Exploring a Financial Product Model with a Two-Population Genetic Algorithm

Steven O. Kimbrough
Ming Lu
University of Pennsylvania
Operations & Information Management Dept.
Philadelphia, PA 19104
Email: {kimbrough, milu}@wharton.upenn.edu

Soofi M. Safavi
University of Pennsylvania
Systems Engineering
Email: ssafavi@seas.upenn.edu

***Abstract*— This paper describes a successful application of evolutionary computation to a difficult and commercially significant constrained optimization problem. The Financial Product model, for optimizing mortgage refinancing packages, is introduced. It is a realistic, and very challenging, optimization problem, for which standard solvers leave much to be desired. An untuned two-population genetic algorithm (GA) has been remarkably successful in finding good, feasible and nearly optimal solutions. In addition, the genetic solver provides important information for management decision making besides simply a good solution to the model. Finally, the paper undertakes a case study in order to investigate the details of how and why the two-population GA works.**

I. Introduction: Context and Purposes

We describe in this paper a successful application of evolutionary computation to a difficult and commercially significant constrained optimization problem. Our purposes and reasons in presenting this case may be grouped into two kinds. First, we wish to place in the record of the Evolutionary Computation community a constrained optimization problem, arising from a Management Science context, that is representative of contemporary commercial problems, both in its formulation and in its interpretation and use. The model we have selected—the Financial Product (FP) model, described below—is a commercially leading-edge constrained optimization problem for assembling mortgage refinancing transactions. The state-of-the-art today has elementary decision aiding tools (simple exploratory calculations) in support of mortgage refinancing sales personnel. The tools in use are adequate but far from optimal. As we shall see, the underlying, implicit optimization problem is quite challenging. If Evolutionary Computation can, as we think it can, make a valuable and comparatively superior contribution to this problem, this will meaningfully enlarge the scope of practical optimization as well as the scope of application of evolutionary methods.

Our second cluster of reasons for presenting this case study is that we wish to explore the effectiveness of the two-population genetic algorithm (GA) on difficult, real-world constrained optimization problems. In this way of organizing an evolutionary process, two populations of solutions are maintained and evolved: the feasible population (consisting only of feasible solutions) and the infeasible population (consisting only of infeasible solutions). Genetic operations are applied to the two populations sequentially. Offspring are placed in one of the two populations, depending only on whether they are feasible or not. Thus, feasible parents may produce infeasible children and infeasible parents may produce feasible children. See the Appendix for the particular details of the two-population GA we use in this study.

The two-population GA for constrained optimization is motivated by the fact that, while evolution programs (EPs) in general and genetic algorithms in particular are optimum-seeking algorithms, feasibility constraints on optimization problems pose special difficulties for EPs and GAs. The fundamental problem is that the genetic operators are normally blind to feasibility considerations. Thus, e.g., mutation of a successful chromosome, or crossover between two highly fit parents may easily yield infeasible offspring. Various methods have been devised for dealing with this problem, including penalty functions, repair, discarding infeasible solutions, and special encoding. See [1] for a review. Success has been mixed. It is, however, evident from prior work [2], [3], [4] and from new results reported here, that the two-population GA has considerable merits, as measured by quality of solution found for computational effort (among other criteria). We wish to understand why this happens. To this end, there are three main issues we wish to address:

1) Charactristic properties. What are the characteristic properties of this search process?
2) Infeasible contributions. How does the infeasible population contribute to the search process?
3) Uses for decision making. Does the infeasible population contain information that is useful for decision making based on the model?

Our approach to handling these questions will be an empirical one, in which we follow the details of the evolutionary process under the two-population GA. We have been impressed by the yield from analogous studies of biological systems, e.g., [5], [6], [7], and we note that some of these scientists have turned to digital agents and computational systems in order to examine the microstructure of evolutionary processes, e.g. [8]. Let us see whether what works in nature works in Management

Science, too.

II. The Financial Product Model

The Financial Product (FP) model is designed to optimize mortgage refinancing from the perspective of the lender, who wishes to issue as large a loan as possible, given the customer's requests. The lender, however, also operates under a series of policy constraints arising from law and good business practice.

