Constraint Handling Guided by Landscape Analysis in Combinatorial and Continuous Search Spaces

Katherine M. Malan malankm@unisa.ac.za
Department of Decision Sciences, University of South Africa, Pretoria, South Africa

I. Moser imoser@swin.edu.au
Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne, Australia

https://doi.org/10.1162/evco_a_00222

Abstract

The notion and characterisation of fitness landscapes has helped us understand the performance of heuristic algorithms on complex optimisation problems. Many practical problems, however, are constrained, and when significant areas of the search space are infeasible, researchers have intuitively resorted to a variety of constraint-handling techniques intended to help the algorithm manoeuvre through infeasible areas and toward feasible regions of better fitness. It is clear that providing constraint-related feedback to the algorithm to influence its choice of solutions overlays the violation landscape with the fitness landscape in unpredictable ways whose effects on the algorithm cannot be directly measured. In this work, we apply metrics of violation landscapes to continuous and combinatorial problems to characterise them. We relate this information to the relative performance of six well-known constraint-handling techniques to demonstrate how some properties of constrained landscapes favour particular constraint-handling approaches. For the problems with sampled feasible solutions, a bi-objective approach was the best performing approach overall, but other techniques performed better on problems with the most disjoint feasible areas. For the problems with no measurable feasibility, a feasibility ranking approach was the best performing approach overall, but other techniques performed better when the correlation between fitness values and the level of constraint violation was high.

Keywords

Constraint handling, search space analysis, combinatorial optimisation, continuous optimisation, Broken Hill Problem.

1 Introduction

Complex optimisation problems of realistic dimensions are generally not solvable with deterministic methods, hence the popularity of metaheuristics. The optimiser takes feedback from a fitness function which defines the "landscape" that the algorithm moves in. Choosing the best algorithm for a given problem is an optimisation problem in itself.

Research into fitness landscapes has been motivated by the goal of finding characteristics that help select the most successful algorithm for a given class of problems. Many metrics for fitness landscapes have been proposed in the last three decades (Malan and Engelbrecht, 2013). While they sometimes succeed in explaining algorithm

Manuscript received: 10 August 2017; revised: 22 January 2018; accepted: 6 March 2018.

performance, they generally do not account for the effects constraints have on the landscape.

Many practical problems have constraints and a number of constraint-handling techniques have been proposed to tackle them (Michalewicz, 1995; Coello Coello, 1999). These techniques can be largely categorised into penalty functions, repair approaches, feasibility maintenance, and multi-objective approaches. The success of different constraint-handling approaches depends not only on the parameter settings, but also on the features of the problem. Michalewicz (1995) observed that the death penalty, where infeasible solutions are rejected, performs badly when the proportion of feasible solutions is low, because an evolutionary algorithm cannot collect information to build on when most solutions are discarded. Penalty methods in general are assumed to be unsuitable for problems where the optimum is on the boundary between feasible and infeasible space or when the feasible region is disjoint (Mallipeddi and Suganthan, 2010).

Michalewicz and Schoenauer (1996) raised the question of whether features of the problem can guide the choice of which constraint-handling technique is to be incorporated into an evolutionary algorithm. The objective of this study was to answer this question with the focus on problem landscape features measured through sampling of the search space. The study is significant as it covers problems and algorithmic approaches from both combinatorial and continuous domains. We provide empirical evidence that high-level landscape features can be used as a guiding principle when deciding on the appropriate use of constraint-handling techniques for a new problem. An additional contribution is the introduction of the Broken Hill Problem: a generator of combinatorial problems with tunable levels of complexity and constraint violation that could be used in further studies of constrained search spaces.

2 Background

In general, the challenge of selecting the most appropriate algorithm to solve a given problem was formulated by Rice (1976) and was applied in the context of optimisation by Smith-Miles (2008). Further studies have investigated solving aspects of the algorithm selection problem for optimisation using fitness landscape analysis (Bischl et al., 2012; Malan and Engelbrecht, 2014; I. Moser et al., 1997; Muñoz et al., 2013; Ochoa et al., 2014), but all of these studies have been restricted to unconstrained (or only bound constrained) problems.

Most real-world optimisation problems have constraints defined on the search space. Consider, for example, the 25 papers published in the "Real World Applications" session of the GECCO 2016 conference (Friedrich, 2016). The papers cover a range of interesting and complex optimisation problems including habitat restoration planning, optimisation of chemical processes, and office space allocation. Of the 21 papers that present solutions to optimisation problems, 16 are constrained problems, some with as many as ten different constraints.

Existing techniques for characterising fitness landscapes are not necessarily appropriate for constrained landscapes. For example, autocorrelation (Weinberger, 1990; Manderick et al., 1991) is frequently used to characterise landscape ruggedness, but this technique assumes that the problem landscape is statistically isotropic (statistical information is invariant with respect to the starting position of a random walk when it is long enough). If a penalty function is used to convert a problem from a constrained into an unconstrained problem, this can result in a landscape that is no longer statistically isotropic (Greenwood and Hu, 1998a) and can therefore give misleading results

on the ruggedness of a landscape (Greenwood and Hu, 1998b). Alternative approaches are needed for characterising constrained landscapes. Poursoltan and Neumann (2014) proposed a method for quantifying the ruggedness of constrained continuous landscapes. This method uses stochastic ranking (Runarsson and Yao, 2000) to perform a biased walk that favours feasible solutions. The measure of ruggedness, however, characterises only the fitness landscape.

Malan et al. (2015) proposed the notion of a violation landscape to be studied with fitness landscapes and introduced five metrics to capture its properties. In this study, we apply the same metrics to constrained problems in both continuous and combinatorial spaces, assess their capabilities in capturing the characteristics of the constrained landscape of a problem and relate the characteristics to six basic constraint handling techniques which are applied in conjunction with evolutionary metaheuristics.

3 Constrained Optimisation Benchmarks

In the continuous optimisation domain, a suite of constrained benchmark problems was developed and extended over the years for evolutionary algorithms (Michalewicz and Schoenauer, 1996; Koziel and Michalewicz, 1999; Runarsson and Yao, 2000) to finally form the basis of a competition of the IEEE Congress on Evolutionary Computation (CEC) in 2006 (Liang et al., 2006). A new set of 18 scalable problems were defined for the 2010 CEC competition (Mallipeddi and Suganthan, 2010) and these problems were used for the continuous problems in this study.

Benchmark problems with known and definable constrained areas do not exist in the combinatorial space. In combinatorial problems, the steps through the search landscape are naturally discrete, and gradients to optima can easily be devised. In this article, we introduce the Broken Hill Problem, an optimisation challenge where sequences of assignments can be defined such that each consecutive assignment adds to the value of the solution. At the same time, sequences can be defined as infeasible, rendering a solution invalid. In this way, it is possible to design instances that have longer or shorter gradients with longer or shorter infeasible sequences along these gradients. These instances can be used to test the ability of an algorithm to traverse shorter and longer infeasible areas.

Section 3.1 describes the Broken Hill Problem for tuning constrained combinatorial problems and Section 3.2 describes the CEC 2010 problem suite.

3.1 Broken Hill Problem

Devised as a test problem for constraint handling methods, the Broken Hill Problem (BHP) is an assignment problem with a sequence of locations to allocate items to. Items have a numerical value and duplicate assignments—assignments of items with the same value—are permitted. Consecutive allocations of the same item carries a reward, but constraints are applied to sequences of duplicated items of particular lengths.

A solution $\mathbf{x}$ of the BHP, with m items to be allocated to n locations, is defined as:

$$\mathbf{x} = (x_1, x_2, \ldots, x_n), \text{where } x_i \in \{1, 2, \ldots, m\}. \tag{1}$$

Given a solution $\mathbf{x}$, a *duplication sequence* is two or more successive occurrences of the same item in $\mathbf{x}$. For each solution $\mathbf{x}$, two structures are defined: a list S of all duplication sequences in $\mathbf{x}$ and a set I of all indices of $\mathbf{x}$ that are not part of any duplication sequence. For example, a solution $\mathbf{x} = (8, 3, 7, 7, 2, 5, 5, 5, 5, 8)$, would result in list $S = \{\{7, 7\}, \{5, 5, 5, 5\}\}$ and set $I = \{1, 2, 5, 10\}$.

