10-2008

On a Feasible–Infeasible Two-Population (FI-2Pop) Genetic Algorithm for Constrained Optimization: Distance Tracing and no Free Lunch

Steven. O. Kimbrough
University of Pennsylvania

Ming Lu
University of Pennsylvania

Gary J. Koehler

David H. Wood

On a Feasible–Infeasible Two-Population (FI-2Pop) Genetic Algorithm for Constrained Optimization: Distance Tracing and no Free Lunch

Abstract
We explore data-driven methods for gaining insight into the dynamics of a two-population genetic algorithm (GA), which has been effective in tests on constrained optimization problems. We track and compare one population of feasible solutions and another population of infeasible solutions. Feasible solutions are selected and bred to improve their objective function values. Infeasible solutions are selected and bred to reduce their constraint violations. Interbreeding between populations is completely indirect, that is, only through their offspring that happen to migrate to the other population. We introduce an empirical measure of distance, and apply it between individuals and between population centroids to monitor the progress of evolution. We find that the centroids of the two populations approach each other and stabilize. This is a valuable characterization of convergence. We find the infeasible population influences, and sometimes dominates, the genetic material of the optimum solution. Since the infeasible population is not evaluated by the objective function, it is free to explore boundary regions, where the optimum is likely to be found. Roughly speaking, the No Free Lunch theorems for optimization show that all blackbox algorithms (such as Genetic Algorithms) have the same average performance over the set of all problems. As such, our algorithm would, on average, be no better than random search or any other blackbox search method. However, we provide two general theorems that give conditions that render null the No Free Lunch results for the constrained optimization problem class we study. The approach taken here thereby escapes the No Free Lunch implications, per se.

Keywords
constrained optimization, heuristics, blackbox search, no free lunch

Disciplines
Theory and Algorithms

On a Feasible-Infeasible Two-Population (FI-2Pop) Genetic Algorithm for Constrained Optimization: Distance Tracing and No Free Lunch

Steven Orla Kimbrough
University of Pennsylvania
kimbrough@wharton.upenn.edu

Gary J. Koehler
University of Florida
koehler@ufl.edu

Ming Lu
University of Pennsylvania
milu@wharton.upenn.edu

David Harlan Wood
University of Delaware
wood@cis.udel.edu

April 10, 2007

Abstract

We explore data-driven methods for gaining insight into the dynamics of a two population genetic algorithm (GA), which has been effective in tests on constrained optimization problems. We track and compare one population of feasible solutions and another population of infeasible solutions. Feasible solutions are selected and bred to improve their objective function values. Infeasible solutions are selected and bred to reduce their constraint violations. Interbreeding between populations is completely indirect, that is, only through their offspring that happen to migrate to the other population. We introduce an empirical measure of distance, and apply it between individuals and between population centroids to monitor the progress of evolution. We find that the centroids of the two populations approach each other and stabilize. This is a valuable characterization of convergence. We find the infeasible population influences, and sometimes dominates, the genetic material of the optimum solution. Since the infeasible population is not evaluated by the objective function, it is free to explore boundary regions, where the optimum is likely to be found. Roughly speaking, the No Free Lunch theorems for optimization show that all blackbox algorithms (such as Genetic Algorithms) have the same average performance over the set of all problems. As such, our algorithm would, on average, be no better than random search or any other blackbox search method. However, we provide two general theorems that give conditions that render null the No Free Lunch results for the constrained optimization problem class we study. The approach taken here thereby escapes the No Free Lunch implications, per se.

Keywords: Constrained optimization, Heuristics, Blackbox search, No free lunch

File: JointPaper-FI-2PopGA-20070410.doc.

1 Introduction

This paper introduces and discusses empirical techniques for gaining insights into the dynamics of genetic algorithms (GAs) for constrained optimization. The techniques apply to GAs and more generally to evolutionary algorithms (EAs, roughly, any metaheuristic that is population-based). Given the present state of analytic results for understanding GAs (and EAs), and the No Free Lunch theorems, limiting all blackbox algorithms (Wolpert and Macready, 1995, 1997), empirical investigations of these metaheuristics are desirable, even unavoidable, if we are to understand them better. Additional research is desirable both from an analytic and from an empirical perspective. Indeed, the two approaches should be seen as complementary. In this spirit, we nibble from both sides at the problem of understanding GAs. In what follows, we provide general results that show the No Free Lunch theorems do not apply in our setting. Having accomplished this, we discuss data-driven methods for obtaining insight into the evolutionary dynamics of the FI-2Pop GA (a variety of GA explained below) that has proven effective for interesting constrained optimization problems.

Among other methods, we introduce and report here an empirical method involving distances between population centroids evolving during a run of the GA. Our discussion hardly exhausts the scope of our distance-based method, let alone empirical methods generally. Even so, insights or at least intriguing hypotheses are produced. The contribution here is by its nature something of a case study, yet it adds to the long-term accumulation of evidence on how GAs work.

2 Context and Background

Genetic algorithms meliorize. Heuristics, including GAs and the more general category of evolutionary algorithms (EAs), seek better and better decisions for a given objective. In doing so, they offer neither a practical guarantee that an optimal decision will be found, nor an ability to recognize one if it were found. These properties are disadvantages, compared to many optimization procedures established in the Operations Research (OR) literature. On the other hand traditional OR solvers also have these disadvantages, when problems are nonlinear and have mixtures of boolean, integer and real number decision variables. In any event, there is the question of what value GAs (and EAs) bring to the table as a practical matter in solving difficult optimization problems. But it is often the case that using heuristics, including GAs, is the only workable approach. More generally, there is a case to be made for the deployment and use of a GA if a GA can get a better decision, even if it takes longer, or if a GA can get an equally good decision faster, or if a GA can provide valuable decision making information not otherwise readily available.

In all these ways, GAs are promising, are contenders. *Constrained* optimization, however, presents a fundamental impediment to their deployment and use. The difficulty is that the genetic operations—including mutation and crossover—are not guaranteed to preserve feasibility. A single mutation in a highly-fit individual can, and often in practice will, result in an infeasible individual. Similarly, offspring produced by crossover from

two highly-fit parents may be infeasible. We give, in §3, a brief introduction to constraint handling in GAs. Further elaboration would divert us from the main purposes of this paper (but see references cited). In a nutshell, although the matter of constraint handling is a severe one for GAs and although much attention has been given to this issue, no generally accepted solution has emerged.

Recent work, however, has shown promise and prowess for a feasible-infeasible two-population (FI-2Pop) genetic algorithm for constrained optimization (Kimbrough, et al. 2002, 2003, 2004a, 2004b). The FI-2Pop GA has beaten standard methods for handling constraints in GAs (Kimbrough, et al. 2002); has regularly produced better decisions for comparable computational effort than GENOCOP, a high-quality GA solver engine for constrained optimization problems (Kimbrough, et al. 2003, 2004b); has produced excellent decisions for problems that cannot be handled by GENOCOP (Kimbrough, et al. 2004a, 2004b); and has produced better decisions than GAMS solvers, including a case in which GAMS could only solve a relaxed version (Kimbrough, et al. 2004a,).

Promising as these results appear, the No Free Lunch (NFL) theorems for search (Wolpert and Macready, 1995, 1997) prove that all blackbox algorithms have the same average performance over the set of all problems. Like most GA (or EA) algorithms FI-2Pop GA appears to be a blackbox algorithm. If so, FI-2Pop GA would, on average, be no better than random search or any other blackbox search method. We provide two general theorems (Theorems 4 and 5) that give conditions that render null the NFL results under certain assumptions for the constrained problems we study. Using these we show the approach taken by FI-2Pop GA escapes the NFL implications, per se. So, in our setting, it is possible for one algorithm to dominate another.

Given these (and other) results, it is fair to conclude that the FI-2Pop GA has something going for it on constrained optimization problems and merits further investigation. Piling up results from new, wider, more difficult tests is, of course, necessary and always welcome. Complementing this is the need to understand, to have insight into, when and why the FI-2Pop GA should work well and indeed to better understand how it works. This latter goal, of understanding the algorithm better, is the focus of this paper.

3 Constraint Handling in Genetic Algorithms

The difficulty that arises in using GAs for *constrained* optimization problems is that feasible parents may give rise to infeasible offspring. Considerable attention has been paid to treating constraints in constrained optimization problems (see Bäck 1996, Bäck et al. 2000, Coello 1999, Coello 2002 Eiben 2001, Michalewicz et al. 1996, Michalewicz 1995, 1996, 1997, 2000, Michalewicz and Fogel 2002 (chapter 9), Sarker 2002 and Smith and Coit 1997 for excellent reviews and treatments; there is even a dedicated Web site Coello, 2003). But no consensus method has emerged.

For example, one useful approach is repair, that is, some process for making infeasible individuals become feasible. A second approach is to modify the operators of the GA (mutation, crossover, etc.) so that infeasible offspring are less often produced. A third approach is to use decoders, which translate any genotype into a feasible phenotype.

There are other methods as well. (See Michalewicz (1996) and Eiben and Smith (2003) for detailed discussions of established constraint handling methods.)

Another natural approach does not require problem-dependent knowledge and is the most commonly used approach. This is the approach of penalizing each infeasible individual in proportion to the size of its constraint violations. A feasible individual is not penalized; an infeasible individual is penalized as a function of the magnitude of the violation(s) of the constraint(s). Putting this more formally, here is the general form of a maximization constrained optimization problem.

$$\max_{x_i \in S_i} Z(x), \text{subject to } F(x) \le b, G(x) = c \tag{1}$$

Here, $Z(x)$ is the objective function value produced by the candidate solution *x* (a vector; *b* and *c* are also vectors; our discussion is limited to the single-objective case). F and G each yield zero or more constraint inequalities or equalities. Taken as functions, in the general case Z, F, G can be any functions at all (on *x*) and in particular need not be linear. S_i is the set of permitted values for the x_i s (the components of the vector x), which are called the *decision variables* for the problem. S_i may consist of boolean, real value, or integer variables. Problems of the form of expression (1) are not directly translatable into the linear encodings normally used for EA (including GA) solutions.