Here is the generalized form of a refinance mortgage product:

$$L = \sum_{i=1}^{n} x_i d_i + c + p + e + M \tag{1}$$

where:

- $L =$ loan amount
- $n =$ number of debts
- $x_i \in \{0, 1\}$ Debt payoff flags; 1 if a debt is paid off by the loan, 0 else
- $d_i \in R^+$ Debt balance
- $c \in R^+$ Cash out amount
- $p =$*constant* Dollar amount of points
- $e =$*constant* Fees
- $M = \sum_{t=1}^{\#Mortgages} y_t M_t, \quad M \in R^+$; $y =$requested payoff flag, 1 if a debt is paid off, 0 else; $M =$ mortgage balance

The genererlized form of the constrained optimization problem for the financial product—what we shall call the *Financial Product (FP) model*—is

$$\min z = \tag{2}$$

$$\sum_{i=1}^{n} \mid (p_{x_i} - x_i) \mid d_i + \mid \sum_{i=1}^{n} x_i d_i - \sum_{i=1}^{n} p_{x_i} d_i \mid + \frac{c' - c}{H} \tag{3}$$

Subject to:

$$L \leq A \cdot V_m - \sum_{t=1}^{\#Mortgage} (1 - y_t) M_t \tag{4}$$

$$\sum_{i=1}^{n} (1 - x_i) q_i + \frac{L \cdot R}{12} \leq D_m \cdot I - Q - M_P \tag{5}$$

$$c \leq c' \tag{6}$$

$$L \leq L_m \tag{7}$$

$$L_U \leq L \tag{8}$$

$$V < \sum_{j=1}^{m} \lambda_j \alpha_j \tag{9}$$

$$V \geq \sum_{j=1}^{m} \lambda_{j+1} \alpha_j \tag{10}$$

$$L < \sum_{j=1}^{r} \gamma_j \kappa_j \tag{11}$$

$$L \geq \sum_{j=1}^{r} \gamma_{j+1} \kappa_j \tag{12}$$

$$L < \sum_{s=1}^{l} \tau_s l_{d_{s+1}} \tag{13}$$

$$L \geq \sum_{s=1}^{l} \tau_s l_{d_s} \tag{14}$$

$$L < \sum_{s=1}^{z} \omega_s l_{v_{s+1}} \tag{15}$$

$$L \geq \sum_{s=1}^{z} \omega_s l_{v_s} \tag{16}$$

$$\sum_{j=1}^{m+1} \lambda_j = 1 \tag{17}$$

$$\sum_{j=1}^{r+1} \gamma_j = 1 \tag{18}$$

$$\sum_{s=1}^{l} \tau_s = 1 \tag{19}$$

$$\sum_{s=1}^{z} \omega_s = 1 \tag{20}$$

where:

$$L = \sum_{i=1}^{n} x_i d_i + c + p + e + M \tag{21}$$

$$D_m = \sum_{s=1}^{l} \tau_s D_{B_s} \tag{22}$$

$$V_m = \sum_{s=1}^{z} \omega_s V_{B_s} \tag{23}$$

$$V = \frac{L + \sum_{t=1}^{\#Mortages} (1 - y_t) M_t}{A} \tag{24}$$

$$R = \sum_{j=1}^{m+1} \sum_{k=1}^{r+1} \lambda_j \gamma_k R_{jk} \tag{25}$$

$$M_P = \sum_{t=1}^{\#Mortages} (1 - y_t) m_t \tag{26}$$

$$M = \sum_{t=1}^{\#Mortages} y_t M_t \tag{27}$$

with these domains for decision variables:

$x_i \in \{0, 1\}, \;\; c \in R^+, \;\; \lambda_j \in \{0, 1\}$

$\gamma_j \in \{0, 1\}, \;\; \tau_s \in \{0, 1\}, \;\; \omega_s \in \{0, 1\}$

The Parameters (constants): $d_i, q_i, \alpha_j, \kappa_j, R_{jk}, D_{B_s}, V_{B_s}, l_{d_s}, l_{v_s}, m_t, M_t, c', A, H, I, p, e, Q, L_m, L_U,$ all $\in R^+$. p_{x_i} and $y_t \in \{0, 1\}$

Points arising:

1) Not all decision variables appear in the objective function.
2) See the objective function, expression (3). The goal is to come as close as possible to meeting the client's preferences: $p_{x_i} \in R^+$ Preferred debt payoff flags. $c' \in R^+$ Preferred cash out amount. $H \in R^+$ is the weight on the cash out.
3) Expression (4) is the Loan-to-Value constraint.
   $A \in R^+$ Appraised value of the property
   $V_m \in R^+$ Maximum loan-to-value possible relative to the loan amount
4) Expression (5) is the Debt-to-Income constraint
   $q_i \in R^+$ Monthly payment for d_i
   $R \in R^+$ Rate
   $V = \frac{L+\sum_{t=1}^{\#Mortgages}(1-y_t)M_t}{A}$ Loan to value ratio
   $D_m \in R^+$ Maximum debt-to-income possible relative to the loan amount
   $I \in R^+$ Monthly income. $Q \in R^+$ Other monthly liabilities. $M_P = \sum (1-y_t)\, m_t$ Total monthly payment of mortgages not refinanced.
5) Expression (6) is the Cash-Out Constraint. Cash-Out, c, cannot be larger than cash requested by the client, c'.
6) Expression (7) is the Maximum Loan Amount constraint. $L_m \in R^+$ is the maximum loan amount possible.
7) Expression (8) is the Minimum Loan Amount contraint. $L_U \in R^+$.

Further points follow.

A. Points and Rate Structure

When constructing a refinance mortgage product, there is the concept of "buying up" or "buying down" the rate by decreasing or increasing the number of points for the mortgage. The dollar value of a point is 1% of the loan amount and is included in the loan amount. That points are included in the loan amount causes a circularity issue when calculating the loan amount. Below we discuss a technique to closely estimate the appropriate loan amount giving this circularity.

The rate for each mortgage is derived from a pricing model that includes a "base points" structure. This model assigns an initial rate, R, given the points amount, p. Any change to the base points requires recalculating the rate.

B. Determining the Rate

The initial rate, R, is a function of loan amount L and loan-to-value ratio V. In this model, the loan amount and V are divided into brackets, and depending on where the actual values fall, R is assigned. The brackets are divided according to risk, with high loan amount and V values assigned higher rates because of higher risk, and vice versa. The mathematical representation of this submodel is as follows. This formulation forces λ_j to be 1 when V is between α_j and α_{j+1}, and forces γ_j to be 1 when L is between κ_j andκ_{j+1}.

$$R = \sum_{j=1}^{m+1}\sum_{k=1}^{r+1} \lambda_j \gamma_k R_{jk} \tag{28}$$

$$\sum_{j=1}^{m+1} \lambda_j = 1 \tag{29}$$

$$\sum_{j=1}^{r+1} \gamma_j = 1 \tag{30}$$

$$V < \sum_{j=1}^{m} \lambda_j \alpha_j \tag{31}$$

$$V \geq \sum_{j=1}^{m} \lambda_{j+1} \alpha_j \tag{32}$$

$$L < \sum_{j=1}^{r} \gamma_j \kappa_j \tag{33}$$

$$L \geq \sum_{j=1}^{r} \gamma_{j+1} \kappa_j \tag{34}$$

***

This is the general form of the FP model. We have studied a number instances of it. In what follows we report on a particular instance, called Soofi, which is described in the Appendix. We do this for the sake of concreteness and specificity. The instance Soofi is representative of customer cases; it has 42 binary and one floating point decision variable. Soofi has 16 existing debts (d_is) and is seeking to payoff a mortgage balance, M of $100,000. The problem is to find a feasible loan amount, L, that comes as close as possible to paying off the existing debts (as requested by the client) and giving the cash out requested by the client, all as weighted in importance by the objective function.

III. Existing Solution Methods

Existing tools for this class of product optimization problems (FP is a member of that class) are based on attempts to automate the strategies employed by subject matter experts and do not involve any attempts at classical or modern optimization. For example, if a product violates the debt-to-income constraint, the subject matter expert will take one of three different strategies:

1) Pay off more debt with rates higher than the new mortgage rate, by reducing cash out.
2) Pay off more debt with rates higher than the new mortgage rate, by increasing the loan amount.
3) Reduce the loan amount by reducing cash out.

This strategy does not guarantee optimality. Typical real-world scenarios are complex enough to not guarantee finding a feasible solution, even if one exists.

There is also a private, in-house calculation tool, called IDSS, for heuristically finding solutions to the Financial Product model. These solutions have generally been deemed satisfactory, but the question remains of whether an optimizer

TABLE I

Generations	Infeasibility	InF→Fea	InF-med z	N-Fea	$\sigma_{z_{\mathrm{InF}}}$	SumSlacks
0–99	-193174.17	0.00	91631.17	24	22563.94	1974883.03
900–999	-98969.10	0.00	71902.79	30	19838.45	1930736.77
1900–1999	-132317.55	0.00	74113.44	0	22369.97	2125371.73
2900–2999	-122013.11	0.00	67507.97	0	21298.14	1968684.10
3900–3999	-99153.61	0.00	71797.76	0	19535.74	1923273.02
4900–4999	-97213.60	0.00	82552.62	0	22488.83	1904357.05
5900–5999	-36049.56	0.35	62673.89	1679	10710.91	1854298.65
6900–6999	-41520.62	0.47	59147.20	1486	12518.97	1859558.95
7710–7809	-41131.82	0.61	62276.57	1401	11598.10	1862406.93

