1-1-2013

Usefulness of infeasible solutions in evolutionary search: an empirical and mathematical study

Lyndon While

Philip Hingston
Edith Cowan University

10.1109/CEC.2013.6557723

While, L., & Hingston, P. (2013). Usefulness of infeasible solutions in evolutionary search: an empirical and mathematical study. Proceedings of the 2013 IEEE Congress on Evolutionary Computation. (pp. 1363-1370). Cancun, Mexico. IEEE. Availablehere

Usefulness of Infeasible Solutions in Evolutionary Search: an Empirical and Mathematical Study

Lyndon While
School of Computer Science & Software Engineering
The University of Western Australia
Email: lyndon.while@uwa.edu.au

Philip Hingston
School of Computer and Security Science
Edith Cowan University
Email: p.hingston@ecu.edu.au

***Abstract*—When evolutionary algorithms are used to solve constrained optimization problems, the question arises how best to deal with infeasible solutions in the search space. A recent theoretical analysis of two simple test problems argued that allowing infeasible solutions to persist in the population can either help or hinder the search process, depending on the structure of the fitness landscape. We report new empirical and mathematical analyses that provide a different interpretation of the previous theoretical predictions: that the important effect is on the probability of finding the global optimum, rather than on the time complexity of the algorithm. We also test a multi-objective approach to constraint-handling, and with an additional test problem we demonstrate the superiority of this multi-objective approach over the previous single-objective approaches.**

Keywords: **evolutionary algorithms, constraint-handling, multi-objective optimization.**

I. Introduction

Evolutionary algorithms (EAs) have been very successful at solving complex real-world optimization problems (see for example [1], [2], [3]). One of the difficulties of these problems is that the real world often imposes constraints on the feasibility of potential solutions, so it is important to include in an EA some means for dealing with these constraints.

A key decision in algorithm design is whether to allow populations to include infeasible solutions during the search process [4]. Whilst infeasible solutions are clearly of no interest in the final population, their presence during the search might enable the algorithm to move between disjoint feasible regions of the search space, to more-easily explore solutions at the edge of feasibility (as optimal solutions are often found there), and/or to take a shorter path to the global optimum by traversing the infeasible part of the search space. On the other hand, allowing the search to include infeasible solutions enlarges the search space and may cause the algorithm to waste resources running up blind alleys, or even to become trapped in infeasible local optima and be unable to return from there to the feasible part of the search space.

A recent theoretical analysis of two subset sum problems [5] demonstrated this conundrum. The analysis defined two single-objective algorithms, one that allows infeasible solutions and one that does not, and made several theoretical predictions of the time complexity of the two algorithms on the two problems. Although the two problems are very similar in nature, one of them could be solved in practical time only by allowing infeasible solutions in the population, while the other could be solved only by excluding infeasible solutions.

The principal contributions of this paper are threefold.

- We give new empirical and mathematical analyses of [5]'s algorithms and test problems that provide a different interpretation: that the differences can best be expressed in terms of the probability of solving the problem (i.e. avoiding getting trapped in local optima), rather than the time complexity.
- We define a new subset sum problem, closely related to the original two problems, which neither of the algorithms is able to solve reliably.
- We define a multi-objective EA *MORF* that maintains a measure of feasibility for each solution in the population as a separate objective, and we show that MORF is able to solve all three of the subset sum problems quickly 100% of the time.

Adding a constraint objective is not in itself a new idea [6], [7], [8], but we feel that the analysis reported here contributes to the understanding of this important issue.

The rest of the paper is organised as follows. Section II briefly reviews constraint-handling methods for EAs. Section III describes the two constrained problems and the two single-objective algorithms introduced in [5], and Section IV presents our empirical and mathematical analyses of these problems. Section V introduces our simple multi-objective algorithm MORF and shows that it easily solves both problems. Section VI introduces our third problem and shows that while it cannot be solved reliably by either single-objective algorithm, it is easily solved by MORF. Section VII concludes the paper and suggests possible future lines of research.

II. Related Work

The most generic statement of a constrained optimization problem is

$$\begin{array}{l}\text{minimize } f_i(x), 1 \le i \le n,\\ \text{subject to } g_i(x) = 0, 1 \le i \le k,\\ \text{and } h_i(x) \ge 0, 1 \le i \le m,\\ \text{where each } f_i : S \to R, g_i : S \to R^+, h_i : S \to R\end{array}$$

That is, from a search space S, find x that minimizes multiple real-valued functions f_i, subject to the requirement that each

non-negative real-valued function g_i maps x to 0 and each real-valued function h_i maps x to a non-negative number. The f_i are known as *objectives*, the g_i are known as *equality constraints*, and the h_i are known as *inequality constraints*.

Mezura-Montes and Coello [9] recently surveyed the state of the art in constraint-handling in nature-inspired optimization, including evolutionary algorithms and swarm intelligence algorithms. Popular kinds of constraint-handling methods discussed include

1) feasibility rules,
2) stochastic ranking,
3) ϵ-constrained method,
4) novel penalty functions,
5) novel special operators,
6) multi-objective concepts, and
7) ensemble of constraint-handling techniques.