The purpose of a penalty function formulation is to produce a representation of the problem that can be directly and naturally encoded as an EA/GA. To illustrate a penalty function representation, let *x* be a solution to a maximization constrained optimization problem. Its value, $W(x)$, in the presence of penalties for constraint violation is $Z(x) - P(x)$ and the COP (constrained optimization problem) is replaced by:

$$\max_x W(x) = Z(x) - P(x) \tag{2}$$

where $P(x)$ is the total penalty associated with constraint violations (if any) by *x*. This new value is used in calculating fitness by the GA. Problems representable as in expression (2) are directly and naturally encoded as EAs. If an EA finds a solution *x* that is feasible ($P(x) = 0$) and has a high value for $W(x)$, then we may congratulate ourselves on the successful application of this EA metaheuristic.

Typically, and by design, the penalty imposed on an infeasible solution will severely reduce the net fitness of the solution in question, leading to quick elimination of the solution from the EA population. This may be undesirable and it may be responsible for the often weak performance of EAs for constrained optimization problems. (The extreme form of this policy is to remove infeasible solutions whenever they appear. Many authors have discouraged this policy, e.g., Michalewicz (1995a).)

4 The Feasible-Infeasible Two-Population GA and Intuitions on Its Workings

The key to our approach is the following. Conventionally, we select feasible individuals with the goal of increasing payoff, while disregarding potential constraint violations. Unconventionally, we select infeasible individuals with the goal of repairing them, while disregarding potential payoffs. (Details of our algorithm are given in Appendix A.)

The mechanics of the Feasible-Infeasible Two-Population (FI-2Pop) genetic algorithm for constrained optimization are as follows. Two populations are provided at initialization and maintained throughout the run of the GA. The feasible population contains only feasible individuals (aka: solutions, decisions) and the infeasible population contains only infeasible individuals. Every individual is tested for feasibility and placed in the appropriate population. Individuals in the feasible population are selected by fitness with respect only to their objective function values. Individuals in the infeasible population are selected by fitness with respect only to a function of their constraint violations. Following selection, individuals are created using genetic operators (we use mutation and crossover). They are tested for feasiblity and placed in the appropriate population. That is to say, within each population we operate a rather standard GA.

Often in our experiments with various models, simply finding any feasible solution whatsoever is a difficult task. It is one of the virtues of the FI-2Pop GA that it can be initialized with an empty feasible population and run with the prospect of eventually finding feasible solutions. The paper by Kimbrough, Lu and Safavi (2004) presents a detailed case study of a good success in which no feasible solutions were found until after more than 2500 generations.

We note that there are other varieties of two-population GAs. The FI-2Pop GA is distinct, so far as we know, in maintaining a crisp distinction between evaluation of a feasible solution by the objective function and evaluation of an infeasible solution by constraint violations. SGGA is a GA that maintains two violation-penalized populations, using different penalty functions, and crossbreeds between them (Le Riche, et al., 1995a and 1995b). GENOCOP III is based on repair and maintains two populations. The FI-2Pop GA resembles GENOCOP III in maintaining two populations, one feasible and one infeasible (or "not fully feasible" in the case of GENOCOP). GENOCOP repairs infeasible solutions and the FI-2Pop GA evolves infeasible solutions towards feasibility. See Michalewicz (1996) for a review of both SGGA and GENOCOP. Chu and Beasley (1997, 1998) have explored a single-population GA in which each solution receives two fitness scores, one on the objective function (or 'fitness'), one on infeasibility (or 'unfitness'). Parents are selected for breeding based only their 'fitness' values; individuals are removed from the population in order of their 'unfitness' values. Yuchi and Kim (2004) report success with a two-population scheme in which a portion of the infeasible solutions are probabilistically accepted for breeding each generation, based on their objective function values. BenHamida and Schoenauer (2000, 2002) have originated and explored an approach that, of all of these, is perhaps closest to the FI-2Pop GA. In this approach (called ASCHEA), both feasible and infeasible individuals may co-exist in

a single population, and the selection operator is designed to encourage both kinds of solutions.

The FI-2Pop GA was conceived as a principled response to the known limitations of using penalty functions on constraints (described above), and to the tempting view that "The ideal way of handling constraints is to have the search space limited to the feasible space" (Michalewicz, 1996). We have been motivated by the sentiment of ``Do not kill unfeasible individuals'' (Michalewicz, 1995a), since they may well contain information valuable enough for them to be preserved. Penalty functions (using constraint violations) are problematic for a number of reasons. First, and most obviously, there is no known way to compute the size of the penalties from basic principles. Completely eliminating infeasible decisions is generally recognized to produce poor performance. Very high penalties are tantamount to eliminating infeasible decisions. Very low penalties lead to misdirecting the GA's search by seeding the population with infeasible decisions that score well on the objective function. A middling degree of penalization would seem to be called for, but on what scale is it to be measured? No principled answer has been forthcoming and penalty-function approaches are generally thought to perform less well than systems such as GENOCOP, which use forms of repair and other devices.

Second, and more fundamentally, penalty function approaches conflate objective function evaluation and constraint violation evaluation. If these could be separated by applying one or the other, then arguably or at least intuitively, it would be possible to do better at each. The FI-2Pop GA can be seen as achieving this, at least approximately. Feasible decisions are evaluated only with regard to the objective function, infeasible decisions only with regard to the constraints. In a penalized approach, an infeasible decision is subjected to evaluation with regard to both its objective value and its constraints, thereby reducing the effect of selection for feasibility.

Third, optimal solutions to constrained problems are either in the interior of the feasible region or they are on or near the boundary of the feasible region. Selection on the infeasible population in the FI-2Pop GA will drive the population to, and eventually over, the boundary. If the optimal solution is on or near the boundary, this represents enhanced exploration. The infeasible population will tend to probe the neighborhood of the boundary of the feasible region (or boundaries; nothing in this argument requires a connected boundary). Feasible children of infeasible parents will tend to resemble them (as all children do), and hence tend to be near by. Because the infeasible parents were selected without regard to their objective functions, mutation and crossover will tend to create feasible solutions that presumably are somewhat different than those already in the main population. These solutions either succeed or fail; in either case they contribute to an exploration of the feasible region. Thus, if the optimal solution is on or near the boundary, one would expect that the infeasible population would contribute to finding it.

On the other hand, if the optimal solution is in the interior, away from the boundary, the infeasible population may not be useful; the result is merely a slower, but otherwise unimpeded GA on the feasible population. The quality of the decisions found may not be affected. Further, it might seem that maintaining a progressively diminishing population

of infeasible solutions is wasteful if the optimal solutions are far from the boundary. However, for problems where it happens to be difficult to obtain feasible solutions the breeding of infeasible solutions near the boundary may be a useful, even primary, source of feasible solutions.

Of course, in the special case of a problem having its optimal solutions in the interior, and also having near-optimal solutions near the boundary, one might expect the FI-2Pop GA to be deceived and to settle on these near-optimal, near-boundary solutions. Perhaps this is a case in which the death penalty approach would perform better.

We are pleased to note that we are hardly the first to have the intuition that probing the boundary of the feasible region of a constraint set is likely to yield useful information. In the evolutionary computation community, important early work by Schoenauer and Michalewicz (1996, 1998) recognized the value of boundary exploration. Their work presented techniques by which GAs may explore the boundary of the feasible region. These techniques achieved excellent results in certain cases. As noted by these authors themselves and in a review by Coello, these particular approaches are unlikely to be the final word on the subject.

> The main drawback … is that the operators designed are either highly dependent on the chosen parameterization…, or more complex calculations are required to perform crossover and mutation. Also, many problems have disjoint feasible regions and the use of operators of this sort would not be of much help in those cases since they would explore only one of those feasible regions. (Coello 2002)

In any event, it is well worth investigating multiple approaches by which GAs may probe boundaries between feasible and infeasible regions for a COP.

In sum, by separating objective function evaluation and feasibility evaluation the FI-2Pop GA circumvents the problem of misdirecting the genetic search by infelicitous compromise between criteria for the objective function and for the constraints. Further, the infeasible population serves as a reservoir of genetic information that probes the boundary region of the feasible region, and contributes variation to the feasible population. And variation is a prerequisite for selection.

We admit these are intuitive arguments, however plausible at some level. But they are testable. Let us see if there are data that corroborate them. However, we first contend with possible *a priori* reasons for rejecting our intuitive arguments – the NFL theorems.

5 No Free Lunch and Penalty Function Search

Wolpert and Macready (1995, 1997) introduced the No Free Lunch theorem for search which shows that any two blackbox algorithms (deterministic or stochastic) have the same average performance over all possible problems. Some implications of this result

are that (1) all blackbox algorithms have the same average performance as random search over all problems and (2) showing that one's favorite blackbox algorithm dominates another on a subset of problems is also showing its inferiority averaged over the remaining problems.

Using a new characterization of NFL given by Schuhmacher, Vose and Whitley (2001), Koehler (2004) looked at three types of combinatorial problems and determined when the NFL results do not apply. Igel and Toussaint (2003) provided another approach for deciding when NFL does not apply. This section continues that line of inquiry while allowing constraints to be combined with objective values (as discussed in Section 3 above) or to be used separately (as discussed in Section 4).

Let $\mathcal{X}$ be a finite search space, $\mathcal{Y}$ a finite set of objective values and $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$ be the set of all possible discrete objective functions for this problem class. We consider non-repeating, blackbox algorithms (henceforth, called just algorithms) that, after t steps, choose a new point, $x_{t+1} \in \mathcal{X}$, from the search space depending on the history of prior points examined, $x_1, \ldots, x_t$ and any function on them, including the objective function. Genetic Algorithms, Simulated Annealing, and Tabu Search largely fall into this category.