Given S and I above, the fitness function for a BHP with m items to be allocated to n locations is defined as:

$$f(\mathbf{x}) = \sum_{s \in S} (|s|_k \cdot v(s))^{\frac{|s|}{k}+1} + \sum_{i \in I} \frac{x_i}{m \cdot n}, \tag{2}$$

where

- k is the optimal length of a duplication sequence (a parameter to be set by the problem designer);
- $|s|_k$ is the length of a duplication sequence s with a cap of k. If the number of items in s is larger than k, $|s|_k = k$; and
- $v(s)$ is the value of one of the repeating items in a duplication sequence s.

The first summation term of Eq. (2) rewards duplication sequences through the exponent (with a value always >1 and a maximum of 2). The largest possible exponent occurs when the sequence length equals or exceeds the optimal length of k, which ensures that more value is given to sequences closer to the optimal length k. The second summation term of Eq. (2) can be regarded as the base landscape that can be tuned to be neutral, guiding or deceptive. The current formulation is a guiding landscape where higher-valued items contribute more to the fitness. In this formulation, the value of the items that are not part of duplication sequences are added, but scaled by $m \cdot n$. This ensures that non-repeating items never dominate the fitness, since the highest-valued item carries a smaller weight than the lowest value when it repeats only once. A neutral base landscape can be formed by setting the second term to 0 and a deceptive landscape can be formed by rewarding lower valued non-repeating items.

Given the example of $\mathbf{x} = (8, 3, 7, 7, 2, 5, 5, 5, 5, 8)$, with $n = 10$, $m = 8$ and $k = 3$,

$$f(\mathbf{x}) = (2 \cdot 7)^{\frac{2}{3}+1} + (3 \cdot 5)^{\frac{3}{3}+1} + \frac{8+3+2+8}{80} \approx 81.32 + 225 + 0.26 = 306.58.$$

In this way longer duplication sequences (up to k) are rewarded exponentially more than shorter sequences. Non-repeating assignments do contribute to the fitness, but only a very small amount. The parameter k has the effect of creating multiple "hills" that serve as gradients needed for stochastic optimisers to search a fitness space. The optimal solution is formed by a repetition of duplication sequences of k items of maximal value (with value m), alternating with duplication sequences of k items of next-to-maximal value (with value $m - 1$). The fitness of the optimal solution is expressed by Eq. (3).

$$f^* = \sum_{i=1}^{r_1} (k \cdot m)^{\frac{k}{k}+1} + \sum_{i=1}^{r_2} (k \cdot (m-1))^{\frac{k}{k}+1} + r_3, \tag{3}$$

where

$$r_1 = \begin{cases} \left\lfloor \frac{n}{2k} \right\rfloor & \text{if } \left\lfloor \frac{n}{k} \right\rfloor \text{ is even} \\ \left\lfloor \frac{n}{2k} \right\rfloor + 1 & \text{if } \left\lfloor \frac{n}{k} \right\rfloor \text{ is odd} \end{cases}$$

$$r_2 = \left\lfloor \frac{n}{2k} \right\rfloor$$

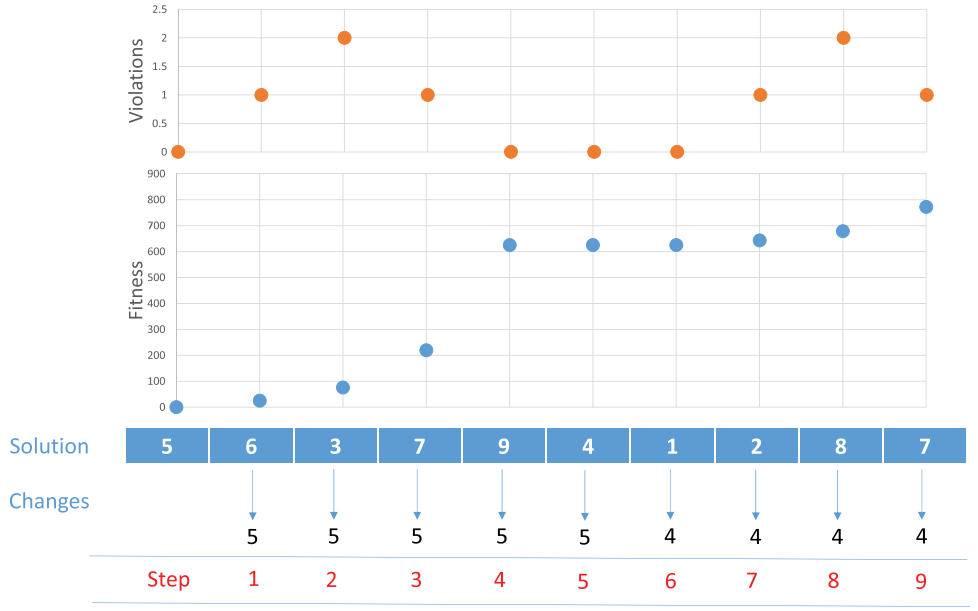


Figure 1: Development of fitness and constraint violations during a hill climb towards a local optimum with $c = 3$ and $k = 5$. The local optimum would be attained with a 10th step that changes the item 5 of step 5 back to an item 4.

$$r_3 = \begin{cases} ((n \bmod k) \cdot m)^{\frac{(n \bmod k)}{k}+1} & \text{if } (n \bmod k) > 1 \text{ and } \left\lfloor \frac{n}{k} \right\rfloor \text{ is even} \\ ((n \bmod k) \cdot (m-1))^{\frac{(n \bmod k)}{k}+1} & \text{if } (n \bmod k) > 1 \text{ and } \left\lfloor \frac{n}{k} \right\rfloor \text{ is odd} \\ \frac{m}{m \cdot n} & \text{if } (n \bmod k) = 1 \text{ and } \left\lfloor \frac{n}{k} \right\rfloor \text{ is even} \\ \frac{m-1}{m \cdot n} & \text{if } (n \bmod k) = 1 \text{ and } \left\lfloor \frac{n}{k} \right\rfloor \text{ is odd} \\ 0 & \text{otherwise.} \end{cases}$$

3.1.1 Constraints

As a benchmark problem for constraint-handling techniques, the BHP defines part of the path to an optimum as infeasible. The constrainedness variable c defines what subsequence of a duplication sequence is infeasible. The offset o permits the researcher to set the length at which infeasible duplication sequences start. More precisely, any solution containing a duplication sequence of a length in the range $o + 1, \ldots, o + c$ is defined as infeasible. For example, if $o = 2$ and $c = 3$, then any solution containing a duplication sequence of length 3, 4, or 5 is infeasible. For a "broken hill," it is desirable for k to be greater than $o + c$, so that the optimum is feasible and located beyond the feasible area.

Figure 1 illustrates an example path to a local optimum in a neutral base landscape (i.e., the second term of (2) is set to 0), with $k = 5$, $o = 1$, and $c = 3$. The blue dots show the fitness increasing as the solution changes (the "hill"), while the red dots show the constraint violations ("broken" parts of the hill). The violations are counted as the smallest distance to the feasible area, leading to two peaks at steps 2 and 8, where the solution

contains a duplication sequence of length 3 (in the middle of the range of infeasible sequences).

In Figure 1, the initial solution, (5, 6, 3, 7, 9, 4, 1, 2, 8, 7), contains no duplication sequences and the fitness and level of constraint violation is zero. Each of the steps 1–5 changes the assignments to a 5 which leads to six contiguous items of 5. Since $k = 5$, setting the sixth assignment to a 5 (step 5) has no effect on the fitness. Step 6 also has no effect on the fitness, since the 4 does not repeat.

Step 7 produces the second 4 in the sequence, and the fitness rises again. Steps 8 and 9 complete the sequence of four items of value 4. The hill climb does not end here, because it is possible to make one more improvement by exchanging the 5 made by step 5 to a 4 and achieve two sequences of five 5's and five 4's respectively. From this solution, it is not possible to reach the global best of five 9's and five 10's without making deteriorating moves.

The base landscape in the example is neutral. Note that defining it as guiding or deceptive does not change the (lack of) contribution to the objective function of the sixth five assigned in step 5, exceeding k.