Category 5 includes repair methods designed to ensure that all solutions are feasible. For example, Leguizamón and Coello [10] proposed a boundary search method based on a binary search between a feasible and an infeasible solution.

The other categories allow infeasible solutions. For example, in Category 1, infeasible solutions are allowed but are considered inferior. A popular example of this is Deb's *constraint domination* [7], [11], which uses the following rules when comparing potential solutions:

- all feasible solutions are preferred to all infeasible ones;
- more-feasible ones are preferred to less-feasible ones; and
- among feasible solutions, Pareto dominance determines ranking (the underlying problem can be multi-objective).

There are many other approaches and variations to constraint-handling based on feasibility rules, but Mezura-Montes and Coello [9] state that these kinds of approaches are prone to premature convergence.

In Category 2, first introduced by Runarsson and Yao [12], [13], randomness is added to the selection process to obtain a balance between objective values and feasibility.

Category 3 includes methods based on Takahama, Sakai and Iwane's ϵ-constraint concept [14], in which equally infeasible, feasible and nearly feasible solutions are compared based on their objective values, and all other cases are compared based on their feasibility.

Category 4 consists of improvements on the naïve penalty function approach, in which constraint violations are represented by penalty functions, which are added to the original objective function, creating a modified objective function. Examples include adaptive penalty functions (for example [15], [16]) and dynamic penalty functions (for example [17]).

Methods in Category 6 also allow infeasible solutions, adding additional objectives that quantify the degree of infeasibility, and then solving the resulting multi-objective problem. For example, in [18], an *objective-first ranking rule* was introduced to solve an ore-processing circuit design problem. This scheme was generalised and further studied in [8].

Many methods have been proposed and used on a wide variety of constrained optimization problems, but to date, theoretical results are sparse. Some recent work provides results for Evolution Strategies on real-valued linear problems with a single linear constraint [19], [20]. The second of these compares the strategies of repairing infeasible solutions versus resampling them.

In [21], Zhou and He obtained results concerning the effect of the choice of penalty coefficients on different problems. He and Zhou [22] presented results comparing penalising or repairing infeasible solutions on certain knapsack problems. In [5], Yu and Zhou build on theoretical results in [23] concerning *EFHTs* (expected first hitting times — the number of generations expected before the algorithm reaches the global optimum), to obtain a sufficient condition and a necessary condition concerning which of two simple single-objective EAs, one which allows infeasible solutions and one which does not, has the better EFHT.

III. Two Test Problems and Two Single-Objective Algorithms

In order to explore factors that affect the usefulness of infeasible solutions during search, we use some configurable test problems. All of the problems are based on the *subset sum problem*. Problems 1 and 2 were introduced in [5]. In Section VI we introduce a third problem, closely related to the first two.

In this version of the subset sum problem, n integer weights $W = (w_1, w_2, \ldots, w_n)$ and a constant C are given. The problem is to select a subset of the weights that maximizes the total weight, while not exceeding C. More formally,

$$\begin{array}{l}\text{maximize } \sum_{i=1}^{n} x_i \times w_i \\ \text{subject to } \sum_{i=1}^{n} x_i \times w_i \leq C, \\ \text{where each } x_i \in \{0,1\}\end{array}$$

The three problems differ with respect to the values of w_n and C. Different choices give rise to different fitness landscapes, which we can visualise by partitioning $\{0,1\}^n$ into equivalence classes based on fitness values. To this end, we introduce some notation. Assuming that the first $n-1$ weights are all equal, the fitness of a solution x depends on how many 1s occur in $x_1, x_2, \ldots, x_{n-1}$, and on whether x_n is 0 or 1. Therefore we use

$$< m > k$$

to denote the set of solutions that have m 1s in their first $n-1$ bits, and whose last bit is k. Note that this equivalence class contains $\binom{n-1}{m}$ distinct solutions.

A. Single-objective algorithms

In [5], Yu and Zhao presented two simple single-objective EAs, one which allows infeasible solutions and one which does not, and made predictions concerning the *EFHT* of the two algorithms based on a theoretical argument. In Section IV, we report empirical testing of these predictions We also present an alternative mathematical analysis.

Both algorithms from [5] can be described as single-objective, $(n+n)$-EAs. Both algorithms begin with a randomly generated population of n initial solutions. Reproduction is by single-bit mutation: from each parent, exactly one child is produced which differs from its parent in exactly one bit. In each generation, the n best solutions are selected from the n parents and their n children.

For both algorithms, the fitness of a solution is the difference between the solution's total packed weight and the capacity C, i.e.

$$F(x) = -|C - \textstyle\sum_{i=1}^{n} x_i \times w_i|$$

(Note that in [5], the fitness for one of the algorithms is presented without the absolute value, but this formulation is equivalent.)

In the first algorithm, *EA-1*, any infeasible solutions in the initial population are rejected and regenerated, and all infeasible offspring are rejected in subsequent generations. In the second algorithm, *EA-2*, infeasible solutions are allowed.