Let α represent an algorithm, t the number of iterations or steps so far, $f \in \mathcal{F}$ the particular objective function being optimized and the history of objective values as

$$Y(f, t, \alpha) = \langle f(x_1), \ldots, f(x_t) \rangle .$$

A performance measure, ρ, maps $Y(f, t, \alpha)$ to the real numbers. Wolpert and Macready (1995, 1997) proved the following No Free Lunch theorem.

Theorem 1 *(NFL-1):*

For any two algorithms α_1 and α_2, any $t \in \{1, \ldots, |\mathcal{X}|\}$, any $z \in \Re$, and any performance measure, ρ,

$$\sum_{f \in \mathcal{F}} \delta(z, \rho(Y(f, t, \alpha_1))) = \sum_{f \in \mathcal{F}} \delta(z, \rho(Y(f, t, \alpha_2))) .$$

Here $\delta(\ )$ is the usual Kronecker delta function. This theorem states precisely the informal characterization above, viz., that any two algorithms, α_1 and α_2, will have the same performance when averaged over all possible problems.

Remark (Technicalities): The NFL model applies also to (1) algorithms that can revisit already observed points by recording all discovered points and outputting only newly discovered ones (Wolpert and Macready 1997); (2) stochastic algorithms, which in practice use pseudorandom number generators, so including these with an initial seed converts a stochastic algorithm to a deterministic one (Radcliffe and Surry 1995); (3) algorithms that stop since they can be restarted with a randomly chosen new point in the search space; and (4) continuous problems if they are run on finite computing machines.

This last point pertains to the problem we study later in this paper.

A common criticism of NFL-1 is that one does not usually claim their algorithm is able to solve "all instances of problems" better than some other method and so NFL may not apply. Schuhmacher, Vose and Whitley (2001) examined whether subsets of $\mathcal{F}$ could also have such implications. While affirming this they present an alternative characterization of NFL that we find useful. Let π be a permutation mapping of $\mathcal{X}$ (i.e., $\pi : \mathcal{X} \to \mathcal{X}$) and $\Pi(\mathcal{X})$ be the set of all permutation mappings of $\mathcal{X}$. That is, if one considers $\mathcal{X}$ as an ordered set, π is a reordering of the set. A set $F \subseteq \mathcal{F}$ is "closed under permutation of $\mathcal{X}$" (or "F is cup" for short) if for any $\pi \in \Pi(\mathcal{X})$ and any $f \in F$ then $f \circ \pi \equiv g \in F$.

Theorem 2 *(NFL-2):*

For any two algorithms α_1 and α_2, any $t \in \{1, ..., |\mathcal{X}|\}$, any $z \in \Re$, any $F \subseteq \mathcal{F}$ and any performance measure, ρ,

$$\sum_{f \in F} \delta\left(z, \rho\left(Y\left(f, t, \alpha_1\right)\right)\right) = \sum_{f \in F} \delta\left(z, \rho\left(Y\left(f, t, \alpha_2\right)\right)\right)$$

if and only if F is cup.

We now operationalize the idea of "cup" to examine constrained problems handled using penalty functions. Suppose $\mathcal{Z}$ is a finite set of non-negative values and $\mathcal{G} = \mathcal{Z}^{\mathcal{X}}$. One interpretation might be that $g \in \mathcal{G}$ specifies the level of infeasibility for each entry of $\mathcal{X}$ (a zero level means the related point is feasible). For example, this might arise in a problem having just one constraint or many constraints mapping to some overall measure of infeasibility (say by a linear combination).

As discussed earlier, there are several approaches used by blackbox algorithms to handle infeasibilities. A common approach to solving constrained problems (whether by blackbox algorithms or some other approach) is to convert them into unconstrained problems. Two approaches in our context are immediate. In the first approach, we can redefine $\mathcal{X}$ to include only feasible points. For example, when using genetic algorithms as a search engine, some problems naturally having infeasibilities can be searched using operators that maintain feasibility. Operators such as cycle crossover, order crossover and partially matched crossover (see Goldberg 1989) maintain the feasibility of tours in traveling salesman problems. However, these problem sets may still fall under the NFL results.

A second approach is to keep the original search space, $\mathcal{X}$, but incorporate penalties into the objective function for violations of the constraints. For an objective function $f \in \mathcal{F}$, and infesibility measure $g \in \mathcal{G}$, and $\lambda > 0$, a standard penalty function objective value looks like $p(x) = f(x) - \lambda g(x)$ for $x \in \mathcal{X}$. Suppose $F \subseteq \mathcal{F}$ and $G \subseteq \mathcal{G}$ where both F and G are cup. Let

$$\mathcal{P}_\lambda = \{f - \lambda g : f \in F, g \in G\}.$$

Clearly $\mathcal{P}_\lambda$ is also cup and the NFL results would hold. So, even allowing any penalty function appears to suffer the same fate as the original algorithm.

Suppose, however, that the infeasibility measure $g \in \mathcal{G}$ is fixed and has at least two different values (e.g., there is at least one feasible and one infeasible point of $\mathcal{X}$). Define $\mathcal{P}_\lambda(g) = \{f - \lambda g : f \in F\}$. We will show that $\mathcal{P}_\lambda(g)$ cannot be cup and, hence, that the No Free Lunch results do not apply on $\mathcal{X}$. Before giving this result, we give a matrix interpretation of cup.

Let $\overline{F}$ be an $|\mathcal{X}|$ by $|F|$ matrix formed using the elements of F as columns, we have the following alternative definition.

Definition *(Closed under Permutation)*

F is cup if and only if for any row permutation matrix P there is a column permutation Q_P such that $\overline{F} = P\overline{F}Q_P$.

With this view of cup, we have the following useful result.

Lemma 3:

If F is cup then $\overline{F}e = \beta e$ for some β where e is a vector of ones of size $|F|$.

Proof:

For any permutation matrix P, $\overline{F}e = P\overline{F}Q_P e = P\overline{F}e$. Then $\overline{F}e$ must have identical row elements.

□

The following shows that $\mathcal{P}_\lambda(g)$ cannot be cup when g has at least two distinct values even though F is cup. In the following, the transpose of a vector (say x) is given as x'.

Theorem 4*:*

Suppose F is cup. For fixed g with at least two different values and any non-zero λ, $\mathcal{P}_\lambda(g)$ is not cup.

Proof*:*

Suppose $\mathcal{P}_\lambda(g)$ is cup. Then for any $f \in F$ and permutation matrix, P, $Pf - \lambda Pg \in \mathcal{P}_\lambda(g)$. Then the columns of $\overline{F} - \lambda ge'$ are the elements of $\mathcal{P}_\lambda(g)$ where e is a vector of ones of size $|F|$. Since $\mathcal{P}_\lambda(g)$ is cup, for any permutation matrix P there is a column permutation Q_P such that

$$\left(P\overline{F} - \lambda Pge'\right)Q_P = \overline{F} - \lambda ge'.$$

Summing over all P gives

$$\sum_P \left(P\overline{F} - \lambda Pge'\right) Q_P = |\mathcal{X}|! \left(\overline{F} - \lambda ge'\right).$$

Postmultiplying by vector e and dividing both sides by $|\mathcal{X}|$! gives

$$\frac{1}{|\mathcal{X}|}\left(ee'\overline{F}e - \lambda ee'ge'e\right) = \overline{F}e - \lambda ge'e .$$

(Note, $\sum_P P = \left(|\mathcal{X}| - 1\right)! ee'$). Noting that $\overline{F}e = \beta e$ for some β, since F is cup, and collecting terms gives

$$\gamma e = \lambda g$$

for some γ. Since $\lambda \neq 0$ this gives that g is proportional to the vector of ones. However, this contradicts the requirement that g has at least two different values. Hence, $\mathcal{P}_\lambda(g)$ is not cup.

□

Remark: The motivation for assuming there is a fixed g comes from considering a search space that has additional structure (such as constraints) that are immutable regardless of the particular objective values. So Theorem 4 shows that the NFL theorems do not apply to any blackbox algorithm searching over $\mathcal{X}$ using penalty functions $p \in \mathcal{P}_\lambda(g)$ when g has at least two different values.

FI-2Pop GA does not directly use penalty functions but rather operates over two populations that are subsets of $\mathcal{X}$. The feasible population has members, x, where $g(x) = 0$ and fitness function $f(x)$. The infeasible population has members with non-zero $g(x)$ and greater fitness based on smaller $g(x)$ values. When the two populations evolve, they may produce members for the other population. We contend that this procedure will also escape the NFL curse.

To capture the driving objective of FI-2Pop GA, we first form a composite search function representing the two parts of the FI-2Pop GA approach. We also use the standard GA view of fitness where more is better. For any $f \in F$ define a related function $\overline{f}$ defined by

$$\overline{f}(x) = \begin{cases} M - \lambda g(x) & g(x) > 0 \\ \\ f(x) - \lambda g(x) & g(x) = 0 \end{cases}$$

where $M > \lambda \max_{x \in \mathcal{X}} g(x)$ and $\lambda > 0$. This function captures the effective overall search function used in FI-2Pop GA. Let $F_{M,g,\lambda}$ be the set of these functions for a fixed M, g and $\lambda > 0$.

Remark: Typically GA fitness values are required to be positive so selection operators, such as the roulette method (Goldberg , 1989), don't lead to divisions by zero. Provided $\mathcal{Y}$ has positive entries, the choice of M above is sufficient to guarantee a positive fitness measure.

As was done earlier with F, we find it useful to have a matrix counterpart for $F_{M,g,\lambda}$. For fixed $g \in \mathcal{G}$ let

$$\bar{g}(x) = \begin{cases} 1 & g(x) > 0 \\ \\ 0 & g(x) = 0 \end{cases}$$

We can write a matrix of $F_{M,g,\lambda}$ functions as

$$\bar{F} - d(\bar{g})\bar{F} + M\bar{g}e'$$

where $d(\bar{g})$ is the diagonal matrix having diagonal $\bar{g}$.

The following shows that these $F_{M,g,\lambda}$ functions also escape NFL.