3.2 Continuous Optimisation Benchmarks

In general, a constrained real-valued minimisation problem can be defined as follows:

$$\text{Minimise } f(\mathbf{x}), \quad \mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \in R^n \tag{4}$$

$$\text{subject to } \begin{array}{l} g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, p \\ h_j(\mathbf{x}) = 0, \quad j = p+1, \ldots, m, \end{array} \tag{5}$$

where $\mathbf{x}$ is an n-dimensional solution to the problem, $f(\mathbf{x})$ is the fitness function to be minimised, p is the number of inequality constraints, $g_i(\mathbf{x})$, and $m - p$ is the number of equality constraints, $h_j(\mathbf{x})$. Equality constraints are usually expressed as inequalities within an error margin, so a solution $\mathbf{x}$ is feasible if it satisfies all the inequality constraints and

$$|h_j(\mathbf{x})| - \epsilon \leq 0, \quad j = p+1, \ldots, m \tag{6}$$

for some small value of ϵ, such as 10^{-4}.

A number of continuous constrained optimisation benchmark suites have been proposed and extended over the years (Michalewicz and Schoenauer, 1996; Koziel and Michalewicz, 1999; Runarsson and Yao, 2000; Mezura-Montes and Coello Coello, 2004). For the CEC 2010 Competition on Constrained Real-Parameter Optimization (Mallipeddi and Suganthan, 2010), a new set of 18 problems were defined that were harder to solve than the previous benchmark problems and were scalable to any dimension. These problems were used in this study to generate the continuous problem instances.

3.3 Problem Instances

For the combinatorial domain, 70 BHP problem instances with varied sizes of infeasible areas and offset values were defined. The length of the solution (number of locations for assignment) was set as $n = 50$; and the number of options to assign, set as $m = \{8, 10\}$. These values were combined with infeasible ranges of $c = \{1, \ldots, 9\}$ and offsets of $o = \{1, \ldots, 5\}$ (stopping when $o + c = k$). The optimal length of duplication sequences k was set to 10 in all instances.

Small offsets mean that the infeasible range begins "early" in a path "up a hill"; large offsets allow the algorithm to build solutions of higher quality (beyond the quality of the base landscape) without having to "cross" the infeasible range defined by c.

The higher c, the longer the successive assignments that render a solution infeasible. For example, an instance with $o = 5$ and $c = 1$ would have small infeasible areas (solutions with duplication sequences of length 6), whereas an instance with $o = 1$ and $c = 9$ would have feasible solutions at the bottom of the hills (containing no duplication sequences) and all other solutions, including those with optimal fitness would be infeasible.

For the continuous domain, the 18 CEC 2010 problems (Mallipeddi and Suganthan, 2010), which are problems of the form of Eq. (4), were defined in 2, 10, 20, and 30 dimensions, resulting in 72 problem instances. The problems have different objective functions and numbers of inequality and equality constraints and for most problems, the constraints are rotated to prevent feasible patches that are parallel to the axes. Optimal solutions are not reported for these functions, so the quality of solutions returned by algorithms is reported in relation to the quality of solutions reported by other algorithms.

All of the CEC 2010 problems with equality constraints have a reported feasibility region in 10 and 30 dimensions of approximately zero in relation to the size of the search space. This does not imply that there are no feasible solutions. Since equality constraints are commonly re-expressed as inequality constraints within an error margin, each equality constraint effectively defines a very narrow margin of feasibility around the equality constraint function. These regions are so narrow that they do not significantly contribute to the size of the feasible set in relation to the overall size of the search space.

4 Constrained Landscape Characterisation

The notion of a violation landscape was introduced by Malan et al. (2015) as a complementary viewpoint to fitness landscapes for constrained optimisation problems. Stadler (2002) defines a fitness landscape as consisting of three elements: a set X of solutions to the problem; a notion of neighbourhood, nearness, distance, or accessibility on the set; and a fitness function. A violation landscape simply replaces the fitness function with a violation function, $\phi : X \rightarrow R$, which quantifies the level of constraint violation for all solutions.

For the BHP, the level of constraint violation of a solution $\mathbf{x}$ containing the list of duplication sequences S, is defined by Eq. (7), which sums the violations for all duplication sequences.

$$\phi(\mathbf{x}) = \sum_{s \in S} \phi(s), \tag{7}$$

where

$$\phi(s) = \begin{cases} |s| - o, & \text{if } 0 < (|s| - o) \leq \left\lceil \frac{c}{2} \right\rceil \\ c - (|s| - o) + 1, & \text{if } \left\lceil \frac{c}{2} \right\rceil < (|s| - o) \leq c \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

and $|s|$ is the length of a duplication sequence s.

The level of constraint violation of a solution should ideally reflect the distance to feasible space when used in penalty functions in order to direct the search algorithm towards feasible regions (Richardson et al., 1989). For this reason, Eq. (8) defines ϕ such that it is highest in the middle of the range defined by c.

For the continuous problems, the level of constraint violation was defined as follows (Mallipeddi and Suganthan, 2010):

$$\phi(\mathbf{x}) = \frac{\sum_{i=1}^{p} G_i(\mathbf{x}) + \sum_{j=p+1}^{m} H_j(\mathbf{x})}{m}, \tag{9}$$

where

$$G_i(\mathbf{x}) = \begin{cases} g_i(\mathbf{x}) & \text{if} \quad g_i(\mathbf{x}) > 0 \\ 0 & \text{if} \quad g_i(\mathbf{x}) \leq 0 \end{cases}, \tag{10}$$

and

$$H_j(\mathbf{x}) = \begin{cases} |h_j(\mathbf{x})| & \text{if} \quad |h_j(\mathbf{x})| - \epsilon > 0 \\ 0 & \text{if} \quad |h_j(\mathbf{x})| - \epsilon \leq 0 \end{cases}, \tag{11}$$

and ϵ is defined as 10^{-4}.

The following sections describe the metrics (proposed by Malan et al. (2015)) used to characterise the constrained space.

4.1 FsR Metric

The feasibility ratio (FsR) approximates the size of the feasible space in relation to the entire search space. The metric is based on a sample of n solutions and is defined as:

$$\text{FsR} = \frac{n_f}{n}, \tag{12}$$

where n_f is the number of points in the sample that are feasible.

4.2 RFB× Metric

The ratio of feasible boundary crossings (RFB×) quantifies how disjoint the feasible regions are. Based on walks through the search space, RFB× counts the proportion of steps that cross between feasible and infeasible space. More formally, given a sequence of n solutions, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ obtained by a walk, the binary string $\mathbf{b} = b_1, b_2, \ldots, b_n$ is defined such that $b_i = 0$ if $\mathbf{x}_i$ is feasible and $b_i = 1$ if $\mathbf{x}_i$ is infeasible. RFB× is then defined as:

$$\text{RFB}\times = \frac{\sum_{i=1}^{n-1} cross(i)}{n-1} \tag{13}$$

where

$$cross(i) = \begin{cases} 0 & \text{if } b_i = b_{i+1} \\ 1 & \text{otherwise.} \end{cases} \tag{14}$$

Note that if FsR is 0 (no feasible solutions in the sample) or 1 (all feasible solutions in the sample), then RFB× based on the same sample will be 0.

4.3 FVC Metric

The fitness violation correlation (FVC) quantifies the extent to which the fitness and violation guide search in a similar direction. Based on a sample of solutions resulting in fitness-violation pairs, the FVC is calculated as the Spearman's rank correlation coefficient between the fitness and violation values. For maximisation problems, the fitness values are negated before calculating FVC, so that the metric can be interpreted in the same way for both minimisation and maximisation problems.

The range of FVC values is $[-1, 1]$, where a value of 1 indicates that fitness and violation guide search in the same direction, whereas a value of -1 indicates that the fitness and violation landscapes guide search in maximally opposite directions.

4.4 IZ Metrics

If one considers the scatterplot of fitness-violation pairs of a sample of solutions, the "ideal zone" would be the area of this plot where fitness is good and violations are low (bottom left corner for a minimisation problem). A large proportion of solutions in the ideal zone could indicate relatively large basins of attraction around the better solutions in a landscape combining fitness and violation functions (as with a penalty-based approach). On the other hand, a small proportion of solutions in the ideal zone could indicate narrow basins of attraction (isolated optima) in a penalised fitness landscape. The IZ metrics quantify the proportion of points in a sample that are in two of these ideal zones.

Metric 25_IZ is defined as the proportion of points in a sample that are below the 50% percentile (the median) for both fitness and violation for minimisation problems, and above the 50% percentile for fitness and below the 50% percentile for violation for maximisation problems. If the sample of points are distributed evenly throughout the fitness-violation scatterplot, 25_IZ would have a value of 0.25.