We now describe each of the problems introduced by Yu and Zhao [5].

B. Problem 1

For Problem 1, as introduced in [5],

$$\begin{array}{l} w_1, w_2, \ldots, w_{n-1} > 1, \\ w_n \geq 1 + 2\sum_{i=1}^{n-1} w_i, \text{ and} \\ C = w_n \end{array}$$

More specifically, in our experiments and diagrams to follow:

$$\begin{array}{l} w_1, w_2, \ldots, w_{n-1} = 2, \text{ and} \\ C = w_n = 1 + 2\sum_{i=1}^{n-1} w_i = 4n - 3 \end{array}$$

The fitness landscape for Problem 1 is shown in Fig. 1. The figure shows the case of an odd number of bits, rewriting n as $2m+1$, for easier comparison with later figures (although the reasoning is the same for even numbers of bits). The optimum is at $(0, 0, \ldots, 0, 1)$, i.e. the sole member of the equivalence class $< 0 > 1$. Feasible solutions are shown in a box. Arcs show possible mutation transitions. From this it can be seen that the only feasible paths to the optimum that respect fitness either start at the optimum, or start with all 0s and mutate the last bit. If the population at any point contains no solutions from $< 0 > 0$ and $< 0 > 1$, then EA-1 must fail, as it cannot produce a $< 0 > 0$ mutant (any such mutant would be less fit than all members of the parent population), and so there is no available route to the optimum.

Therefore, most feasible paths will eventually be trapped at a local optimum in $< 2m > 0$, so that this problem is very difficult for any algorithms that does not allow infeasible solutions, such as EA-1. On the other hand, if infeasible solutions are allowed, as for EA-2, then the fitness gradient will reliably drive solutions toward the optimum.

This intuitive analysis is consistent with Proposition 1 of [5], which states that Problem 1 should be easier for algorithms that allow infeasible solutions. It is also consistent with Proposition 3, which suggests that the *EFHT* for EA-2 should be at worst quadratic in n, while for EA-1 it should be at best exponential in n.

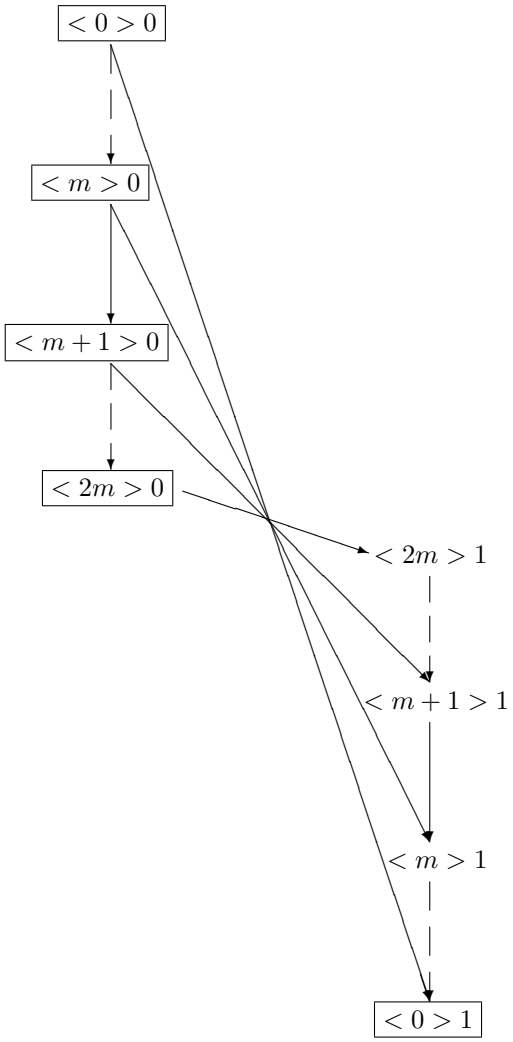

Fig. 1. Problem 1 in $n = 2m + 1$ bits, with the first $2m$ weights all equal. EA-1 recognises only $< k > 0$ and the optimum, $< 0 > 1$. Feasible equivalence classes are highlighted. Fitnesses improve down the page.

C. Problem 2

For Problem 2, [5] specifies that

$$\begin{array}{l} w_1, w_2, \ldots, w_{n-1} > 1, \\ w_n = 1 + \sum_{i=1}^{n-1} w_i, \text{ and} \\ C = \sum_{i=1}^{n-1} w_i \end{array}$$

More specifically, in our experiments:

$$\begin{array}{l} w_1, w_2, \ldots, w_{n-1} = 2, \\ w_n = 2n - 1, \text{ and} \\ C = 2n - 2 \end{array}$$

The fitness landscape for Problem 2 is shown in Fig. 2. The optimum is at $(1, 1, 1, \ldots, 0)$, i.e. in this case $< 2m > 0$. It is easy to see that the feasible paths that respect fitness all lead directly to the optimum. Infeasible solutions are not