Theorem 5*:*

Suppose F is cup. For fixed g with at least two different values and positive λ, $F_{M,g,\lambda}$ is not cup.

Proof*:*

Suppose $F_{M,g,\lambda}$ is cup. Since $F_{M,g,\lambda}$ is cup, for any permutation matrix P there is a column permutation Q_P such that

$$\left(P\bar{F} - Pd(\bar{g})\bar{F} + PM\bar{g}e' - \lambda Pge'\right)Q_P = \bar{F} - d(\bar{g})\bar{F} + M\bar{g}e' - \lambda ge'.$$

Summing over all P, postmultiplying by vector e and dividing both sides by $|\mathcal{X}|!$ gives

$$\frac{ee'}{|\mathcal{X}|}\left(\bar{F}e - d(\bar{g})\bar{F}e + M\bar{g}e'e - \lambda ge'e\right) = \bar{F}e - d(\bar{g})\bar{F}e + M\bar{g}e'e - \lambda ge'e.$$

Noting that $\bar{F}e = \beta e$ for some β, since F is cup, and collecting terms gives

$$\gamma e = M\bar{g} - \lambda g$$

for some γ. Let $x_1, x_2 \in \mathcal{X}$ such that $g(x_1) \neq g(x_2)$. There are three cases.

$\bar{g}(x_1)$	$\bar{g}(x_2)$	Implication
0	1	$M = \lambda g(x_2)$
1	0	$M = \lambda g(x_1)$
1	1	$\lambda g(x_1) = \lambda g(x_2)$

All three lead to contradictions. In the first two cases the contradiction arises from the fact that

$$M > \lambda \max_{x \in \mathcal{X}} g(x) \geq g(x_k)$$

for k=1,2. In the third case the contradiction comes from the fact that $g(x_1) \neq g(x_2)$. Hence $F_{M,g,\lambda}$ is not cup.

□

As noted, when any constraint set is possible, the resulting problem class $\mathcal{P}_\lambda = \{f - \lambda g : f \in F, g \in G\}$ still falls under NFL. That is for any subset $\bar{G}$ of G for which one's algorithm does better than random search, then on the complement subset the algorithm must do worse than random search. Theorem 5 shows that the NFL results do not apply to constrained problem classes involving "fixed constraints". Many practical problems have constraints that reflect domain structure (like routes or relationships) that do not change very often, so considering settings with fixed constraints is not unreasonable.

When the constraints are fixed and NFL does not apply, a given algorithm may be better or worse on average than random search. FI-2Pop GA is one such algorithm. Whether it performs better or worse than random search for a fixed constraint set, on average, is unknown. So we turn to empirical insights. Let us see if there are data that corroborate them for the Yuan problem.

6 Example Problem: Yuan

Empirical investigations are by nature focused on particulars. We discuss results for a single constrained optimization problem, called Yuan, as it is solved by the FI-2Pop GA. The particular results we present for Yuan are entirely typical and representative in our experience. Generalization to other constrained optimization problems is, of course, more problematic. But one has to begin somewhere and finding felicitous ways of empirically examining particular cases is surely among the foundational elements of the enterprise.

Yuan, the model for our case study, is discussed in Floudas, et al. (1999), page 266 and was originated in Yuan et al. (1988). The model is nonlinear (quadratic and logarithmic) in the objective function and quadratic in the constraints. Moreover, it is a mixed-integer model, with three continuous variables and four binary variables. Genocop III is unable to handle models of this sort (Michalewicz, 2003). Yuan's formulation is as follows.

Objective function:

$$\min_{\mathbf{x},\mathbf{y}} z = (y_1 - 1)^2 + (y_2 - 2)^2 + (y_3 - 1)^2 - \ln(y_4 + 1) + (x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 \quad (3)$$

Constraints:

$$y_1 + y_2 + y_3 + x_1 + x_2 + x_3 \le 5 \quad (4)$$

$$y_3^2 + x_1^2 + x_2^2 + x_3^2 \le 5.5 \quad (5)$$

$$y_1 + x_1 \le 1.2 \quad (6)$$

$$y_2 + x_2 \le 1.8 \quad (7)$$

$$y_3 + x_3 \le 2.5 \quad (8)$$

$$y_4 + x_1 \le 1.2 \tag{9}$$

$$y_2^2 + x_2^2 \le 1.64 \tag{10}$$

$$y_3^2 + x_3^2 \le 4.25 \tag{11}$$

$$y_2^2 + x_3^2 \le 4.64 \tag{12}$$

$$x_1, x_2, x_3 \ge 0 \tag{13}$$

$$y_1, y_2, y_3, y_4 \in \{0,1\} \tag{14}$$

As reported by Floudas, et al. (1999), page 266 and Yuan et al. (1988), the value of the objective function at optimality is $z^* = 4.5796$, with the decision variables set at $\mathbf{x}^{*'} = (0.2, 0.8, 1.908)'$ and $\mathbf{y}^{*'} = (1,1,0,1)'$.

We describe now results from one quite typical run of the FI-2Pop GA on the Yuan problem. (The FI-2Pop GA quite reliably solves Yuan, at least to a close approximation, regardless of which (pseudo) random stream is used to drive the algorithm. In general, we have found the FI-2Pop GA to be robust to mutation rate, crossover rate, and random number streams. We have not systematically investigated changing the population sizes from 50, which we use here, as does GENOCOP.) At the end of 5,000 generations of alternating feasible and infeasible genetic exploration (fitness evaluation effort equivalent to 10,000 generations in an ordinary GA), z^+, the objective function value of the best solution found, is 4.579588292413069. The variable settings in this solution are $\mathbf{y}' = (1,1,0,1)$, $x_1 = 0.199998178908325$, $x_2 = 0.799999776184869$, $x_3 = 1.90787728616851$.

Table 1 presents selected summary data from the feasible population, and Table 2 presents corresponding data from the infeasible population. Table 3 is a statistical summary, comparing the feasible and infeasible populations during this (typical) run.

Gen	avg(z)	med(z)	best(z)	Out
0	12.69753170008833	12.398377718036665	7.298187351684621	0
1	10.52503068528554	7.630487622288854	7.298187351684621	9
⋮	⋮	⋮	⋮	⋮
132	5.36305664951297	4.610611805470927	4.610611805470927	6
133	4.99901681120808	4.610611805470927	4.610611805470927	7
134	4.81104648604527	4.610611805470927	4.603632841228659	4
135	4.96313052073980	4.610611805470927	4.603632841228659	6
136	5.26538645226870	5.079621323943361	4.603632841228659	6
⋮	⋮	⋮	⋮	⋮
4995	4.68126360720397	4.579599345059113	4.579591950631037	5
4996	4.60738886320887	4.579603549120526	4.579591799618665	4
4997	4.62737103827847	4.579599499443619	4.579589330419115	3
4998	4.73349190807531	4.579603549120526	4.579588292413069	1
4999	4.59961540370377	4.579597549265095	4.579588292413069	2

Table 1. By-generation summary of the feasible population in a typical run of Yuan. Gen=generation. avg(z)=average objective function value by generation. med(z)=median objective function value by generation. best(z)=best solution found so far. Out=number of infeasible progeny produced by generation.

6.1. Evaluation of the Feasible Population

The feasible population rather quickly finds solutions within 0.5% of optimal by generation 136. Table 3 shows a subsequent slow, steady improvement in z^+ on the average. The variance of the feasible population appears to stabilize (Table 3, column 7), but cannot go to zero because of mutation and continuing immigration. In any event, the two-population GA evidences excellent performance on this challenge problem from the literature, which is nonlinear in both the objective function and the constraints, and has a mixture of integer and real (floating point) decision variables.

6.2. Evolution of the Infeasible Population

As seen in Table 3, the average infeasibility of solutions in the infeasible population

becomes closer to 0 as the run progresses. In the run under display here, the average infeasible solution moved towards feasibility from a distance of 0.2005 during generations 900 through 999, to a distance of 0.0126 during generations 4900 through 4999. (Distance is measured as the sum of constraint violations.)

The infeasible solutions are not subjected to selection with regard to the z (objective function) values. Most interestingly, the z values in the infeasible population nevertheless clearly move towards z^+ (or z^*) as the run progresses. In the end, the best infeasible z values are not far from the optimal value, z^*. Compare the rightmost two columns of Table 3. Note that there is an overall reduction in the variance in z in the infeasible population as the generations progress. This is in contrast to the variance in z for the feasible population, which does not appear to decline during the last 1000 generations.

Gen	avg(z)	med(z)	avgInfeas	Out
0	8.143224744257274	8.368698923454343	-1.847537037871579	0
1	8.397975429018375	9.022887244519183	-1.117432495052733	1
⋮	⋮	⋮	⋮	⋮
132	6.962531847852802	6.519229712067366	-0.2484526749970332	5
133	7.267970477305616	6.823406269075488	-0.1299943868051112	2
134	7.706505082745875	6.038927899546921	-0.0885399858116163	1
135	6.521347185201272	5.257514914110913	-0.1733602611829584	5
136	7.175989168243694	7.516553449635433	-0.1418062323207774	6
⋮	⋮	⋮	⋮	⋮
4995	5.715105134746020	4.926231680014258	-3.1601861983921E-6	0
4996	5.516678970813462	4.926229526679555	-0.0159576078217629	0
4997	5.890533806672197	4.926189743129405	-0.0159577549401262	1
4998	6.115080232697109	5.079615346523230	-2.3668425574097E-6	2
4999	5.742802940813199	4.926184475169487	-0.0159587755577582	0

Table 2. By-generation summary of the infeasible population in a typical run of Yuan

6.3. Mixing between the Feasible and Infeasible Populations

The infeasible population can produce offspring that are feasible. Although this might seem to be unlikely, our example data show the infeasible population was steadily producing feasible solutions. See the InF $\rightarrow$ Fea column in Table 3. We also note some indication of modestly declining productivity as the run progressed.

