, indicating that they are able to reach the optima through small neighbourhood moves. These results demonstrate the multi-objectivization principle, clearly and can be understood with reference to the discussion given to the problem in Section 4. In the following section we examine how the multi-objectivization technique established here fares on a real-world problem with a much less readily decomposable structure, the TSP.

Table 1. Results of the comparison between algorithms on a 64-bit H-IFF problem. Two of the algorithms are multi-objective and use the MH-IFF decomposition of the problem, namely PAES and PESA. The other three algorithms use the H-IFF objective function directly. Essentially the results compare two single-point hill-climbers, SHC and PAES, and two multi-point hill-climbers, DCGA and PESA. In both cases, the multi-objective algorithm significantly outperforms (using any statistical test) its SOO counterpart. The results of the simulated annealing algorithm (SA) act as a benchmark, indicating the level of performance that can be achieved when escape from local optima is made possible on the original landscape. The columns, '% one' and '% both', indicate the percentage of the runs where respectively one of the optima and both optima were found over the thirty independent runs of each algorithm. Note that only PAES and PESA are able to find both optima.

Algorithm	p_m	best	mean	σ	% one	% both
SHC	$1/n$	288	242.13	22.52	0	0
	$2/n$	336	267.47	29.46	0	0
PAES	$1/n$	448	415.20	51.26	70	47
	$2/n$	448	418.13	50.68	74	43
DCGA	$1/n$	300	270.06	13.80	0	0
	$2/n$	448	323.93	26.54	3	0
PESA	$1/n$	448	448.00	0.00	100	100
	$2/n$	448	448.00	0.00	100	100
SA	$1/n$	448	435.20	26.04	80	0
	$2/n$	448	435.20	26.04	80	0

5.2 TSP/MTSP

We present results for a range of TSP instances of varying size and type. All problems are symmetric, i.e. the distance from A to B is the same as from B to A, where A and B are any two cities. The RAN-20 and RAN-50 problems have 20 and 50 cities respectively, and are non-Euclidean random weight problems where the distance between each pair of cities is a random real number in [0,1). The EUC-50 and EUC-100 are two randomly generated Euclidean problems where the cities are given co-ordinates in a 2-d plane, $x \in [0,1)$, $y \in [0,1)$ and the distance between pairs of nodes is then the Euclidean distance. The problem, kroB100, is taken from TSPLIB and is also a 100-node Euclidean TSP problem. The last problem, mnPeano-92, is a 92-node fractal problem[2] with a known optimal solution [13].

In each algorithm, the representation, initialization, and mutation operators used are identical: the representation of a tour is an N-gene permutation of the numbers $1..N$; the initialization procedure simply generates a random permutation of N cities; and the mutation operator used is the 2-change operator, originally put forward in [3]. It works by selecting two non-identical cities and reversing the order of all the cities between (and including) them. This operator preserves all but two edges in the tour.

[2] We conjecture that fractal TSP problems may be particularly suitable for multi-objectivization because their inherently hierarchical structure suggests they may be amenable to decomposition.

On each different TSP instance, all the algorithms are run for the same number of function evaluations, *num_evals*, given for each problem in Table 2. The PESA and DCGA algorithms further require population sizes to be set. As before, we use the default values of $P = 100$ for DCGA, and $P_I = 10$ for PESA. Setting the SA parameters is carried out as before using preliminary runs to derive an appropriate T_0 and T_f.

The choice of city pairs to be used in the multi-objective algorithms is investigated. First, we present results (Table 2) in which - for the multi-objective algorithms, PAES and PESA - a single, random pair of cities was selected and this pair used in all runs. Later, we report the maximum deviation from these results for other choices of city pairs. Finally, we investigate the choice of city pairs using the EUC-50 problem by selecting a pair of cities that are very close, and another pair where they are maximally distant, and repeating our experiments for these choices.

Table 2. Summary TSP results. In the 'Optimum' column, figures given in bold font represent the known optimum value; other figures are estimates. For the RAN-20 problem, the optimum is an estimate based on the fact that SA reached this value on 30 consecutive runs, and given the small size of the problem. For the RAN-50 problem, the estimated figure is based on the expected limiting value of an optimal tour [6], and similarly for EUC-50 and EUC-100, the estimates are based on the formula for expected tour length = $K\sqrt{NA}$ with N the number of cities, $A = 1.0$ the area in which the cities are placed, and $K \approx 0.7124$ [6].