Summary of Infeasible Results. Infeasibility=$-1\cdot$sum of absolute violations of constraints (averaged over each solution for 100 generations). InF→Fea=number of feasible offspring from the infeasible population (by generation, averaged over 100 generations). InF-med z=median objective function value in the infeasible population (by generation, averaged over 100 generations). N-Fea: Count of infeasible solutions with infeasibility ≥ -200.0, totaled over 100 generations. $\sigma_{z_{\mathrm{InF}}}$=standard deviation of objective function values in the infeasible population, averaged over 100 generations. SumSlacks=total slack in non-binding constraints in the infeasible population, averaged over each solution for 100 generations.

could do better. IDSS's best solution to the Soofi instance of the problem is at about $z^+ = 90,000$ (z^+ is our notation for the objective function value of the best discovered feasible solution. In the special case that global optimality can be ascertained, z^* is used instead.)

IV. Attempted Solutions

The problem as formulated in the Financial Product model is a difficult non-linear IP (integer program). As a first pass at solving it, we replaced expression (25) for R with an estimated constant, R_c. This *relaxed* version of the problem is now a linear IP and can be solved by CPLEX. We modeled the relaxed problem in GAMS and obtained a solution via CPLEX. At optimality the objective function value to the relaxed Soofi problem is $z^* = 66281.987710$. The associated solution, however, is infeasible in the original problem. GENOCOP IV was unable to find an initial feasible solution to the original (unrelaxed) problem, even when primed with the CPLEX solution to the relaxed problem. Further, GENOCOP IV was unable to proceed when primed with a feasible solution discovered by the two-population GA (below). We note that GENOCOP III is capable of handling non-linear constraints and objective functions. Unfortunately it is not able to handle integer / binary variables. Since we had 42 integer variables, relaxing it into a non-linear form, without binary/integer, is of little value.

V. Solution with the Two-Population GA

We sought solutions to the Financial Product model using an untuned two-population GA. That is, we used standard settings from previous experiments (e.g., population sizes of 50, 10,000 total number of generations; see [2], [3], [4] and the Appendix to this paper) and made no attempt to optimize them. Typically, and specifically in the Soofi instance, the two-population GA had difficulty finding feasible solutions. Using the integers between 1 and 1000 as different random seeds, the two-population GA found feasible solutions to the Soofi model instance in about 800 of the 1000 cases (again, this is using the standard settings). In most and probably all of these cases, no feasible solution was found at initialization, so that generations of infeasible solutions evolved until a feasible offspring appeared. After that event, elite selection (copying the best of generation solution to the next generation) guaranteed continued feasible populations. The best solution found among the 1000 runs with random seeds 1–1000 had an objective function value of $z^+ = 66478.98479$. This solution and most of the other best solutions found (in a given run) were significantly better than the solutions obtained by IDSS, the existing heuristic for the problem. Recall that the IDSS z^+ for Soofi was about 90,000 and that we are minimizing. Comparing IDSS's solutions with solutions produced by the two-population GA validated the GA at the expense of IDSS. It was found that IDSS had improperly failed to pay off debts and consequently paid out too much cash. The IDSS solutions, however, were feasible.

TABLE II

Generations	z^+	Fea→InF	$\sigma_{z_{\mathrm{Fea}}}$	SumSlacks
0–99	83015.04	15.18	8858.87	1806409.66
400–499	66570.98	14.09	1372.97	1806396.98
900–999	66570.98	14.04	1443.59	1806415.74
1400–1499	66570.98	14.53	1315.93	1806379.89
2091–2190	66482.98	14.19	1295.44	1806183.39

Summary of Feasible Results. Generation i feasible follows generation $5619+i$ infeasible and precedes generation $5620+i$ infeasible. z^+=best objective value found in generation, averaged over 100 generations. Fea→InF=number of infeasible solutions created, averaged over 100 generations. $\sigma_{z_{\mathrm{Fea}}}$=standard deviation of objective function values in the feasible population, averaged over 100 generations. SumSlacks=total slack in non-binding constraints, averaged over each solution for 100 generations.