Column Fea $\rightarrow$ InF in Table 3 shows the feasible population produced infeasible offspring at roughly double the rate InF $\rightarrow$ Fea of the infeasible population. Fea $\rightarrow$ InF may, however, be declining more rapidly than InF $\rightarrow$ Fea.

These data support a consistent picture of what is happening in the two-population GA data (in this typical run for this problem, but in our experience it is representative). Selection is driving the feasible population closer to the boundary of the feasible region. Not only is there improvement in z^+ over time; there is steady but fluctuating improvement in the average z each generation. Similarly, selection is driving the infeasible population closer to the boundary separating the feasible and infeasible regions.

Generations	Infeasibility	InF $\rightarrow$ Fea	Fea $\rightarrow$ InF	z^+	med z_{InF}	$\sigma^2 z_{Fea}$	$\sigma^2 z_{InF}$
0–99	-0.2824	3.5400	7.3000	5.222503	7.123	2.302	6.839
900–999	-0.2005	3.4100	6.6200	4.594130	6.577	0.840	8.928
1900–1999	-0.0453	3.3100	6.4000	4.581232	9.468	1.015	7.713
2900–2999	-0.0858	3.0400	6.4800	4.579938	5.926	0.426	3.302
3900–3999	-0.0501	2.7000	6.3300	4.579845	5.103	0.251	1.775
4900–4999	-0.0126	3.2900	4.8200	4.579653	5.245	0.253	0.948

Table 3. Yuan Results: Averages over 100 generations. Infeasibility= $-1 \cdot$ sum of absolute violations of constraints (averaged over each solution for 100 generations). InF $\rightarrow$ Fea=number of feasible offspring from the infeasible population (by generation, averaged over 100 generations). Fea $\rightarrow$ InF =number of infeasible offspring from the feasible population (by generation, averaged over 100 generations). z^+ =best solution found in the feasible population (by generation, averaged over 100 generations). med z_{InF} =median objective function value in the infeasible population (by generation, averaged over 100 generations). $\sigma^2 z_{Fea}$ =variance of objective function values in the feasible population (averaged over all solutions in 100 generations). $\sigma^2 z_{InF}$ =variance of objective function values in the infeasible population (averaged over all solutions in 100 generations).

x_1	x_2	x_3	y_1	y_2	y_3	y_4
0.019176639369240966	0.12188533340384183	1.5120185756147242	1	1	0	1

Table 4. Gen=20, z=8.010270704398303, sumSlack= 10.515981932347778. First appearance of (1, 1, 0, 1) in the feasible population.

6.4 Genetic Benefits of Population Immigration

The ebb and flow of offspring from one population to the other has two benefits. First, useful genetic materials are less likely to be lost than in a one-population GA. Second, these useful materials tend to diffuse into both populations.

Detailed examination of the run reveals that alleles and patterns of alleles (building blocks) will often arise in one population, quickly move to the other, and then be maintained indefinitely in both populations. Consider, for example, the pattern $\mathbf{y}' = (1,1,0,1)$ that appears in z^*. As shown in Table 4, this pattern first appears in the feasible population in generation 20. It is lost to the feasible population, but reappears beginning in generation 26 of the infeasible population and is present in generally increasing numbers for the remainder of the run.

As shown in Table 5, starting in generation 26 this pattern maintains its presence in most of the remaining generations of the infeasible population. The floating point x_i variables evidence a similar dynamic. All of this supports the hypothesis that both populations are actively exploring valuable genetic material.

x_1	x_2	x_3	y_1	y_2	y_3	y_4
1.0255729805767875	0.556585764572967	1.5120185756147242	1	1	0	1

Table 5. Gen=26, z=5.605040171124676, sumInfeas = -1.7453232819180533.
First appearance of (1, 1, 0, 1) in the infeasible population.

6.5 Potential Tradeoffs Revealed in Infeasible Population

At generation 236, there appears in the infeasible population a solution with **z** = 4.47002605609438, which is much better (smaller) than the **optimal solution** z^*. This infeasible solution is at: $x_1 = 0.195462908809646$, $x_2 = 0.795752247026746$, $x_3 = 1.96768190221611$, $y_1 = y_2 = y_4 = 1$ and $y_3 = 0$. All the variable values in this solution are close to their correspondents in z^+ (and z^*), except x_3. Further, only one constraint is violated, ($y_2^2 + x_3^2 \le 4.64$), which comes in at 4.871772068.

This is potentially valuable information. Constraints may often be relaxed, for a price. This infeasible solution at generation 236 provides a specific measure of the *shadow price* for constraint ($y_2^2 + x_3^2 \le 4.64$). If the benefit achieved by improving (here,

reducing) z^+ is greater than the cost of relaxing this constraint enough to make this solution feasible, then the decision maker has been presented with an opportunity discovered by the algorithm. Such opportunities often occur in practice. Hundreds if not thousands of specific potential such opportunities are apparent from the data in the infeasible populations of this typical run. (We note that this concept is explored, but in the context of a conventional single-population GA in Kimbrough et al. (1993, 1994) and Branley et al. (1997). Kimbrough and Wood (2006) explore the concept in the context of the FI-2Pop GA.)

7 Distance Tracing: Yuan

7.1 Distance Metric

Much can potentially be learned about the operation of the FI-2Pop GA by viewing decisions (aka: solutions, i.e., settings of the decision variables) as points in decision space and by measuring distances between decisions or populations of decisions. Given two decisions, *x* and *y***,** quite a few distance measures have appeared in the literature, but the square of the Euclidean distance between *x* and *y* is natural and robust,

$$(x-y)'(x-y)$$

and it is what we have used in our investigations. We have focused on comparing populations, and have represented the centroids of the populations as the averages of the solution vectors in each population. For our two populations we denote these averages as the vectors μ_x and μ_y.

Two complications intervene. First, as in Yuan, decision variables may be integers, and may thus have non-integral means. We simply treat all decision variables as real numbers for the purpose of computing distances. Second, distance metrics are sensitive to scale. It is appropriate, even mandatory, to convert all decision variables to a common scale for proper comparison. Further, it is desirable (for treating each decision variable equally) to remove linear associations between decision variables. To this end, we rescale our data with an inverse covariance matrix obtained from a large sample of both the feasible and infeasible populations. Specifically, we sample both populations every 25 generations, combine the results, and calculate C^{-1} the inverse covariance matrix of the sample. Incorporating this rescaling, using the covariance matrix, our scaled distance function is the square of the Mahalanobis distance measure (Mahalanobis 1936):

$$d(\mu_x, \mu_y) = (\mu_x - \mu_y)' C^{-1} (\mu_x - \mu_y).$$

We shall now discuss some of what may be learned using this distance metric.

7.2 Plots with the Metric

Figure 1 presents two dimensions of information about the evolution of the system as it confronts the Yuan problem. The density plot displays the distances between the centroids of the feasible (ordinate, vertical dimension) and infeasible (abscissa, horizontal dimension) populations for the first 200 generations of the run. Darker cells indicate smaller distances, lighter cells, larger distances. There are two lighter-colored bands, one running horizontally, one vertically (and somewhat less pronounced). The darker squarish area near the origin, in the southwest corner, evidences very little pattern, indicating no very clear trends during the first 25 or so generations. The horizontal lighter-colored band indicates that after about 30 generations every infeasible population is comparatively far away from the feasible populations of the first 25 generations or so. A similar interpretation in stronger form attends the horizontal band. North of the horizontal band and east of the vertical band lies most of the territory during the first 200 generations. This area is remarkably uniform. It does not get discernibly darker as it moves east, indicating little if any movement of the two populations towards each other. Most interestingly, however, the area is replete with vertical stripes, suggesting a function of one variable, and in particular suggesting that after roughly 30 generations every feasible population is essentially equidistant from every infeasible population (older than roughly 30 generations). Put more carefully, if we fix on any infeasible population (older than roughly 30 generations), then *every* feasible population (older than roughly 30 generations) is nearly equally distant from it. Recall that distances are for population centroids. What this indicates is that the centroid of the feasible population quickly settles down and remains fairly stable, while the centroid of the infeasible population is moving about.

*** Insert Figure 1 about here ***

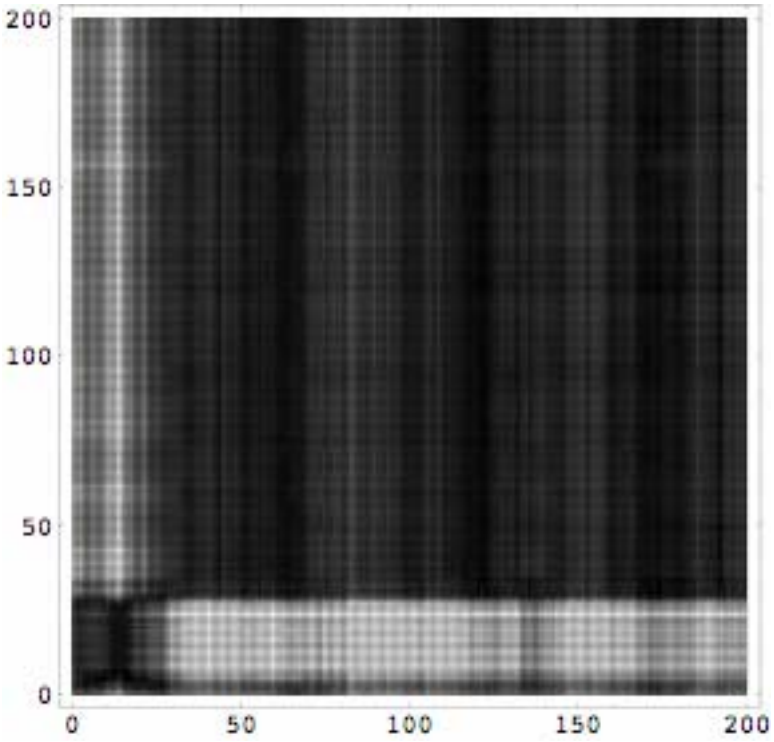

Figure 1. Yuan Plot. First 200 generations. Feasible population: abscissa. Infeasible population: ordinate. Light=further apart. Dark=closer.