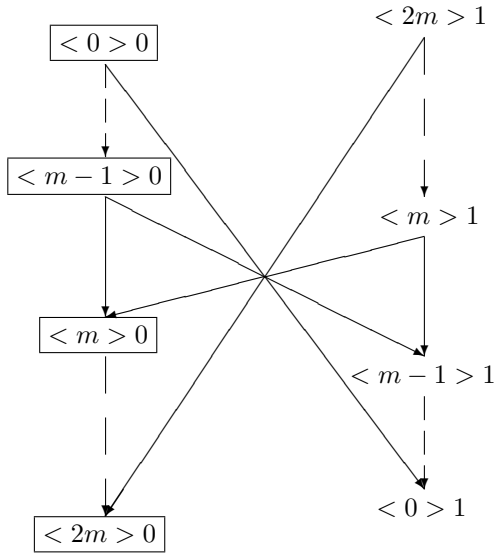

Fig. 2. Problem 2 in $n = 2m + 1$ bits, with the first $2m$ weights all equal. EA-1 recognises only $< k > 0$. The optimum is $< 2m > 0$.

helpful, but many will mutate onto the feasible path eventually. Others will become trapped in the basin of the local optimum at $< 0 > 1$, due to the combination of small fixed mutation steps (always exactly one bit), and a 100% elite selection scheme. These factors make it impossible to climb out of the basin of the attraction of the local optimum.

EA-2 can fail to solve this problem: if the population at any point consists entirely of solutions in $< k > 1, k < n/2$, then EA-2 cannot produce a feasible mutant, and will eventually become trapped at the local optimum $< 0 > 1$.

Again, this is consistent with Proposition 2 of [5], which states that an algorithm that does not allow infeasible solutions should do better on Problem 2. Proposition 4 suggests that the *EFHT* for an EA-2 should be at best exponential in n, while for EA-1 it should be at worst quadratic in n.

IV. Experiments and Results with Problems 1 & 2

In order to empirically test the performance of these algorithms, we executed each one 100 times on each problem, for $n = 10, 20, 30, \ldots, 300$. In each case, the population size was also n, as in [5]. Each run was continued until the first hit, or, if the optimum was not reached, for 1,000 generations (by which time the population has converged). We recorded, for each run, the first hitting time.

A. Problem 1

The results for Problem 1 are summarized in Fig. 3. The theoretical lower bound for EA-1 was exponential, and empirically, we found that, in fact, EA-1 *almost never solves the problem* (except for a small proportion of runs with $n = 10$). EA-2 solves the problem 100% of the time, and the *FHT* appears to be almost linear in problem size (the theoretical upper bound for EA-2, given in [5], was quadratic).

B. Problem 2

The results for Problem 2 are summarized in Fig. 4. Recall

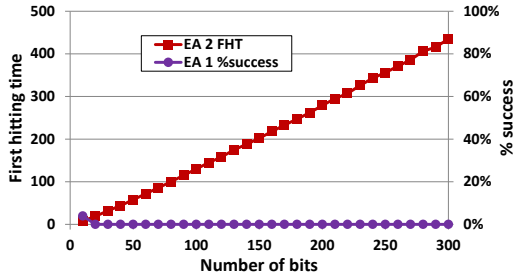

Fig. 3. Performance of the single-objective EAs on Problem 1. The x-axis is the problem size, i.e. n. The right axis is the percentage of runs on which the algorithm reached the optimum (% success). The left axis is the mean first hitting time for the algorithm, over the successful runs.

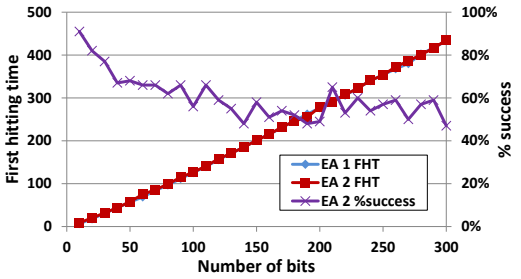

Fig. 4. Performance of the single-objective EAs on Problem 2. The performance lines for the two algorithms coincide.

that [5] predicts that EA-1 should beat EA-2 on this problem, and further that *EFHT* should be at best exponential for EA-2. Examination of our empirical results shows that, while the theoretical results hold, a more refined description of the behaviour of the algorithms is possible.

EA-1 does indeed do better than EA-2, but in an interesting way. For those runs where EA-2 succeeds, the *FHT* appears to be approximately linear (note: not quadratic) in problem size, as it is for EA-1 (in fact the curves for the two algorithms are indistinguishable in the figure). The real difference between the two algorithms is their success rate. For smaller values of n, say up to about 30, EA-2 succeeds more than 75% of the time, a percentage that drops to around 50-60% for $n > 100$. While the *EFHT* is indeed at least exponential, in practical terms, it either succeeds quickly (about half the time), or fails entirely.