Metric 4_IZ is defined as the proportion of points in a sample that are below the 20% percentile for both fitness and violation for minimisation problems, and above the 20% percentile for fitness and below the 20% percentile for violation for maximisation problems. If the sample of points are distributed evenly throughout the fitness-violation scatterplot, then the 4_IZ would have a value of 0.04.

5 Constraint Handling Techniques

This section specifies the base algorithms used to solve the combinatorial and continuous problems and describes the six constraint handling approaches used in this study.

5.1 Base Optimisation Algorithms

The aim of this study was to investigate different constraint handling techniques as abstractions from the underlying optimisation approach. To isolate the effect of the constraint handling from the underlying search algorithm, the same base algorithm was used for all combinatorial problems and for all continuous problems, namely a genetic algorithm (GA) for the combinatorial problems and a differential evolution (DE) algorithm (Storn and Price, 1995) for the continuous problems.

For the optimisation of the combinatorial BHP, we chose a simple generational EA with a population size of 100, a mating pool of 50% formed using binary tournament ensuring no duplication, uniform crossover with a probability of 0.01 and a mutation rate of $\frac{1}{n}$. The initial population was created with uniform random assignments. For selection, the solutions were valued as prescribed by the respective constraint handling technique. The algorithm stopped when a budget of function evaluations had been spent.

For the continuous problems, the classic form of DE, *DE/rand/1* (Storn and Price, 1996), was used as the base algorithm with uniform crossover, a population size of 100, a scale factor (F) of 0.5, and a crossover rate of 0.5.

The base algorithms were combined with one of the following six constraint handling techniques.

5.2 No Constraint Handling

The simplest implementation disregards constraints in the hope to discover a good-quality solution which happens to be feasible. For selection, the solutions are compared by fitness alone. This approach is abbreviated to NCH.

5.3 Death Penalty

The death penalty (DP) rejects infeasible solutions. For the combinatorial problems, DP was implemented through the selection operator by removing all infeasible solutions from the population. Some BHP configurations (e.g., low values of o) lead to high levels of infeasibility. For this reason, the algorithm ensures that the population is valid after initialisation.

For the continuous problems, many of the instances have a zero feasibility ratio. It was therefore not possible to ensure a valid initial population in all cases. For this reason, DP was implemented in the continuous domain by assigning all infeasible solutions a constant maximum fitness value (the maximum real value using a double representation as defined by the compiler).

5.4 Weighted Penalty

The weighted penalty (WP) approach combines the level of constraint violation as a penalty in the fitness function using a static weighting between the two components. In this study, an even weighting of 50% penalty and 50% fitness was applied.

5.5 Feasibility Ranking

Deb (2000)'s feasibility ranking approach uses different comparison criteria depending on the feasibility and is implemented as follows:

- Two feasible solutions are compared by fitness.
- A feasible solution is preferred to an infeasible one.
- Two infeasible solutions are compared by their level of constraint violation.

This approach is abbreviated to FR.

5.6 ϵ-Based Feasibility Ranking

Takahama and Sakai (2005) published an extended feasibility ranking with an ϵ tolerance that reduces over time, denoted ϵFR:

- Two solutions whose constraint violations are lower or equal to ϵ are compared by their fitnesses $f(\mathbf{x})$.
- Two solutions whose violations $\phi(\mathbf{x})$ are equal are compared by their fitnesses $f(\mathbf{x})$.
- All other solutions are compared by their feasibility violations $\phi(\mathbf{x})$.

The most informative description of how to set and adapt the parameter ϵ is given in a later publication (Takahama and Sakai, 2006). It defines a cut-off, T_c, of generations after which $\epsilon = 0$. The cut-off can be set in the range $0.1, \ldots, 0.8 \times T_{max}$, with T_{max} representing the maximum number of generations. Following Takahama and Sakai (2006), the cut-off was defined in terms of the maximum function evaluations (FE_{max}) as $FE_c = 0.8 \times FE_{max}$. Before FE_c had been reached, ϵ was reset whenever a comparison was made between solutions as follows

$$\epsilon = \phi(\mathbf{x}_\theta) \times (1 - \frac{FE_i}{FE_c})^{cp}, \tag{15}$$

where x_θ is the θ-th solution in a population ordered by violations $\phi(\mathbf{x})$ from lowest to highest (least to most infeasible), FE_i is the current number of function evaluations, and cp is a parameter to control the speed of reducing relaxation of constraints. Following Takahama and Sakai (2006), θ was set to $0.8 \times popsize$ and cp was set to 5.

5.7 Bi-Objective Formulation

An alternative approach to a penalty or a feasibility ranking method is to model the constraint violations as a second objective, abbreviated to BO (Bi-objective).

The non-dominated sorting procedure of NSGA II (Deb et al., 2002) was used for the ranking of solutions during mating pool selection for the GA and for the selection of the next generation from the current population and trial vector population for the DE. Ties were broken randomly. The generational selection was carried out as described by Deb et al. (2002): successive fronts were added until the prescribed population size had been attained.

6 Results

This section provides results on the performance of the different constraint-handling approaches on the problem instances and then characterises the problem instances using the landscape metrics described in Section 4.

6.1 Algorithm Performance

The purpose of the first sets of experiments was to determine whether there are differences in the performance of the constraint-handling techniques on the benchmark problems. The expectation was that different algorithms would be suited to different problems, giving a range of performances.

Thirty independent runs of each algorithm were conducted on each problem instance. The budget of function evaluations was set to $20000 \times D$ (the dimension) for the continuous problems and 500000 for the discrete problem instances. The performances of algorithms were compared according to the CEC competition rules (Mallipeddi and Suganthan, 2010):

- If two algorithms produced feasible solutions in all of the 30 runs, they were compared by the mean fitness of the best feasible solutions produced in the final generation of each run.
- If two algorithms produced feasible solutions in some or all of the runs, they were compared by the proportion of runs that produced feasible solutions (known as the success rate).
- If two algorithms had zero runs with feasible solutions (a success rate of 0), the algorithms were compared by the mean of the violations of the least infeasible solutions in the final generation of each run.

For each problem instance, the algorithms were assigned a rank based on the relative performance using the rules above. Two algorithms were considered different if they differed in the third decimal. Tied algorithms received the same rank, and the subsequent rank remained unassigned. For example, Table 1 shows the ranks assigned to six constraint-handling techniques for two problem instances of the CEC2010 suite. For C01 in 2 dimensions, three of the techniques (DP, FR, and ϵFR) achieved a success rate of 1

Table 1: Ranking of algorithms based on performance measures of two example problem instances from the CEC2010 problem suite.

	CEC2010 problem C01 in 2 dimensions					
	NCH	DP	WP	FR	ϵFR	BO
Success rate	0	1	0	1	1	0
Mean fitness of successful runs	n/a	−0.365	n/a	−0.362	−0.365	n/a
Mean violation	0.320	0	0.179	0	0	0.314
Algorithm Rank	6	1	4	3	1	5
	CEC2010 problem C09 in 20 dimensions					
	NCH	DP	WP	FR	ϵFR	BO
Success rate	0	0	0.033	0	0	0.967
Mean fitness of successful runs	n/a	n/a	0.000	n/a	n/a	0.000
Mean violation	8.650	703.648	4.187	0.278	0.257	0.037
Algorithm Rank	5	6	2	4	3	1

(all 30 runs resulted in feasible solutions). Comparing the mean fitness of these algorithms, DP and ϵFR both achieved the lowest value of −0.365, so these two algorithms share the rank of 1 and FR is ranked 3. The remaining three algorithms are assigned the ranks of 4, 5, and 6 based on mean violation.

On problem C09 in 20 dimensions (the second example in Table 1), the BO algorithm achieved a success rate of 0.967 (29 of the 30 runs found a feasible solution), the WP algorithm achieved success rate of 0.033 (only 1 of the 30 runs found a feasible solution), and the other algorithms achieved success rates of 0. The BO and WP are compared based on their success rates and so BO is assigned rank 1 and WP rank 2. The remaining algorithms are ranked based on the mean violation, with DP achieving the worst rank of 6.

The performance of the different algorithms on all 142 problem instances (70 BHP and 72 CEC2010 instances) in terms of ranks is summarised by the boxplot in Figure 2. The whiskers on the plots show that the range of all algorithms is 1 to 6, meaning that each algorithm performed the best and also the worst in at least one problem instance. The median lines inside the boxes show that NCH performed the worst overall with a median rank of 6, whereas BO performed the best overall with a median rank of 2.