Algorithm	Problem	*num_evals*	Optimum	Best	Mean	σ
SHC	RAN-20	500000	2.547394	2.550811	2.81	0.14
PAES	RAN-20	500000	2.547394	2.547394	2.66	0.14
SA	RAN-20	500000	2.547394	2.547394	2.55	0.00
SHC	RAN-50	500000	2.0415	2.620087	3.09	0.28
PAES	RAN-50	500000	2.0415	2.259948	2.73	0.22
DCGA	RAN-50	500000	2.0415	2.307587	2.46	0.09
PESA	RAN-50	500000	2.0415	2.189421	2.32	0.28
SA	RAN-50	500000	2.0415	2.130675	2.30	0.10
SHC	EUC-50	500000	5.0374	5.904673	6.23	0.20
PAES	EUC-50	500000	5.0374	5.801026	6.03	0.13
DCGA	EUC-50	500000	5.0374	5.707789	5.76	0.05
PESA	EUC-50	500000	5.0374	5.692169	5.78	0.08
SA	EUC-50	500000	5.0374	5.692169	5.72	0.03
SHC	EUC-100	2000000	7.124	8.143720	8.55	0.23
PAES	EUC-100	2000000	7.124	8.028227	8.35	0.24
DCGA	EUC-100	2000000	7.124	7.902731	8.16	0.14
PESA	EUC-100	2000000	7.124	7.795515	7.97	0.10
SA	EUC-100	2000000	7.124	7.853258	7.98	0.07
PESA	kroB100	2000000	**22141**	22141	22546.1	324.2
SA	kroB100	2000000	**22141**	22217	22529.2	173.0
SHC	mnPeano-92	1000000	**5697.93**	5857.47	6433.45	197.1
PAES	mnPeano-92	1000000	**5697.93**	5879.63	6255.30	197.6

From Table 2 we can see that, without exception, the results of the PAES algorithm are superior to the SHC algorithm at a statistically significant level[3], over the range of problem instances in the Table. This shows that the number of local optima in the TSP problem, using the 2-change neighbourhood, is reduced by the method of multi-objectivization we have proposed.

Although the number of local optima has successfully been reduced by multi-objectivization, this does not make PAES more effective than other methods for solving TSP. We are only using PAES to indicate the basic advantage of multi-objectivization over hill-climbing in a SOO problem. Compared with DCGA, PAES performs poorly on all problems. However, PESA, the multi-objective counterpart of DCGA, outperforms DCGA on all but one of the problems. PESA is also competitive with the SA algorithm, which has its own explicit means of escaping from local optima. This result adds further evidence that multi-objectivization can enable an algorithm to avoid local optima effectively.

To investigate the effect of the choice of city pairs, the 50-node Euclidean TSP problem was used. First, a choice of city pair in which the cities were very close to each other and relatively far from others was selected. 30 runs were performed using this pair and the results were: best = 5.804719, mean = 6.10, and $\sigma = 0.22$. Then, with cities in opposite corners of the plane the results were: best = 5.818185, mean = 6.09, $\sigma = 0.16$. So both are worse than the 'random' choice used in the results in Table 2: best = 5.801026, mean = 6.03, $\sigma = 0.13$. However, although there is some seeming dependence on city choice, all three of these results are better than the results of the SHC algorithm on this problem. Some alternative choices of city pairs were used on some of the other problems, too. On the random 20-node problem, three different choices of node pair were tried. There was a 1.5% variation in the mean over 30 runs, for the different node-pair choices. On the 50-node Euclidean problem, two different pairs of cities were chosen for the PESA algorithm. The difference in means over 30 runs was 0.4%. On no runs did the choice of city pair affect the mean TSP tour length by more than 2%.

6 Discussion

The examples in the previous sections have illustrated different ways in which additional objectives can be defined for a SOO problem and that this multi-objectivization can facilitate improved search for neighbourhood based algorithms. We suggested earlier that a successful multi-objectivization may involve decomposing the original function into a number of sub-problems and that these sub-problems should be as independent as possible. In the H-IFF/MH-IFF example, we can separate the two competing components of the problem completely, making the problem very easy for a hill-climber. In TSP this is not so easy - there is no (known) method to decompose $\mathcal{NP}$-hard problems, like TSP, that creates independent sub-problems - indeed, it is the interdependency of the problem components that puts them in this class. Nonetheless, we suggest that there may be some merit in examining further the parallels between sub-problems in SOO,

[3] Using a large-sample test of hypothesis for the difference in two population means [12] (which does not depend upon distribution), at the 95% confidence level.

and objectives in MOO. Specifically, if different objectives separate out different components of a problem then different points on the Pareto front correspond to solutions that are more or less specialized to different sub-problems. Our experiments support the intuition that it will be easier to discover a set of different specialists than it is to discover a generalist directly. In the experiments in this paper we used only mutation to investigate this, but if we could find some way to combine together different specialists from the front then perhaps it would facilitate discovery of generalists more directly. In a sense, this is the intuition behind recombination in the multi-objective Messy GA (MOMGA) [15].