C. Expected successful first hitting time for Problem 2

These empirical results suggest that rather than considering only expected first hitting time over all runs of an algorithm, a better understanding is obtained by factoring *EFHT* into

$$ESFHT(n) = \frac{\sum_{j=0}^{n-1} \binom{n-1}{j}(E(n-1-j,0)S(n-1-j,0) + E(j,1)S(j,1))}{\sum_{j=0}^{n-1} \binom{n-1}{j}(S(n-1-j,0) + S(j,1))} \quad (1)$$

$$E(j,0) = 0, \text{ if } j = 0 \text{ (as this is already at the optimum)} \quad (2)$$

$$= 1 + \frac{jE(j-1,0) + (n-j)E(j,0)}{n}, \text{ if } 0 < j < \frac{n}{2} \quad (3)$$

$$= 1 + \frac{jE(j-1,0) + (n-1-j)E(j,0)}{n-1}, \text{ if } j \geq \frac{n}{2} \quad (4)$$

$$E(j,1) = \text{irrelevant, if } j < \frac{n}{2} \text{ (because these cases all fail)} \quad (5)$$

$$= 1 + \frac{E(n-1-j,0) + (n-j-1)E(j,1)}{n-j}, \text{ if } j = \frac{n}{2} \quad (6)$$

$$= 1 + \frac{E(n-1-j,0) + jE(j-1,1) + (n-j-1)E(j,1)}{n}, \text{ if } j > \frac{n}{2} \quad (7)$$

$$S(j,x) = 1 - x, \text{ if } j < \frac{n}{2} \quad (8)$$

$$= \frac{jS(j-1,x) + x}{j+1}, \text{ if } j \geq \frac{n}{2} \quad (9)$$

Fig. 5. Recurrence relations for the ESFHT of EA-2 on Problem 2 in n bits (n even), with a population of 1. $E(j,x)$ defines the ESFHT from the given state, and $S(j,x)$ defines the probability of success from the given state.

success rate, and expected first hitting time over successful runs. This is what we have plotted in Figs. 3 and 4.

In the remainder of the paper, we will refer to the latter as *ESFHT (expected successful first hitting time)*. We can illustrate the utility of this idea by deriving some theoretical performance calculations for EA-2 on Problem 2, for the case of a population size of 1 - i.e. we consider the case of a (1+1)-EA, a hill-climber.

We first introduce some notation. Let n be the total number of bits. Assume that n is even: this means that

- the basin of attraction for the global optimum is $< k > 0, k \geq n/2$, and
- the basin of attraction for the local optimum is $< k > 1, k < n/2$.

Allowing for odd n just complicates the second of these slightly.

We define $S(j,x)$ to be the probability that EA-2 succeeds, starting from a solution that has x as its last bit, and has j bits different from the corresponding local or global optimum. Recall that the global optimum is at $< n-1 > 0$ and the local is at $< 0 > 1$, so if x is 0, then j is the number of 0s in the solution, and if x if 1, then j is the number of 1s. We define $E(j,x)$ to be the expected first hitting time for a population that starts with such a solution, *given that the run is successful*. There are six distinct cases to consider for $E(j,x)$ and two for $S(j,x)$, all shown in Fig. 5.

Equation 1 calculates *ESFHT* as an average of the expected first hitting times for successful runs from each possible starting state, where the average is weighted by success probability, and by the relative frequency of each starting state.

Equation 3 is obtained by considering the possible mutations from $< n-j-1 > 0$. j of these mutations flip one of the initial 0s to a 1, producing a solution in $< n-j > 0$, which then has an expected successful first hitting time of $E(j-1,0)$. All other possible mutations are rejected because the resulting mutant would be less fit than its parent: there are $n-j$ of those. Equation 4 is slightly different because one of the possible mutations leads to failure, and we consider only successful runs in this calculation. Equations 6 and 7 are obtained in the same way as Equations 3 and 4. In all of these equations, the denominator is the number of mutations that do not immediately lead to guaranteed failure.

Equation 8 reflects the fact that success is guaranteed when x is 0 and the other bits are mostly 1s, while failure is guaranteed if x is 1 and the other bits are mostly 0s. Equation 9 is then obtained in a similar way to the other equations.

These recurrence relations can be solved by straightforward algebraic manipulation to obtain the solutions shown in Fig. 6. From these it is trivial to calculate *ESFHT* programmatically. The result is illustrated in Figs. 7 and 8. While we have not been able to derive an exact closed-form solution for these equations, some partial results follow. H_j is the j^{th} harmonic number.

$$E(j,0) = nH_j, \text{ for } j < n/2$$
$$E(j,0) = (n-1)H_j + H_{n/2-1}, \text{ for } j \geq n/2$$
$$E(j,1) < nH_{n/2-1} + n, \text{ for } j \geq n/2$$