An alternative view of performance is to consider whether an algorithm failed or succeeded on a problem instance. Considering the first example in Table 1, the three algorithms with a success rate of 1 have very similar mean fitness values and can all three be regarded as reasonably successful. In contrast, the other three algorithms clearly failed on this problem in comparison to the other approaches.

In this study, an algorithm was regarded as successful on a problem instance if it met either of the two criteria below:

(1) It was the best-performing algorithm as defined by the sub-rules 1a–1c below.
   (a) If two algorithms had a success rate of ≥ 0.5, the better algorithm was decided by comparing the mean fitnesses of all feasible runs.

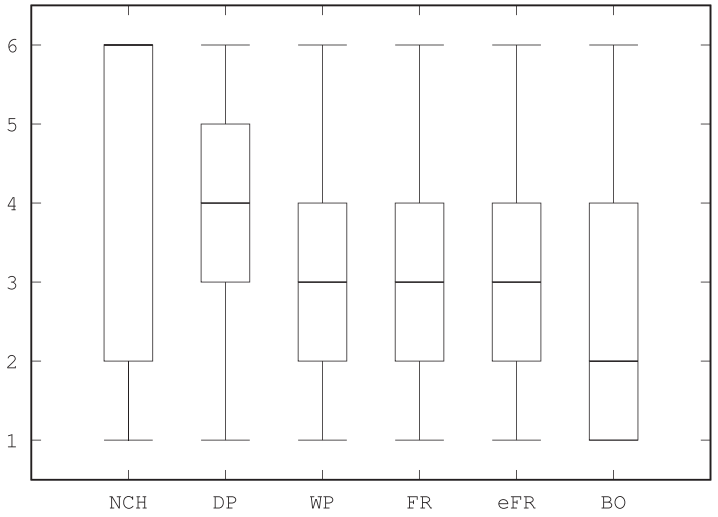


Figure 2: Distribution of algorithm ranks over all problem instances.

Table 2: Percentage of problem instances on which each algorithm failed.

NCH	DP	WP	FR	ϵFR	BO
68%	81%	65%	64%	65%	42%

(b) If either algorithm had a success rate of ≥ 0.5, while the other had a success rate < 0.5, the one with the higher success rate was considered better.
(c) If both algorithms had a success rate of < 0.5, the better algorithm was decided by comparing the mean violations of all runs.

(2) It was within 10^{-1} of the best algorithm on the measure used to compare it as defined by the sub-rules 1a–1c above (mean fitness, success rate, or mean violation).

Considering the examples in Table 1 and using the rules above, DP, FR and ϵFR were labelled as successful on CEC2010 C01 in 2 dimensions and the rest failures, while only BO was labelled as successful on CEC2010 C09 in 20 dimensions.

For all 142 problem instances, the proportion of instances on which each algorithm failed is given in Table 2. From the viewpoint of success/failure, the best performing algorithm was BO, which was the same as in Figure 2 based on the distribution of ranks. The worst performing algorithm was DP, followed by NCH. WP, FR, and ϵFR had very similar failure rates.

In conclusion, the following can be derived from the experiments in this section: although the bi-objective approach performed the best on average on all problems considered, all algorithms achieved some level of success in solving the problems. In particular, each algorithm performed both the worst and the best on at least one problem instance. We conclude that each problem instance can be matched to the most suitable algorithm(s) from the algorithms considered.

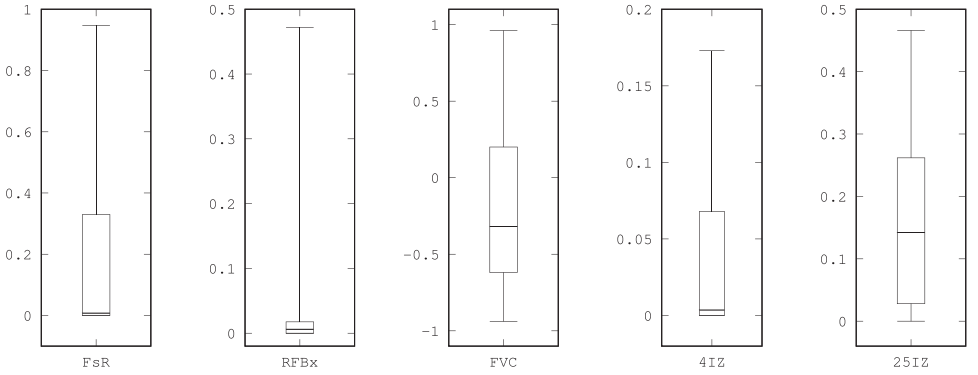


Figure 3: Boxplots showing the distribution of values of metrics for the 142 problem instances.

6.2 Benchmark Problem Characterisation

To characterise all 142 problem instances, samples were generated for each instance using multiple hill climbing walks. Hill climbing walks were chosen over random walks to ensure that the sample contained a mixture of solutions of different quality. In contrast to hill climbing walks, random walks in the BHP search space frequently result in an over-representation of solutions on the "lower levels" of the fitness landscape (since the probability of attaining contiguous identical assignments declines exponentially in random solutions). This is in line with the findings of Smith-Miles (2008) that random sampling may be less successful in describing the landscape, particularly with skewed fitness functions.

During the hill-climbing walks, at least 5000 solutions (1% of the computational budget of actually solving the problem) were created for each BHP problem instance and these samples were used as the basis for the five landscape metrics (FsR, RFB×, FVC, 25_IZ, and 4_IZ). For each instance, initial random solutions were generated and locally optimised until no further improvement was possible. This process was repeated until the sample size exceeded 5000, ensuring that the last walk was completed.

For the characterisation of continuous problem instances, a sample size of $200 \times D$ (1% of the computational budget of actually solving the problem) was used. From a random initial position, a basic hill climbing walk was executed. Neighbours were formed by sampling in each dimension from a Gaussian distribution with the current position as mean and a standard deviation of 5% of the range of the domain of the problem instance. If no better neighbour could be found after sampling 100 random neighbours, the walk was terminated. New walks were started from random positions, until the total number of points in all the walks equalled the sample size.

Figure 3 shows the distribution of the landscape metric values for all 142 problem instances. The first boxplot shows that the FsR values ranged from 0 to close to 1, but that the values were strongly skewed towards 0. The median value for FsR was 0.008, indicating that more than half of the problem instances had feasibility ratios of 0 or close to 0. Most of the FsR values for the BHP instances (62 of 70 instances) were greater than 0, indicating larger proportions of feasibility encountered than for the CEC2010 instances (for which 47 of 72 instances had FsR values of 0). Most of the CEC2010 problems (12 of 18) have equality constraints, which explains the low feasibility rates.

Table 3: Spearman's correlation coefficients between algorithm performances (success/failure) and landscape metrics for all problem instances.

	FsR	RFBx	FVC	4_IZ	25_IZ
NCH	0.13	0.26	0.40	0.26	0.23
DP	0.41	0.40	−0.12	−0.33	−0.33
WP	−0.19	−0.10	0.31	0.21	0.32
FR	−0.09	−0.09	0.08	0.01	0.02
ϵFR	−0.04	−0.06	0.10	0.05	0.03
BO	0.17	0.10	−0.10	0.13	−0.15

The RFB× values in the second boxplot are similarly skewed towards zero (since an instance with a 0 feasibility ratio would also have a 0 value for RFB×) with a median of 0.006 and a maximum of 0.472. The highest RFB× value for a BHP instance was 0.048, which indicates that the feasible solutions are generally clustered together in the search space, rather than distributed in disjoint areas as with some of the CEC2010 instances, such as the CEC2010 problem C08 in 10 dimensions which had a RFB× value of 0.472.

The third boxplot in Figure 3 shows that the FVC values ranged from close to −1 to close to 1, with values slightly skewed towards the negative (indicating that there were more problems instances where the fitness and violation landscapes diverged).

The last two boxplots in Figure 3 show that the maximum 4_IZ value was below 0.2 and the maximum 25_IZ value was below 0.5. The median of 0.0035 for 4_IZ and 0.14 for 25_IZ indicates that more than half of the problem instances had 0.35% or less of the sample in the 4% ideal zone and 14% or less in the 25% ideal zone, respectively.

7 Linking Performance to Characteristics

The purpose of the investigation was to establish whether the problem characteristics allow any conclusions as to the difficulty of a problem for the given constraint-handling technique. To this end, the problem characteristics were investigated in relation to the performances of the algorithms.