However, the MOMGA assumes a number of objectives given *a priori*, and like other MOO methods, it is not intended as a technique for problem decomposition in SOO problems. But related research suggests that it may be possible to automatically discover objectives relating to sub-problems, and thereby apply a MOMGA-like algorithm to SOO problems. The Symbiogenic Evolutionary Adaptation Model [18] shares some features with the MOMGA but uses group evaluation of individuals to encourage them to self-organize to cover the problem space, and automatically discover sub-problems to be used as the objectives. This algorithm successfully solves the H-IFF problem without the introduction of additional objectives provided by the researcher.

7 Conclusions

In this paper, we have defined a process that we call "multi-objectivization" whereby the scalar function of a SOO problem is replaced by a vector of functions such that the resulting MOO problem has Pareto optima which coincide with the optima of the original problem. We investigated the effects of this transformation, in particular, the reduction of local optima for hill-climbing style algorithms. We illustrated the effect of the approach first on an abstract building-block problem, H-IFF, that is trivially amenable to such a decomposition. We then investigated the approach further, using several small instances of the TSP, where decomposition is inherently difficult. We defined a multi-objectivization of the problem based on minimizing two sub-tours. Our results showed that this simple multi-objectivization does seem to reduce the effect of local optima on simple hill-climbing algorithms. These preliminary results, suggest that there is a link between the presence of local optima in SOO problems and an underlying conflict of implicit objectives, and they shed some light on the processes of multi-objective search.

Acknowledgments

The authors would like to thank Anthony Bucci, Michiel de Jong, and the anonymous reviewers for their excellent comments and criticisms.

References

1. R. Bellman. Dynamic programming and multi-stage decision processes of stochastic type. In *Proceedings of the second symposium in linear programming*, volume 2, pages 229–250, Washington D.C., 1955. NBS and USAF.

2. D. W. Corne and J. D. Knowles. The Pareto-envelope based selection algorithm for multiobjective optimization. In *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 839–848, Berlin, 2000. Springer-Verlag.
3. M. M. Flood. The travelling-salesman problem. *Operations Research*, 4:61–75, 1956.
4. J. Horn, D. Goldberg, and K. Deb. Long path problems for mutation-based algorithms. Technical Report 92011, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana IL, 1992.
5. M. Huynen, P. Stadler, and W. Fontana. Smoothness within ruggedness: The role of neutrality in adaptation. *Proceedings of the National Academy of Sciences (USA)*, 93:397–401, 1996.
6. D. S. Johnson and L. A. McGeoch. The travelling salesman problem: a case study. In E. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons, 1997.
7. T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, 1995.
8. J. D. Knowles and D. W. Corne. The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Washington, D.C., July 1999. IEEE Service Center.
9. J. D. Knowles and D. W. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
10. S. J. Louis and G. J. E. Rawlins. Pareto optimality, GA-easiness and deception. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-5)*, pages 118–123, San Mateo, CA, 1993. Morgan Kaufmann.
11. S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 1995. IlliGAL Report 95001.
12. W. Mendenhall and R. J. Beaver. *Introduction to Probability and Statistics - 9th edition*. Duxbury Press, International Thomson Publishing, Pacific Grove, CA, 1994.
13. M. G. Norman and P. Moscato. The euclidean traveling salesman problem and a space-filling curve. *Chaos, Solitons and Fractals*, 6:389–397, 1995.
14. C. R. Reeves. Modern heuristic techniques. In V. Rayward-Smith, I. Osman, C. Reeves, and G. Smith, editors, *Modern Heuristic Search Methods*, chapter 1, pages 1–26. John Wiley and Sons Ltd., 1996.
15. D. A. Van Veldhuizen and G. B. Lamont. Multiobjective Optimization with Messy Genetic Algorithms. In *Proceedings of the 2000 ACM Symposium on Applied Computing*, pages 470–476, Villa Olmo, Como, Italy, 2000. ACM.
16. R. A. Watson, G. S. Hornby, and J. B. Pollack. Modeling building-block interdependency. In *Parallel Problem Solving from Nature - PPSN V*, pages 97–106. Springer-Verlag, 1998.
17. R. A. Watson and J. B. Pollack. Analysis of recombinative algorithms on a hierarchical building-block problem. In *Foundations of Genetic Algorithms (FOGA)*, 2000.
18. R. A. Watson and J. B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In *Proceedings of the Sixth International Conference of Parallel Problem Solving From Nature (PPSN VI)*, pages 425–436. Springer-Verlag, 2000.