(Recall that $E(j,1), j < n/2$ is irrelevant: these are all

$$ESFHT(n) = \frac{\sum_{j=0}^{n-1} \binom{n-1}{j}(E(n-1-j,0)S(n-1-j,0) + E(j,1)S(j,1))}{\sum_{j=0}^{n-1} \binom{n-1}{j}(S(n-1-j,0) + S(j,1))} \quad (10)$$

$$E(j,0) = 0, \text{ if } j = 0 \quad (11)$$

$$= E(j-1,0) + \frac{n}{j}, \text{ if } 0 < j < \frac{n}{2} \quad (12)$$

$$= E(j-1,0) + \frac{n-1}{j}, \text{ if } j \geq \frac{n}{2} \quad (13)$$

$$E(j,1) = E(n-1-j,0) + n - j, \text{ if } j = \frac{n}{2} \quad (14)$$

$$= \frac{n + E(n-1-j,0) + jE(j-1,1)}{j+1}, \text{ if } j > \frac{n}{2} \quad (15)$$

$$S(j,x) = 1 - x, \text{ if } j < \frac{n}{2} \quad (16)$$

$$S(j,0) = \frac{n}{2(j+1)}, \text{ if } j \geq \frac{n}{2} \quad (17)$$

$$S(j,1) = 1 - \frac{n}{2(j+1)}, \text{ if } j \geq \frac{n}{2} \quad (18)$$

Fig. 6. Solutions to the recurrence relations in Fig. 5.

j	$S(j,0)$	$E(j,0)$	$S(j,1)$	$E(j,1)$
9	0.500	27.544	0.500	23.833
8	0.556	26.544	0.444	25.370
7	0.625	25.419	0.375	26.042
6	0.714	24.133	0.286	26.190
5	0.833	22.633	0.167	25.833
4	1	20.833	0	-
3	1	18.333	0	-
2	1	15.000	0	-
1	1	10.000	0	-
0	1	0.000	0	-

Fig. 7. $S(j,x)$ and $E(j,x)$ for EA-2 on Problem 2, with $n = 10$.

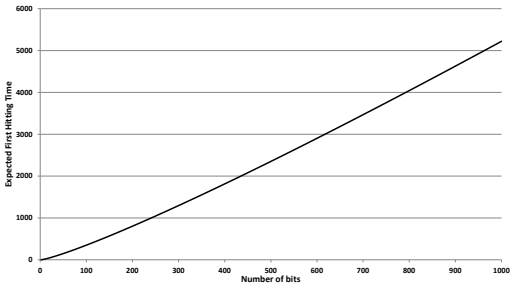

Fig. 8. The ESFHT for EA-2 on Problem 2, with a population of 1.

guaranteed failures.) From these formulae, and recalling that $H_n \approx log(n) + \gamma$, where γ is the Euler-Mascheroni constant, we can see that ESFHT is $O(nlog(n))$.

Also, the overall success rate can be derived from Equations 16–18. The exact formula is $\frac{\lceil \frac{n}{2} \rceil}{n}$. This formula tells us that the success rate is proportional to the relative size of the basin of attraction of the global optimum, which is an intuitively satisfying result.

V. A MULTI-OBJECTIVE ALGORITHM

To compare with these single-objective algorithms, we also test a simplified version of our algorithm from [8]. We explicitly represent both the original weight objective and an 'infeasibility' or 'error' objective:

- original objective: minimize the unused capacity, i.e. minimize $max(0, C - \sum_{i=1}^{n} x_i \times w_i)$, and
- infeasibility objective: minimize the excess load, i.e. minimize $max(0, \sum_{i=1}^{n} x_i \times w_i - C)$.

With these two objectives, the following ranking rule is used:

- solutions are first ranked using Fonseca and Fleming's Pareto ranking [24] (including the additional error objective);
- for solutions with the same rank, the most feasible one is preferred.

Note that in this ranking scheme:

- no infeasible solution can dominate a feasible one, because a feasible solution always has an optimal value in the infeasibility objective;

- the global optimum from the original problem (if there is one) dominates every other solution, because it has the best value in the original objective, and an optimal value in the infeasibility objective.

The initial population is created randomly as for the single-objective case. The ranking scheme above is then used to select the best n solutions from amongst n parents and their n children in each subsequent generation. We refer to this algorithm as *MORF* (multi-objective - ranking first).

A. Performance of MORF on Problems 1 and 2

The performance of MORF on Problem 1 is almost identical to that of EA-2. On Problem 2, the performance of MORF is almost identical to that of EA-1. In both cases, MORF always succeeds, in time that appears to be almost linear with problem size. Fig. 9 illustrates. Thus the multi-objective algorithm performs as well as the best single-objective algorithm on both problems, which suggests that the treatment of infeasible solutions is not necessarily the key issue.

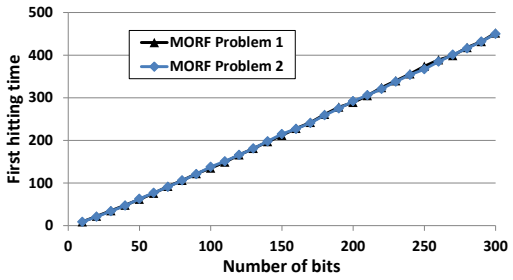

Fig. 9. Performance of the multi-objective EA MORF on Problems 1 and 2. The lines for the two problems coincide.

Fig. 10 shows the 2-D fitness landscape for Problem 2 using MORF. Note that there are now no local optima: compare this with Fig. 2.