The Spearman's correlation coefficients between the metrics and the performances of the algorithm (with success coded as 1 and failure coded as −1) are shown in Table 3. Stronger correlations (positive or negative) are shaded darker.

DP has a high positive correlation with FsR and RFB×. This means that DP failure is associated with low feasibility ratios (and hence low RFB× values), which is understandable since DP either disregards or maximally penalizes infeasible solutions. NCH is positively correlated with FVC. That is, NCH, which ignores constraints and optimises only on fitness, does well if the fitness and violation are correlated. FR, ϵFR, and BO have weak correlations with the metrics.

Due to the large number of problem instances with zero feasibility ratios and corresponding zero values for RFB×, the dataset was divided into two: (1) a dataset with an FsR = 0 (55 problem instances: 47 continuous and 8 discrete), referred to as dataset DS_NoFeas, and (2) a dataset with the instances that scored above 0 for FsR (87 instances: 25 continuous and 62 discrete), referred to as dataset DS_Feas. The DS_Feas dataset therefore consists of problem instances that have both feasible and infeasible regions in the search space, whereas the DS_NoFeas dataset contains problem instances with no measurable feasibility and so there is a high probability that some algorithms

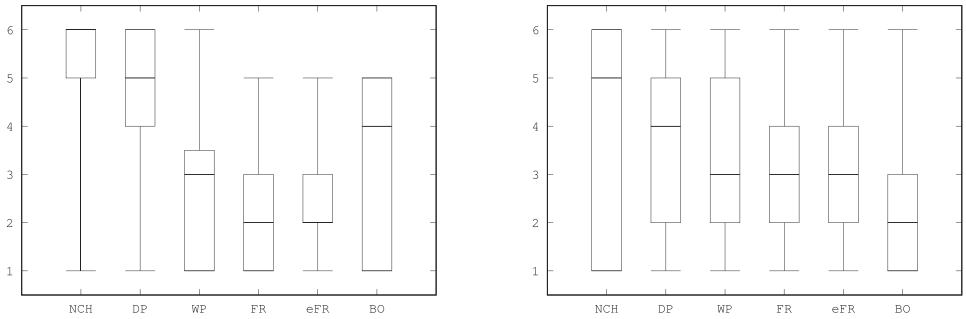
Table 4: Spearman's correlation coefficients between algorithm performances (success/failure) and landscape metrics for the split datasets.

(a) Correlations for dataset DS_NoFeas.

	FVC	4_IZ	25_IZ
NCH	0.43	0.49	0.40
DP	−0.31	−0.23	−0.31
WP	0.02	0.00	0.02
FR	−0.06	−0.10	−0.06
ϵFR	−0.01	−0.04	0.01
BO	0.35	0.57	0.30

(b) Correlations for dataset DS_Feas.

	FsR	RFBx	FVC	4_IZ	25_IZ
NCH	−0.16	0.14	0.51	0.36	0.37
DP	0.31	0.30	0.01	−0.28	−0.22
WP	−0.19	−0.07	0.43	0.24	0.36
FR	0.14	0.13	0.09	−0.06	−0.08
ϵFR	0.08	0.02	0.11	0.03	−0.05
BO	−0.11	−0.30	−0.28	0.15	−0.14



(a) Performance of algorithms on DS_NoFeas.

(b) Performance of algorithms on DS_Feas.

Figure 4: Distribution of performance of algorithms (in terms of algorithm rank) on the split dataset.

will be unable to find any feasible solutions. The correlations of algorithm performance to the split datasets are shown in Table 4. For the DS_NoFeas dataset, the metrics FsR and RFB$\times$ are left out, because these metrics are zero for all instances in this dataset.

Table 4 shows some stronger correlations between the metrics and algorithm performance than for the full dataset in Table 3. In particular, medium positive correlations between all three metrics and the NCH and BO algorithms can be seen for the DS_NoFeas dataset. BO correlates more strongly with 4_IZ when there are hardly any feasible solutions. If there are measurable numbers of feasible solutions, it correlates more, but negatively, with RFBx. When hardly any feasible solutions exist, bi-objective approaches cannot afford to be guided by a violation measure that distracts from the few solutions with higher fitness, hence the high correlation with 4_IZ. When there are a number of feasible solutions, BO does not depend on them residing in the same ideal zone, as long as they are not separated by infeasible areas, hence the stronger correlation with RFBx. While BO correlates negatively with FVC, WP has a positive correlation on the DS_Feas dataset. Combining fitness and violation into one function naturally benefits from them being correlated. As expected, the performance of a biobjective formulation improves when the objectives truly conflict.

The performance of each of the algorithms in terms of ranks on the two split datasets is summarised as boxplots in Figure 4 and in terms of percentages of failed instances

Table 5: Percentage of problem instances on which each algorithm failed.

Dataset	NCH	DP	WP	FR	ϵFR	BO
DS_NoFeas	82%	96%	56%	55%	60%	56%
DS_Feas	60%	71%	70%	70%	68%	32%

Table 6: Ranges of metric values for the DS_NoFeas dataset used as the basis for fuzzy interpretation of rules derived from the decision trees.

	Low	Low-Medium	Medium-High	High
FVC	$[-0.889, -0.506)$	$[-0.506, -0.120)$	$[-0.12, 0.351)$	$[0.351, 0.961]$
4_IZ	$[0, 0.004)$	$[0.004, 0.046)$	$[0.046, 0.077)$	$[0.077, 0.173]$
25_IZ	$[0.03, 0.161)$	$[0.161, 0.229)$	$[0.229, 0.305)$	$[0.305, 0.466]$

in Table 5. The relative performance of the algorithms differs between the two datasets. The best performing algorithm on DS_NoFeas is FR with a median rank of 2 (equal to ϵFR, but with a distribution slightly more skewed toward 1) and the lowest failure rate of 55%. The best performing algorithm on DS_Feas is clearly BO with the lowest median rank of 2 and by far the lowest failure rate of 32%. Even in the case of DS_NoFeas, BO comes a close second with 56%, on a par with WP.

To further investigate the relationship between problem characteristics and algorithm performance, the C4.5 decision tree induction algorithm (Quinlan, 1993), implemented in WEKA (Hall et al., 2009) as J48, was applied to classify the performances of the algorithms on the two datasets.

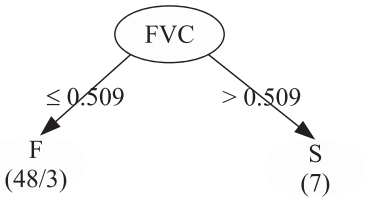
The parameter values of the WEKA J48 classifier were set to a confidence threshold of 25% (the default) and the minimum number of instances per leaf nodes was adjusted per experiment to reduce the size of the tree and reduce overfitting. Accuracies are reported using 10-fold cross validation.

7.1 Dataset with FsR Equal to Zero

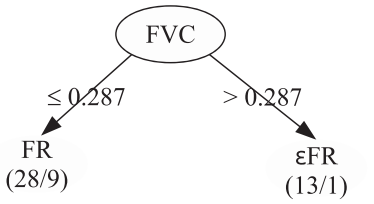
The DS_NoFeas dataset (with zero values for FsR) consisted of 55 instances with three landscape metrics (FVC, 4_IZ, and 25_IZ) and S/F class labels for each of the five constraint handling techniques. To reason about the metrics in relation to performance, the categories of "Low," "Low-Medium," "Medium-High," and "High" were defined based on the ranges: (1) minimum to lower quartile, (2) lower quartile to median, (2) median to upper quartile, and (4) upper quartile to maximum of each of the metrics. These ranges are summarised in Table 6.

The decision tree models generated by the C4.5 algorithm are given in Figure 5. In the visualisations of the trees, the splitting values are rounded off to three decimal places. For each tree, the testing accuracy based on 10-fold cross validation is reported below the tree. The total number of instances from the dataset reaching each leaf node is indicated in parentheses below the node. The number of instances that are incorrectly classified by the model, if any, are indicated after the slash in the parentheses.

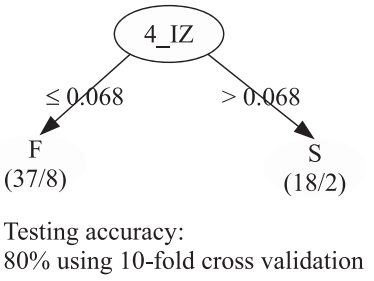
The first tree in Figure 5a predicts the success of the NCH algorithm. The tree can be interpreted in the following fuzzy terms (based on the ranges in Table 6): *When FsR*



(a) Success/Failure tree for NCH.