Figure 2 reinforces Figure 1 by plotting the distance between the feasible and infeasible populations (sampled every 25 generations) over the entire run. (The run lasted 5000 generations, so sampling every 25 generations produces 200 data points.) Clearly, there is a strong trend of diminishing distances between the centroids of the two populations even though the two populations are being selected with quite distinct fitness functions. As measured by their centroids, the feasible and infeasible populations are getting closer to each other.

*** Insert Figure 2 about here ***

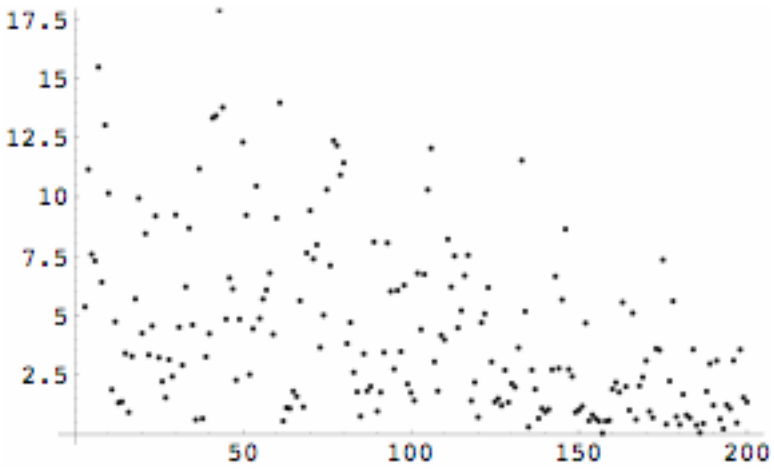

Figure 2. Yuan Plot. Distances between the feasible and infeasible populations, sampled every 25 generations.

Figure 3 also reinforces Figure 1. It plots the distances from the feasible population, sampled every 25 generations, to the infeasible population of the final generation (5000). Not much trend is apparent, indicating that the centroid of the feasible population moves very little and implying, in conjunction with Figure 2, that it is the infeasible population that does most of the moving.

*** Insert Figure 3 **feasVsInfeas200.pdf** about here ***

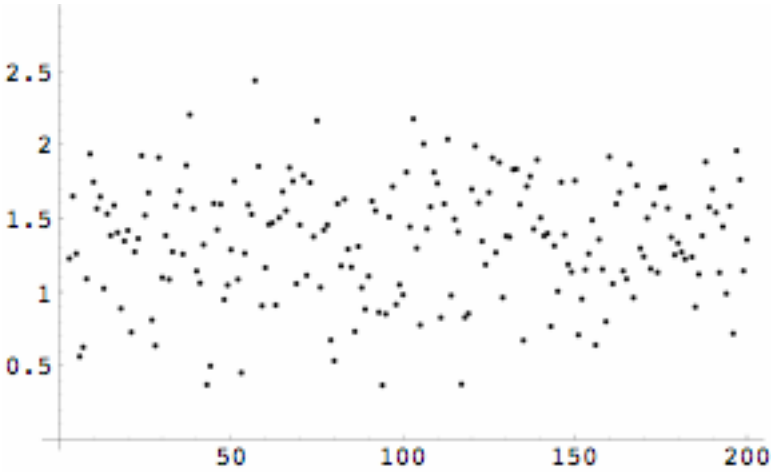

Figure 3. Yuan Plot. Distances between the feasible population and infeasible population at generation 5000 (=25*200), sampled every 25 generations.

So far, we have demonstrated that for this run of Yuan the infeasible population is moving its centroid much more than the feasible population. But is the feasible population moving at all? Figure 4, which shows the distances of the feasible populations, sampled every 25 generations, from the feasible population at generation

5000 (=200*25), establishes that it does.

*** Insert Figure 4 **ListPlotYuanFeaFea200Every25.pdf** about here ***

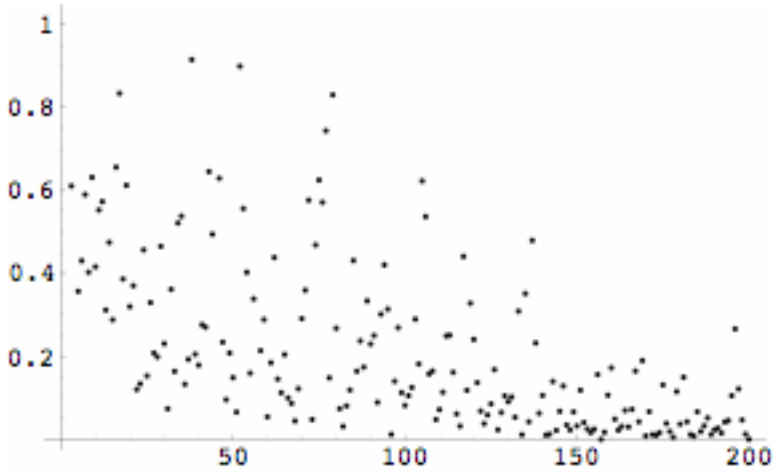

Figure 4. Yuan Plot. Every 25 generations. Feasible population: abscissa. Distance to Feasible population at generation 5000 (=200*25): ordinate.

Figure 5 is the analog of Figure 4, but for the infeasible populations instead of the feasible populations. Again, the same general pattern obtains in the two figures. What is different is the scale on the ordinate, demonstrating that the centroid of the feasible populations has moved little compared to the movement of the centroid of the infeasible populations.

*** Insert Figure 5 **ListPlotYuanInFeaInFea200Every25.pdf** about here ***

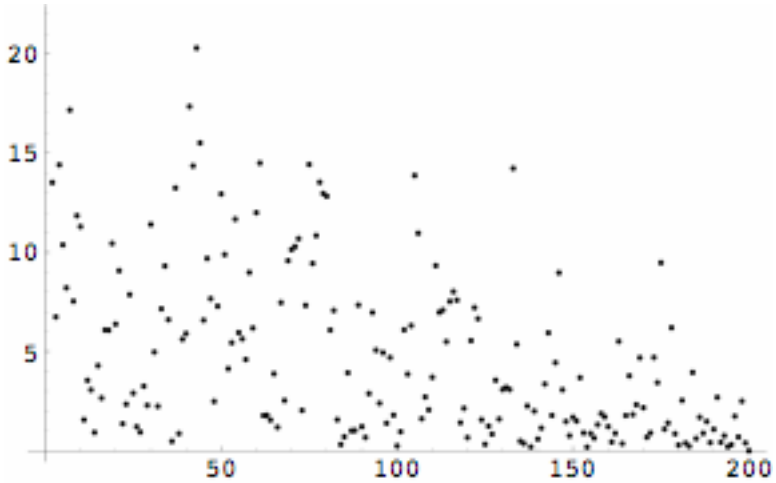

Figure 5. Yuan Plot. Every 25 generations. Infeasible population: abscissa. Distance to Infeasible population at generation 5000 (=200*25): ordinate.

Now we divide the feasible population into incumbents (offspring of feasibles) and

immigrants (offspring of infeasibles). Note that in many generations there were no feasible children of the infeasible population that survived initial selection in the feasible population. In these generations the distance between the daughters of the feasible and the daughters of the infeasible were stipulated to be -10. The effect of this is evident in Figure 6, in which we discern a modest trend of ever closer resemblance between the incumbents and the immigrants.

*** Insert Figure 6 **ListPlotFeaInctVSFeaImgtEvery25.pdf** about here ***

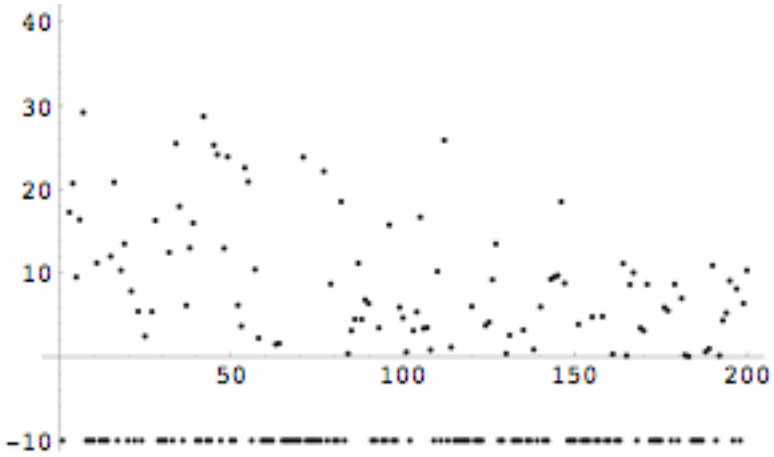

Figure 6. Yuan Plot. Every 25 generations. Feasible population, average distances between incumbents and immigrants by generation. Generation: abscissa. Incumbent-immigrant average distance: ordinate.

It is important to note that the daughters of the infeasibles showing up here have survived a competitive process in the feasible population. A feasible population (of 50) is created, then the infeasible population is processed. Any resulting feasible offspring are put into the feasible population, which is then reduced to the standard size (here 50) by fitness-proportional selection (with 1 elitism). Thus, the feasible daughters showing up have, probabilistically, competitive fitnesses in the feasible population. They are also, as the figure shows, atypical. In short, immigrants from the infeasible population succeed in the feasible population and introduce considerable variation - a healthy situation indeed for a genetic system under selection. Immigration in the opposite direction (from the feasible to the infeasible population) also occurs (but is not charted here) and indeed the feasible population is more prolific in spinning off emigrants (see also Table 3.)

As a final comment, we note what typically happens if we initialize a run but remove the initial feasible population: the overall behavior strongly resembles the charts displayed here.