VI. Problem 3

We have seen that EA-1 can solve Problem 2 easily, but cannot solve Problem 1, while EA-2 solves Problem 1 easily, while solving Problem 2 only around 50% of the time. This poses the question: are there problems like these that are difficult for both EA-1 and EA-2?

The answer is *yes*. We introduce a new problem, Problem 3, which is the same as Problem 1 except that the constant factor 2 is missing, i.e.

$$w_1, w_2, \ldots, w_{n-1} = 2, \text{ and}$$
$$C = w_n = 1 + \sum_{i=1}^{n-1} w_i = 2n - 1$$

The fitness landscape for Problem 3 is shown in Fig. 11. As for Problem 1, the optimum is at $(0, 0, \ldots, 0, 1)$, i.e. it is in $< 0 > 1$. There is a large basin of attraction around

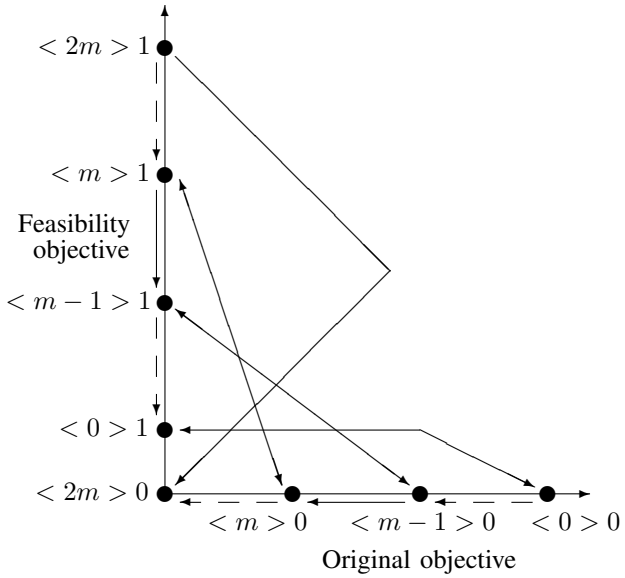

Fig. 10. Objective space for Problem 2 as a multi-objective problem. All feasible solutions are on the x-axis (i.e. the 'error' objective value is 0). An arrow from x to y means that y can be preferred to x under some circumstances.

a feasible local optimum in the bottom left corner of the figure. There is also opportunity for infeasible solutions to fall towards that local optimum. Therefore, a single objective algorithm, whether it allows infeasible solutions or not, might be expected to find this problem difficult. As we have proposed this problem in the present paper, there is no analysis of it in [5].

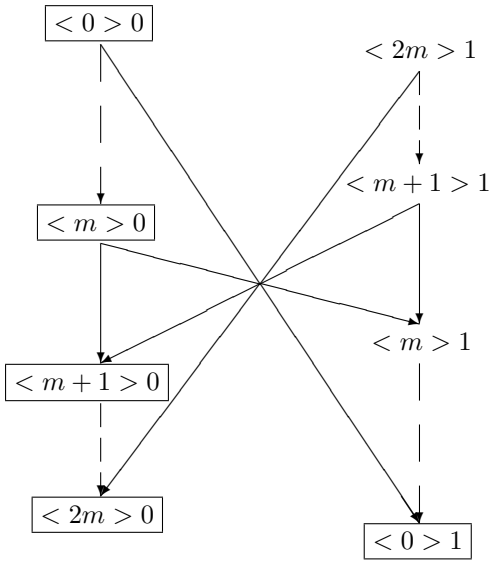

Fig. 11. Problem 3 in $n = 2m + 1$ bits, with the first $2m$ bits all equal. EA-1 recognises only $< k > 0$ and the optimum, $< 0 > 1$.

A. Performance Results for Problem 3

Our empirical results show that Problem 3 is virtually unsolvable for EA-1, presents similar difficulties to those of Problem 2 for EA-2, but is easily solved, in apparently almost linear time, by MORF. Fig. 12 illustrates.

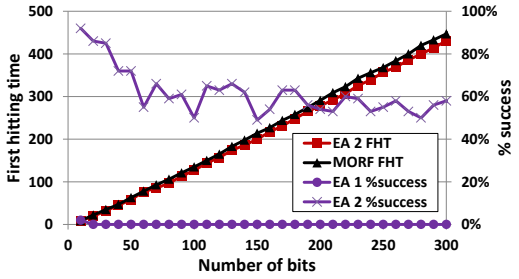

Fig. 12. Performance of all three EAs on Problem 3. The performance lines for EA-2 and MORF almost coincide. MORF has a 100% success rate.

These results reinforce the finding that whether or not infeasible solutions are allowed to persist in the population is not the key issue in determining the success of an EA on these problems. Rather, it seems that the size and location of local optima is the determining factor. EA-1 cannot handle Problem 1 because of a large local optimum in feasible space; EA-2 has trouble with Problem 2 because of the local optimum in feasible+infeasible space; and neither does well on Problem 3, which has a local optimum that appears in the search space as a whole as well as in feasible space. These local optima disappear in the multi-objective landscape.