(b) Comparison tree for ϵFR vs. FR.



(c) Success/Failure tree for BO.

Figure 5: Classification trees created by the C4.5 algorithm based on the dataset DS_NoFeas.

is zero, if the fitness and violation are very highly correlated, then NCH will probably succeed; otherwise it will probably fail. This is understandable, since the NCH algorithm considers fitness alone, so it will only succeed in general if the fitness and the violation landscapes guide search in the same direction.

The second algorithm, DP, fails in 96% of the cases in the DS_NoFeas dataset and so is simply predicted to fail if FsR is zero. Since the death penalty eliminates infeasible solutions immediately after their creation (in the discrete domain) and assigns the maximum penalty to all infeasible solutions in the continuous domain, the algorithm has no opportunity to detect better solutions and is outperformed by the other algorithms.

In the case of the WP and FR algorithms, no success/failure decision tree could be induced, so it can be concluded that the failure of WP and FR on instances with FsR $= 0$ could not be predicted from the landscape metrics. Likewise, in the case of the ϵFR algorithm, no decision tree could be induced. However, since the performance of ϵFR was so similar to the performance of FR (as seen in Figure 4a), an alternative classification was performed. Figure 5b shows a tree based on a subset of the DS_NoFeas dataset where the algorithm performance ranks differed between ϵFR and FR. For each instance, if the algorithm rank (a value from 1–6) of ϵFR was better than the rank of FR, then the final class was denoted ϵFR and vice versa for FR. The leaf nodes therefore denote the better performing algorithm of the two. In fuzzy terms the tree in Figure 5b can be interpreted as: *When FsR is zero, if FVC is not High, then FR will probably perform better than ϵFR.* This can be understood as follows: when there are basically no feasible areas, FR only searches using the level of violation, while ϵFR first searches based on

Table 7: Ranges of metric values for the DS_Feas dataset used as the basis for fuzzy interpretation of rules derived from the decision trees.

	Low	Low-Medium	Medium-High	High
FsR	[0, 0.041)	[0.041, 0.267)	[0.267, 0.488)	[0.488, 0.961]
RFB×	[0, 0.008)	[0.008, 0.014)	[0.014, 0.026)	[0.026, 0.472]
FVC	[−0.938, −0.686)	[−0.686, −0.383)	[−0.383, −0.006)	[−0.006, 0.889]
4_IZ	[0, 0]	[0, 0]	[0, 0.02)	[0.02, 0.146]
25_IZ	[0, 0.006)	[0.006, 0.055)	[0.055, 0.231)	[0.231, 0.437]

fitness (when ϵ is large) and then later switches to search based on the level of violation (when ϵ is small). When the chances of finding a feasible solution are low, a search based on both fitness and violation (ϵFR) will probably only perform better when the fitness and the violation direct search in the same direction.

The success/failure tree of the BO algorithm is shown in Figure 5c. In fuzzy terms, this can be interpreted as: *When FsR is zero, if the proportion of points in the 4% ideal zone is Medium-High to High, then BO will probably succeed, otherwise it will fail.* A value of over 6.8% for 4_IZ means that proportionately more of the points in the sample had both the best fitness and the lowest constraint violation. In general, BO succeeded on these problem instances, but not on the others. Since there are basically no feasible areas, the other algorithms that only searched based on the level of violation (such as FR) performed better by not trying to also optimise based on fitness.

7.2 Dataset with Non-Zero FsR

The DS_Feas dataset (with FsR $\neq$ 0) consisted of 87 problem instances. Based on the failure percentages (Table 5), BO with 32% failures was a clear winner for the DS_Feas dataset. To investigate this behaviour further, a failure/success decision tree was induced for the BO algorithm and the remaining algorithms were analysed in terms of relative performance to the BO algorithm. The ranges of landscape metric values on this dataset are given in Table 7 and the classification trees generated by the C4.5 algorithm are illustrated in Figure 6.

Figure 6a gives the success/failure tree induced by the C4.5 algorithm for BO on instances with some proportion of feasibility in the sample. In fuzzy terms the tree can be interpreted as: *If the feasible areas are highly disjoint, then BO will probably fail; otherwise it will probably succeed.* Recall that the RFB× metric quantifies the proportion of boundary crossings between feasible and infeasible areas during the walks. When RFB× is high, this means that the violation landscape is characterised by multiple small feasible areas (flat sections with zero violation) between small non-feasible areas (with non-zero levels of constraint violation). The BO algorithm optimises the fitness and the level of constraint violation as separate objectives. To this algorithm, the violation landscape is a highly rugged landscape with numerous global optima (all the flat sections) spread throughout the search space. It can therefore be understood that on such problems, BO does not perform well, because the violation landscape does not help to guide search to the best solutions.

The remaining trees in Figure 6 show when the other algorithms performed better than BO. Figure 6b and 6c show that the NCH and DP algorithms performed better than the BO algorithm in general when the feasible areas were disjoint. This corresponds

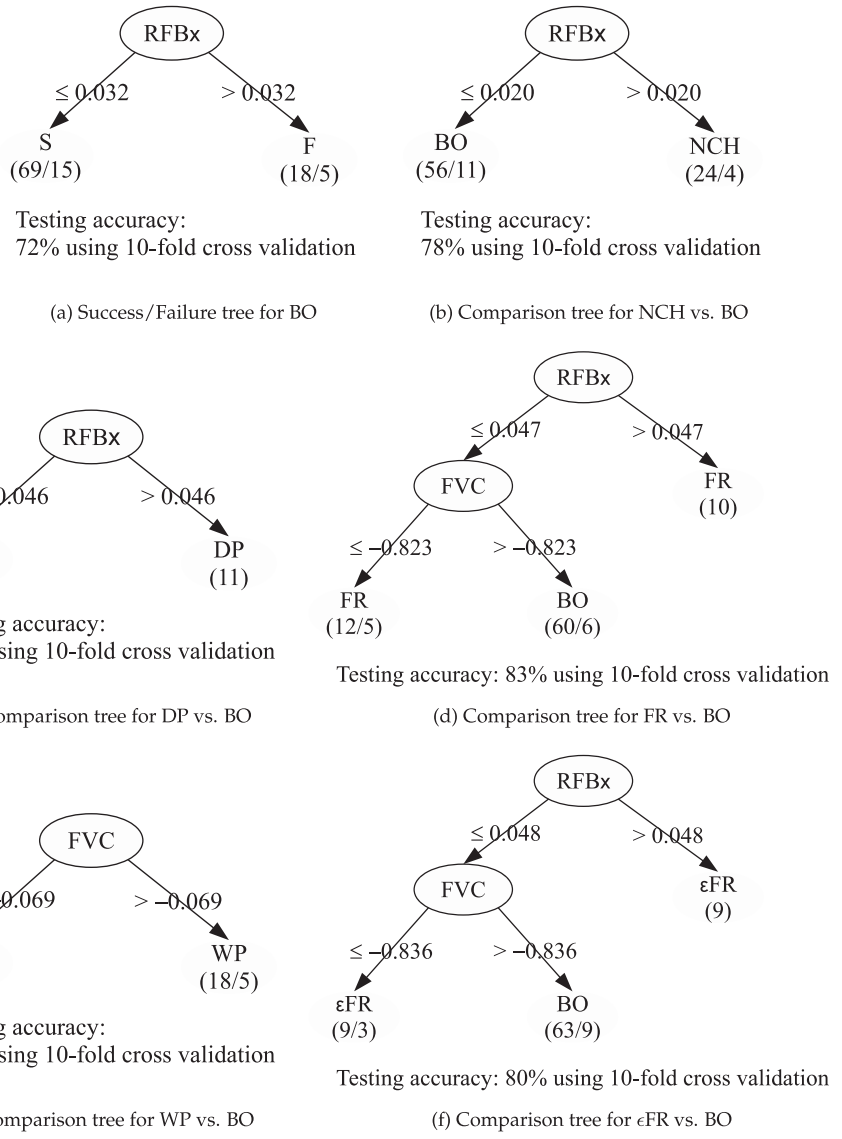


Figure 6: Classification trees created by the C4.5 algorithm based on the dataset DS_Feas.

with the failure graph for BO in Figure 6a and shows that even ignoring the constraints completely or killing off infeasible solutions were superior strategies to a bi-objective approach for these problem instances.