8 Discussion and Conclusion

We began this investigation with (i) the presupposition that the FI-2Pop GA has a credible record in solving constrained optimization problems and (ii) intuitions regarding why this might be so. The challenge, however, is to understand the FI-2Pop GA (as well

as other metaheuristics). To this end we have here shown that the NFL results do not apply to the context (optimization with fixed constraints) in which the FI-2Pop GA is applied and we have undertaken a detailed case study of a single problem, Yuan. In §6 we reported on various summary statistics and patterns found in the data for a single run. §7 describes and exploits a method introduced here—distance tracing with the inverse covariance transform—for describing behavior by populations of decisions in an evolutionary computation. We can, in summary, make the following observations on this case:

1. The centroids of both the feasible and the infeasible populations move during the course of the GA's runs, with the infeasible population moving much more than the feasible.

2. The centroids of both populations move towards each other, and are quite close by the end of the runs.

3. The infeasible population is a continuing source of variation in the feasible population, throughout the runs.

4. Both populations become more homogeneous (are reduced in variance), the feasible population much faster than the infeasible population.

5. The infeasible population is, on average, just barely infeasible towards the end of the run, indicating that its members are all close to the boundary between the feasible and infeasible regions.

6. Starting the FI-2Pop GA with either an empty feasible population or an empty infeasible population has little effect on the macro behavior of the system.

This leads to the following description of what is happening as the FI-2Pop GA solves the Yuan problem. The feasible population rather quickly undergoes a significant reduction in variance, and concentrates in a region near or including z^+, the best solution it will find. After that, the centroid of the population moves steadily but slightly in magnitude, along with steady but modest improvements in the best decision found. Emigration from the feasible population is substantial throughout the run. Approximately 15 to 20 percent of children with feasible parents are infeasible and hence emigrate to the infeasible population each generation. Because children will resemble their parents and because the feasible population is comparatively narrow in its range, the decisions it introduces into the infeasible population will also be near the region of concentration of the feasible population. This trend is reinforced by the GA's use of nonuniform mutation: mutation sizes (but not frequencies) are reduced for the real-valued variables as the run progresses.

Interestingly, the infeasible population is also an ongoing contributor to the feasible population and to its variance. Selection on the infeasible population drives it towards the

boundary. Selection can have no direct effect on the objective function values of the infeasible decisions. Yet, strong indirect influences are observed on both variance and objective function values. This is due to immigration from the (increasingly concentrated) feasible population, which clusters nearer and nearer the feasible-infeasible boundary. Infeasible solutions comparatively far from the feasible population occasionally give rise to feasible descendants, resulting in the parent(s) being removed from the infeasible population. Thus, the infeasible population becomes increasingly concentrated (in the case of Yuan) near to the feasible population. Overall, the two populations continue to exchange solutions throughout the run. As the run progresses, the two populations are increasingly concentrated near to each other, resulting in a focused exploration in the region of z^+, and characterizing convergence of the algorithm.

That is (much of) the story for Yuan and the FI-2Pop GA. The story is a happy one in this case, and it would appear that the behavior of the algorithm is felicitous, rather than merely fortuitous. Finding the neighborhood of the optimum (as it apparently does), and focusing exploration there surely has much to recommend it as a strategy. Yuan, however, is just a particular case. To see what this may tell us in general, we need to consider the FI-2Pop GA in light of the exploration–exploitation dilemma for optimization and indeed for learning. The No Free Lunch theorems tell us that the dilemma is fundamental unless some domain specific knowledge is incorporated into the search. Fortunately, FI-2Pop GA incorporates enough problem specific information (i.e., which points are feasible and which are not) to nullify the NFL implications (as shown in Theorems 4 and 5).

Even had the NFL results not been nullified, is there anything useful to say in general about the FI-2Pop GA, or any other metaheuristic for that matter? There is.

Reverting now to a more general focus, consider standard GAs from the perspective of the exploration–exploitation dilemma. Selection is inherently exploitive, although stochastically selecting a population based on fitness admits a degree of exploration. The selection operator can be restrained in its greed. Conversely, mutation and crossover are predominantly exploratory operators, admitting a degree of exploitation by being restricted to higher fitness individuals. The FI-2Pop GA adds to this mixture of factors. The FI-2Pop GA is a source for additional variance, in both populations. As such, it adds tilt towards exploration. Of course, it cannot be guaranteed that the FI-2Pop GA will add variance to the two populations. It can be expected, however, that feasible offspring from infeasible parents will typically be somewhat different from feasible offspring of feasible parents, since the respective parents have survived in two very different selection regimes. A similar argument applies to the infeasible population.

Of course, and to repeat, there is no guarantee in general that variance will in fact be added or that if added it will prove useful. If the FI-2Pop GA fails to add variance (fails to produce feasible offspring from the infeasible population, or the feasible offspring fail to survive or survive but are not different), the resulting system is in effect operating a death penalty regime on the feasible population. If the FI-2Pop GA adds variance, but the variance is not useful, the system becomes somewhat noisier. None of these failure

conditions would appear to vitiate the search. They would tend to reduce the effectiveness of the search, but would not threaten its ultimate success, given additional resources (more generations or larger populations). In this regard, we note that the pressure of fitness proportional selection is proportional to the standard deviation of the population, which is itself proportional to the square root of the size of the population. Thus, doubling (or halving) the size of the population will less than double (or halve) the rate of response to selection. Further, as we noted above, the FI-2Pop GA made also add value by continuing to discover feasible solutions for a problem in which they are hard to find. It is not inaccurate to think of the FI-2Pop GA as including an evolutionarily-directed repair of infeasible solutions.

On the other hand, there will surely be many cases in which additional variance introduced by the FI-2Pop GA will be useful. This is of course an empirical matter, but we note that over-convergence (premature lack of variance in a population of solutions) is a well-known bugbear for GAs. The FI-2Pop GA has in fact a strong track record of avoiding this. Our thesis is that this strong track record may be explained in part by the additional variance created by the FI-2Pop GA. In this regard we note the following points, all of which are illustrated by the Yuan case.

1. Immigrants into the feasible population will have ancestors selected (with varying degrees of rigor) for proximity to the boundary of the feasible region. Especially if created by mutation, the immigrants will have a tendency to locate near their ancestors, and hence near the boundary, but on the feasible side. This is a desirable property if the optimum decisions (or even many of the very good decisions) are on or near the boundary.

2. We note as well that the algorithm's treatment of infeasible solutions may be viewed as a generic repair mechanism. Repair is the objective of selection pressure towards feasibility acting on the infeasible population.

3. Earlier on in a run, immigrants into the feasible population will tend not to have, and hence tend not to resemble, feasible ancestors. This will tend to introduce a larger variance into the feasible population.

4. Later on in a run, many immigrants into the feasible population will have, and will resemble, feasible ancestors, and thus will introduce less absolute variance into the feasible population. The effect is to localize the added variance in a region of concentration of the feasible population. If indeed a region of concentration is the neighborhood of good decisions, the effect is to intensify the local search.

5. The distributions of the feasible and infeasible solutions, including their trajectories over time, may be diagnostic and yield some degree of assurance that the best solutions found are near-optimal. In particular, the convergence of the algorithm may be judged by the extent to which both populations tend to concentrate in specific region(s). In the Yuan runs, the tight clustering near

one another of the feasible and infeasible populations suggests that the algorithm has indeed discovered a region of the previously-published optimal solution for the problem.

In conclusion, the FI-2Pop GA is seen, both from first principles and from the run data analyzed above, to systematically introduce variance into the feasible population in ways not available to the conventional penalty function approaches. Whether this additional variance often proves useful must be determined by experience. To date, that experience is favorable and the FI-2Pop GA is a credible distinct option, perhaps as part of a suite of approaches, for solving constrained optimization problems with other evolutionary algorithms (EAs). The distance mapping technique for studying the performance of EAs, introduced here, offers the prospect of insight into alternate methods, which insight will complement experience from benchmark testing.

In general, we have also shown that any blackbox algorithm escapes the NFL implications if it optimizes a penalty function with immutable constraints (where there is at least on feasible and one infeasible point). This carries over to settings (like FI-2Pop GA) where a penalty function is implicitly used to drive the search.

References

Bäck, T. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, New York, NY, 1996.

Bäck, T. David Fogel, and Zbigniew Michalewicz, editors. Handbook of Evolutionary Computation. Institute of Physics Publishing, http://www.iop.org/Books/CIL/HEC/index.htm, 1997.

Bäck, T., David Fogel, and Zbigniew Michalewicz, editors. Advanced Algorithms and Operators. Institute of Physics Publishing, Bristol, UK, 2000.

BenHamida, S. and M. Schoenauer. An Adaptive Algorithm for constrained optimization problems. In Parallel Problem Solving in Nature (PPSN 2000), pages 529-539, Berlin, Germany. 2000. Springer-Verlag.

BenHamida, S. and M. Schoenauer. ASCHEA: New Results Using Adaptive Segregational Constraint Handling. In Congress on Evolutionary Computing (CEC 2002), pages 884-889, 2002. IEEE Press.

Branley, B. and R. Fradin, S. O. Kimbrough, and T. Shafer. On heuristic mapping of decision surfaces for post-evaluation analysis. In Ralph H. Sprague, Jr., editor, Proceedings of the Thirtieth Annual Hawaii International Conference on System Sciences, Los Alamitos, CA, 1997. IEEE Press.

Chu, P. C. and J. E. Beasley. A genetic algorithm for the generalized assignment problem. Computers and Operations Research, 24(1):17–23, 1997.

Chu, P. C. and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics, 4(1):63–86, June 1998.

Coello, C. A. C. A survey of constraint handling techniques used with evolutionary algorithms. Technical report Lania-RI-99-04, Laboratorio Nacional de Inform´atica Avanzada, Veracruz, M´exico, 1999. http://www.lania.mx/˜ccoello/constraint.html.

Coello, C. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering, 191(11-12):1245-1287, 2002.

Coello, C. A. C. List of references on constraint-handling techniques used with evolutionary algorithms. World Wide Web, Accessed January 2003. http://www.cs.cinvestav.mx/˜constraint/index.html.