VII. Conclusions

We have given new empirical and mathematical analyses of a recent study of the question of the usefulness of infeasible solutions in evolutionary search. Yu and Zhou [5] presented two similar problems, one which can be solved only by maintaining infeasible solutions in the population, and another which can be solved only by excluding them. We have given empirical and mathematical evidence to support a re-interpretation of Yu and Zhou's results, we have described a third similar problem which cannot be solved reliably by either single-objective approach, and we have defined a simple multi-objective algorithm that solves all three problems 100% of the time by maintaining "degree of infeasibility" as a separate objective.

We plan to extend this work by generalising our analysis into more-complex problem sets.

References

[1] D. Dasgupta and Z. Michalewicz, *Evolutionary Algorithms in Engineering Applications*. Springer, 2010.

[2] R. Chiong, T. Weise, and Z. Michalewicz (eds.), *Variants of Evolutionary Algorithms for Real-world Applications*. Springer, 2012.

[3] P. Hingston, L. Barone, and Z. Michalewicz (eds.), *Design by Evolution: Advances in Evolutionary Design*. Springer, 2010.

[4] Z. Michalewicz, "Do not kill unfeasible individuals," in *4th Intelligent Information Systems Workshop*, 1995, pp. 110–123.

[5] Y. Yu and Z. Zhou, "On the usefulness of infeasible solutions in evolutionary search: A theoretical study," in *World Congress on Computational Intelligence*. IEEE, 2008, pp. 835–840.

[6] P. D. Surry, N. J. Radcliffe, and I. D. Boyd, "A multi-objective approach to constrained optimisation of gas supply networks: the COMOGA method," *Lecture Notes in Computer Science*, vol. 993, pp. 166–180, 1995.

[7] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[8] P. Hingston, L. Barone, S. Huband, and L. While, "Multi-level ranking for constrained multi-objective evolutionary optimisation," *Lecture Notes in Computer Science*, vol. 4193, pp. 563–572, 2006.

[9] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, pp. 173–194, 2011.

[10] G. Leguizamón and C. Coello, "Boundary search for constrained numerical optimization problems with an algorithm inspired by the ant colony metaphor," *IEEE Trans. on Evolutionary Computation*, vol. 13, no. 2, pp. 350–368, 2009.

[11] K. Deb, A. Pratap, and T. Meyarivan, "Constrained test problems for multi-objective evolutionary optimization," in *1st International Conference on Evolutionary Multi-Criterion Optimization*, 2001, pp. 284–298.

[12] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.

[13] ——, "Search biases in constrained evolutionary optimization," *IEEE Trans. on Systems, Man and Cybernetics, Part C*, vol. 35, no. 2, pp. 233–243, 2005.

[14] T. Takahama, S. Sakai, and N. Iwane, "Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm," *AI 2005: Advances in Artificial Intelligence*, pp. 389–400, 2005.

[15] R. Farmani and J. Wright, "Self-adaptive fitness formulation for constrained optimization," *IEEE Trans. on Evolutionary Computation*, vol. 7, no. 5, pp. 445–455, 2003.

[16] B. Tessema and G. Yen, "An adaptive penalty formulation for constrained evolutionary optimization," *IEEE Trans. on Systems, Man and Cybernetics, Part A*, vol. 39, no. 3, pp. 565–578, 2009.

[17] S. Puzzi and A. Carpinteri, "A double-multiplicative dynamic penalty approach for constrained evolutionary optimization," *Structural and Multidisciplinary Optimization*, vol. 35, no. 5, pp. 431–445, 2008.

[18] S. Huband, L. Barone, P. Hingston, L. While, D. Tuppurainen, and R. Bearman, "Designing communition circuits with a multi-objective evolutionary algorithm," in *Congress on Evolutionary Computation*, 2005, pp. 1815–1822.

[19] D. V. Arnold and D. Brauer, "On the behaviour of the (1+1)-ES for a simple constrained problem," in *10th international conference on Parallel Problem Solving from Nature*. Springer-Verlag, 2008, pp. 1–10.

[20] D. V. Arnold, "Analysis of a repair mechanism for the (1,1)-ES applied to a simple constrained problem," in *13th Genetic and Evolutionary Computation Conference*, 2011, pp. 853–860.

[21] Y. Zhou and J. He, "A runtime analysis of evolutionary algorithms for constrained optimization problems," *IEEE Trans. on Evolutionary Computation*, vol. 11, no. 5, pp. 608–619, 2007.

[22] J. He and Y. Zhou, "A comparison of GAs using penalizing infeasible solutions and repairing infeasible solutions on restrictive capacity knapsack problem," in *9th Genetic and Evolutionary Computation Conference*, 2007, pp. 1518–1518.

[23] Y. Yu and Z. Zhou, "A new approach to estimating the expected first hitting time of evolutionary algorithms," *Artificial Intelligence*, vol. 172, no. 15, pp. 1809–1832, 2008.

[24] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *5th International Conference on Genetic Algorithms*, 1993, pp. 416–423.