Figure 6e shows that the WP approach in general outperformed BO when the FVC was fairly high to high—in this case, not negatively correlated. A weighted penalty approach adds the level of constraint violation to the fitness value as a single-objective formulation. It can be understood that such an approach does not do well when the fitness and violation are negatively correlated, since the fitness and violation would essentially cancel each other out in the penalty function.

Similar to NCH and DP, the FR approach performs better than BO when feasible areas are very disjoint (Figure 6d). An FR approach always prefers a feasible solution to an infeasible solution and only considers the level of violation when comparing two infeasible solutions. The algorithm is therefore unaffected by the many flat sections of the violation landscape in the way the BO algorithm is affected. Figure 6d shows that FR also generally outperforms BO when FVC is very low (highly negatively correlated). Since the FR algorithm only considers either fitness or violation at each comparison, it is not as affected as BO by these two objectives guiding search in opposite directions. The tree generated to compare ϵFR and BO (Figure 6f) is the same structure as for FR vs. BO and so can be understood in the same way.

8 Conclusion

This article compared six well-known constraint-handling techniques used with evolutionary algorithms on a range of constrained problems in both continuous and combinatorial spaces. For the combinatorial spaces, a new benchmark problem generator, the Broken Hill Problem, was introduced for instantiating assignment problems with different levels of complexity and constraint violation. For the continuous domain, the CEC 2010 suite of scalable constrained benchmark problems was used.

The 142 problem instances were characterised based on hill-climbing samples using five landscape metrics, combining information from the fitness and violation landscapes. The link between the problem characteristics and algorithm performance was then investigated.

A bi-objective approach to constraint handling (treating constraint violation as a second objective) performed better on average on all problem instances investigated. However, when considering the problems characterised by low levels of feasibility (as measured by the FsR metric), a feasibility ranking approach performed better on average.

Decision tree induction was used to further investigate the link between problem characteristics and algorithm performance. It was found that although a bi-objective approach performed well on average, it had a high probability of failing when the feasible areas were very disjoint (as measured by the RFB$\times$ metric). In these instances, other approaches such as no constraint handling, death penalty, and feasibility ranking out-performed bi-objective. In addition, a weighted penalty approach in general out-performed a bi-objective approach when problems had some measurable feasibility and the fitness and violation values were positively correlated (as measured by the FVC metric).

In the case of problems with no measurable feasibility, the no constraint handling and death penalty approaches performed very poorly, which was to be expected. When the fitness and violation were highly correlated, an ϵ-feasibility ranking approach outperformed a feasibility ranking approach, whereas a bi-objective approach was successful on problems characterised by medium to high values for the 4_IZ metric.

These results can be understood in terms of the behaviour of the constraint-handling techniques and so provide insight into the behaviour of the different constraint-handling techniques. The results could be used as guiding principles when deciding on the appropriate use of constraint-handling techniques for evolutionary algorithms. The next step in this research will be to implement these findings into an adaptive constraint-handling technique approach. This will involve capturing landscape information online, during the optimisation process, so that there is no wasted computation budget expended on a priori characterisation.

References

Bischl, B., Mersmann, O., Trautmann, H., and Preuß, M. (2012). Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the Fourteenth International Genetic and Evolutionary Computation Conference*, pp. 313–320.

Coello Coello, C. A. (1999). *A survey of constraint handling techniques used with evolutionary algorithms*. Technical Report. Laboratorio Nacional de Informtica Avanzada.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(24):311–338.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Friedrich, T. (Ed.) (2016). *GECCO '16: Proceedings of the genetic and evolutionary computation conference 2016*. New York: ACM.

Greenwood, G. W., and Hu, X. (1998a). Are landscapes for constrained optimization problems statistically isotropic? *Physica Scripta*, 57:321–323.

Greenwood, G. W., and Hu, X. (1998b). On the use of random walks to estimate correlation in fitness landscapes. *Computational Statistics and Data Analysis*, 28:131–137.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The Weka data mining software: An update. *Special Interest Group on Knowledge and Discover & Data Mining Explorations Newsletter*, 11(1):10–18.

Koziel, S., and Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44.

Liang, J., Runarsson, T., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello Coello, C., and Deb, K. (2006). *Problem definitions and evaluation criteria for the CEC 2006 competition on constrained real-parameter optimization*. Technical Report. Nanyang Technological University, Singapore.

Malan, K. M., and Engelbrecht, A. P. (2013). A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163.

Malan, K. M., and Engelbrecht, A. P. (2014). Fitness landscape analysis for metaheuristic performance prediction. In H. Richter and A. P. Engelbrecht (Eds.), *Recent advances in the theory and application of fitness landscapes*, pp. 103–132. Complexity and Computation, Vol. 6. Berlin: Springer.

Malan, K. M., Oberholzer, J. F., and Engelbrecht, A. P. (2015). Characterising constrained continuous optimisation problems. In *2015 IEEE Congress on Evolutionary Computation*, pp. 1351–1358.

Mallipeddi, R., and Suganthan, P. N. (2010). *Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization*. Technical Report. Nanyang Technological University, Singapore.

Manderick, B., de Weger, M. K., and Spiessens, P. (1991). The genetic algorithm and the structure of the fitness landscape. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 143–150.

Mezura-Montes, E., and Coello Coello, C. A. (2004). *What makes a constrained problem difficult to solve by an evolutionary algorithm*? Technical Report. Evolutionary Computation Group (EVOCINV), Electrical Engineering Department, Computer Science Department Av. Instituto Politecnico Nacional, No. 2508.

Michalewicz, Z. (1995). A survey of constraint handling techniques in evolutionary computation methods. *Evolutionary Programming*, 4:135–155.

Michalewicz, Z., and Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32.

Moser, I., Gheorghita, M., and Aleti, A. (1997). Identifying features of fitness landscapes and relating them to problem difficulty. *Evolutionary Computation*, 25(3):407–437.

Muñoz, M. A., Kirley, M., and Halgamuge, S. K. (2013). The Algorithm Selection Problem on the continuous optimization domain. In C. Moewes and A. Nürnberger (Eds.), *Computational intelligence in intelligent data analysis*, pp. 75–89. Studies in Computational Intelligence, Vol. 445. Berlin, Heidelberg: Springer.

Ochoa, G., Verel, S., Daolio, F., and Tomassini, M. (2014). Local optima networks: A new model of combinatorial fitness landscapes. In H. Richter and A. P. Engelbrecht (Eds.), *Recent advances in the theory and application of fitness landscapes*, pp. 233–262. Emergence, Complexity and Computation, Vol. 6. Berlin: Springer.

Poursoltan, S., and Neumann, F. (2014). Ruggedness quantifying for constrained continuous fitness landscapes. In R. Datta and K. Deb (Eds.), *Evolutionary constrained optimization*, pp. 29–50. Infosys Science Foundation Series. Berlin: Springer.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco: Morgan Kaufmann.

Rice, J. R. (1976). The Algorithm Selection Problem. *Advances in Computers*, 15:65–118.

Richardson, J. T., Palmer, M. R., Liepins, G. E., and Hilliard, M. (1989). Some guidelines for genetic algorithms with penalty functions. In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 191–197.

Runarsson, T. P., and Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294.

Smith-Miles, K. (2008). Towards insightful algorithm selection for optimisation using meta-learning concepts. In *Proceedings of the IEEE Joint Conference on Neural Networks*, pp. 4118–4124.

Stadler, P. F. (2002). Fitness landscapes. In *Biological Evolution and Statistical Physics*, pp. 183–204. Lecture Notes in Physics, Vol. 585.

Storn, R., and Price, K. (1995). *Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces*. Technical Report TR-95-012. International Computer Science Institute.

Storn, R., and Price, K. (1996). Minimizing the real functions of the icec96 contest by differential evolution. In *Proceedings of the International Conference on Evolutionary Computation*, pp. 842–844.

Takahama, T., and Sakai, S. (2005). *Constrained optimization by ϵ constrained particle swarm optimizer with ϵ-level control*, pp. 1019–1029. Berlin, Heidelberg: Springer.

Takahama, T., and Sakai, S. (2006). Constrained optimization by the ϵ constrained differential evolution with gradient-based mutation and feasible elites. In *2006 IEEE International Conference on Evolutionary Computation*, pp. 1–8.

Weinberger, E. (1990). Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336.