Eiben, A. E. Evolutionary algorithms and constraint satisfaction: Definitions, survey, methodology, and research directions. In Leila Kallel, Bart Naudts, and Alex Rogers, editors, Theoretical Aspects of Evolutionary Computing, pages 13–30. Springer-Verlag, Berlin, Germany, 2001.

Eiben A. E. and J. E. Smith. Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Germany, 2003.

Floudas, C. A., P. M. Pardalos, C. S. Adjiman, W. R. Esposito, Z. H. Gumus, S. T. Harding, J. L. Klepeis, C. A. Meyer, and C. A. Schweiger. Handbook of Test Problems in Local and Global Optimization. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

Goldberg, D. E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, 1989.

Holland, J. H. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, 1975.

Igel, C. and M. Toussaint. On Classes of Functions for which No Free Lunch Results Hold, Information Processing Letters, 86(6), 317-321, 2003.

Kimbrough, S. O. and J. R. Oliver. On automating candle lighting analysis: Insight from search with genetic algorithms and approximate models. In Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., editors, Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences, Volume III: Information Systems: Decision Support and Knowledge-Based Systems, pages 536–544, Los Alamitos, CA, 1994. IEEE Computer Society Press.

Kimbrough, S. O. and D. H. Wood. On how solution populations can guide revision of model parameters. Late breaking papers, GECCO 2006, July 2006. lbp133.pdf at GECCO 2006. Available at http://opim-sky.wharton.upenn.edu/~sok/sokpapers/2007/lbp133.pdf.

Kimbrough, S. O., J. R. Oliver, and C. W. Pritchett. On post-evaluation analysis: Candlelighting and surrogate models. Interfaces, 23(7):17–28, May-June 1993.

Kimbrough, S. O., M. Lu, and S. M. Safavi. Exploring a financial product model with a two-population genetic algorithm. In Proceedings of the 2004 Congress on Evolutionary Computation, pages 855–862, Piscataway, NJ, June 19–23, 2004a. IEEE Neural Network Society, IEEE Service Center. ISBN: 0-7803-8515-2.

Kimbrough, S. O., M. Lu, D. H. Wood, and D. J. Wu. Exploring a two-market genetic algorithm. In W. B. Langdon, E. Cant´u-Paz, and et al., editors, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pages 415–21, San Francisco, CA, 2002. Morgan Kaufmann.

Kimbrough, S. O., M. Lu, D. H. Wood, and D. J. Wu. Exploring a twopopulation genetic algorithm. In Erick Cant´u-Paz and et al., editors, Genetic and Evolutionary Computation (GECCO 2003), LNCS 2723, pages 1148–1159, Berlin, Germany, 2003. Springer.

Kimbrough, S. O., M. Lu, and D. H. Wood. Exploring the evolutionary details of a feasible-infeasible two-population GA. In Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), Birmingham, UK, 18-22 September 2004b.

Koehler, G. J. Conditions that Obviate the No Free Lunch Theorems for Optimization. Forthcoming Informs Journal on Computing.

Le Riche, R. and R. T. Haftka. Improved genetic algorithm for minimum thickness composite laminate design. Composites Engineering, 3(1):121–139, 1995a.

Le Riche, R., C. Knopf-Lenoir, and R. T. Haftka. A segregated genetic algorithm for constrained structural optimization. In Proceedings of the Sixth International Conference on Genetic Algorithms, pages 558–565. Morgan Kaufmann, 1995b.

Mahalanobis, P. C. On the generalised distance in statistics, Proceedings of the National Institute of Science of India, 12: 49-55, 1936.

Michalewicz, Z., D. Dasgupta, R.G. Le Riche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. Computers & Industrial Engineering Journal, 30(2):851–870, 1996.

Michalewicz, Z. A survey of constraint handling techniques in evolutionary computation methods. In Proceedings of the 4th Annual Conference on Evolutionary Programming, pages 135–155, Cambridge, MA, 1995. MIT Press. http://www.coe.uncc.edu/˜zbyszek/papers.html.

Michalewicz, Z. Do not kill unfeasible individuals. In Proceedings of the Fourth Intelligent Information Systems Workshop (IIS'95), Dabrowski, Michalewicz, Ras, eds., Augustow, Poland, June 5-9, pp. 110-123, 1995a.

Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin, Germany, third edition, 1996.

Michalewicz, Z. Numerical optimization: Handling nonlinear constraints. In Bäck, T. David Fogel, and Zbigniew Michalewicz, editors. Handbook of Evolutionary Computation. Institute of Physics Publishing, http://www.iop.org/Books/CIL/HEC/index.htm, 1997, page G9.9.

Michalewicz, Z. Genocop – optimization via genetic algorithms. World Wide Web, Accessed January 2003. http://www.cs.sunysb.edu/_algorith/implement/genocop/implement.shtml.

Michalewicz, Z. and D. B. Fogel. How to Solve It: Modern Heuristics. Springer, Berlin, Germany, 2000.

Radcliffe, N. J. and P. D. Surry. Fundamental imitations on search algorithms: Evolutionary computing in perspective. In J. van Leeuwen, editor, *Lecture Notes in Computer Science 1000*. Springer-Verlag, 1995.

Sarker, R., M. Mohammadian, and X. Yao, editors. Evolutionary Optimization. Kluwer, Boston, MA, 2002.

Schoenauer, M. and Z. Michalewicz. Evolutionary computation at the edge of feasibility. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, eds., Proceedings

of the 4th Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Vol. 1141, pages 245-254, Springer-Verlag, Berlin, Germany, 1996.
Schoenauer, M. and Z. Michalewicz. Sphere operators and their applicability for constrained parameter optimization problems. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, eds., Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, Lecture Notes in Computer Science, Vol. 1447, pages 214-250, Springer-Verlag, Berlin, Germany, 1998.
Schuhmacher, C., M. D. Vose, and L. D. Whitley. The no free lunch and description length. In L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, Genetic and Evolutionary Computation Conference (GECCO 2001), pages 565–570. Morgan Kaufmann, 2001.
Smith, A. E. and D. W. Coit. Penalty functions. In Bäck, T. David Fogel, and Zbigniew Michalewicz, editors. Handbook of Evolutionary Computation. Institute of Physics Publishing, http://www.iop.org/Books/CIL/HEC/index.htm, 1997, page C5.2. 6
Wolpert, D. H. and W. G. Macready. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1(1):67–82, 1997.
Wolpert, D. H. and W. G. Macready. No free lunch theorems for search. Working paper sfi-tr-95-02-010, Santa Fe Institute, 1995.
Yuan, X., S. Zhang, and L. Pibouleau. A mixed-integer nonlinear-programming method for process design. RAIRO - Recherche Operationnelle-Operations Research, 22(3):331–346, 1988.
Yuchi, M. and J. Kim. Grouping-based evolutionary algorithm: Seeking balance between feasible and infeasible individuals of constrained optimization problems. In Proceedings of the 2004 Congress on Evolutionary Computation, pages 280–7, Piscataway, NJ, June 19–23, 2004. IEEE Neural Network Society, IEEE Service Center.

A Specifics of the Feasible-Infeasible Two-Population GA

We describe here key details of the two-population GA for constrained optimization used in the work reported in this paper. The algorithm is essentially that used in the previous constrained optimization studies (Kimbrough, et al. 2002, 2003, 2004a, 2004b). All key parameters were set *ex ante*, were not tuned for any of the problems, and were set identically for all problems. Two populations of solutions are created at initialization and maintained throughout a run: the *feasible population* consists of only feasible solutions to the constrained optimization problem; the *infeasible population* consists of only infeasible (constraint-violating) solutions. Each population is initialized to a size of 50 and, with a qualification described below, this size is maintained throughout the run. The number of generations was 5,000 for each population, corresponding to 10,000 generations in a normal GA. Floating point encoding, rather than binary encoding, is used for real-valued alleles.

Initialization proceeds one population at a time, beginning with the feasible population. A solution is randomly generated and its feasibility determined. If it is feasible the solution is placed into the feasible population; otherwise it is discarded. This continues until 50 feasible solutions are obtained or 5,000 attempts have been made. The algorithm proceeds even if fewer than 50 feasible solutions are found. The analogous process is conducted to initialize the infeasible population.

The two-population GA maintains 4 collections of solutions at each generation: the feasible population, the feasible pool, the infeasible population, and the infeasible pool. Creation of the next generation begins by processing the feasible population of the current generation. Fitness (as objective function value) is calculated for each member of the population and 50 new solutions are generated using the genetic operators (in order): fitness-proportional selection of parents, crossover (probability 0.4, single-point), and mutation (at probability 0.4), non-uniform mutation for floating point alleles, with degrees (*b* in Michalewicz's (1996) formula pages 103 and 111-2) equal to 2. The feasibility of each of the 50 solutions is determined and each solution is placed in one of two pools, either a (next generation) feasible pool or an infeasible pool, as appropriate. The current generation infeasible pool is added to the just-created infeasible pool. (Thus, the 'infeasible population' at generation 0 is really the infeasible pool at generation 0.) Fitness (as sum of constraint violations) is then calculated for each member of the current infeasible pool. If necessary, the size of the infeasible pool is reduced to 50, based on fitness. The result is the next generation infeasible population. Using the next generation infeasible population, 50 new solutions are generated as before. Feasible results are placed in the next generation feasible pool. If necessary, the size of the feasible pool is reduced to 50, based on fitness. The result is the next generation feasible population. Infeasible results are placed in the next generation infeasible pool and the contents of the next generation infeasible population are also placed into the next generation infeasible pool. This completes one generation. Processing continues until 5,000 generations have elapsed. The parameter values, such as crossover probability, mutation rate, *b*, population size, and number of generations, were not tuned for the Yuan problem. They were set antecedently, based on a number of criteria, both pragmatic and performance. Other parameter settings may well result in better performance, but finding best performance of the FI-2Pop GA is not a main subject of this paper.