We describe now the details of a single run of the two-population GA on the Soofi problem. The objective function value of the best (feasible) solution found on this run, z^+, was 66482.9866098969. See the Appendix for details of the solution. Although not the best solution found, it is very close ($66482.9866098969/66478.98479 = 1.000060$ or within 0.006%) and it is typical; very many best of run solutions were in this neighborhood. We note as well that this solution is near to (0.3% away) the solution to the relaxed problem, as found by CPLEX ($66482.986609/66281.987710 = 1.0030$). Thus, with the two-population GA we pay very little penalty (if any) for departure from global optimality and we actually get a feasible solution.

In this typical run, the algorithm had difficulty finding a feasible solution. It did not find one at initialization (after 5,000 attempts), but proceeded to evolve the infeasible population. At generation 5619 of the infeasible population, a single feasible solution appeared and migrated to the feasible population. There followed 2191 generations of feasible evolution (generations 0–2190) and 2190 generations of infeasible evolution—now alternating—for a total of 10,001 generations evaluated (including the initial infeasible generation). All in all, the infeasible population created 1174 feasible solutions, averaging 0.53 feasible solution per generation beginning in 5619. The maximum emigration from the infeasible population was 7, in generation 7183. The average emigration per generation from the feasible to the infeasible population was 14.477.

A solution with the z^+ value appears for the first time in generation 1534 of the feasible population. Three copies are present, with $x_1 = 13.3901031383201$. This allele first appears in the feasible population in generation 341 (following generation 5960 of the infeasible population) and remains present to the end of the run. The infeasible population first sees this allele two generations later, at 5962.

In the solution associated with z^+, we find the pattern of loans paid off: $x_1 = x_2 = x_3 = x_4 = x_5 = 1,\ x_6 = 0$ (again, see the Appendix for details of the solution at z^+). This pattern appears in the *infeasible* population 4,545 times *before* the time of the first feasible solution, following generation 5619. Adding to the z^+ pattern, $x_7 = 1,\ x_8 = 1,\ x_9 = 0,\ x_{10} = 1$, there are 60 occurrences in the infeasible population before generation 5619. This extended pattern, however, does not appear in the feasible population until generation 137. It promptly disappears and returns in generation 146, drops out after generation 147, and comes and goes until generation 1534 where it appears in the z^+ solution, now maintained for the remainder of the run. Its previous appearance in the feasible population was in generation 1121. The pattern appears six times in the infeasible population in the 100 generations preceding the feasible generation 1534, including once in the generation immediately preceding.

Evidently, the infeasible population is discovering building blocks—patterns of settings for the decision variables—that are useful in the feasible population in the sense of being associated with high fitness solutions. Further, after the feasible population has discovered a useful configuration, that pattern tends to appear and be maintained in the infeasible population. In the present case, every generation of the infeasible population (of 50 solutions) beginning with 7161 has more than 25 solutions with this extended pattern (settings of x_1–x_{10}). This is, we find, entirely typical and representative behavior of the two-population GA on constrained optimization problems.

Tables I and II contain summary information from the run on the infeasible and feasible populations respectively. It is instructive to compare them. First, Table I and the infeasible population. Infeasibility (column 2) is the sum of the constraint violations times -1, so that the closer to 0 a solution is on infeasibility the more nearly feasible it is. We observe a sizable and nearly monotonic improvement towards feasibility as the run proceeds. Between the beginning of the run and somewhat before the first feasible solution is found the infeasibility declines by roughly half. After the feasible population is first created and until the end of the run, the infeasibility again drops in half. This indicates that selection is working to reduce infeasibility even without immigration from the feasible population. The data suggest, moreover, that that process speeds up the movement. It took approximately 4,000 generations without immigration to halve the infeasibility, but only 1,000 generations (initially) to do so with immigration from the feasible population. This indicates that recent arrivals to the infeasible population are on average less infeasible than the incumbents. Thus, the two-population GA is homing in on and exploring the feasible–infeasible boundary and doing so from both sides simultaneously.

Column 3 of Table I, labeled InF→Fea, presents the average emigration to feasibility per generation during the periods indicated. The out-migration rate—the discovery of feasible solutions by the infeasible population—is small but increasing over time. We find this a typical result. This indicates, especially when combined with the infeasibility information, that the infeasible population is congregating closer and closer to the feasible-infeasible boundary and thus becoming more able to find feasible solutions.

Column 4 of Table I, labeled InF-med z, shows the median objective value for infeasible solutions per generation, averaged over the indicated period. Remarkably, and consistently with other runs and other models, there is a clear and steady improvement in this value *even though it is not an object of selection.*

Column 5, labeled N-Fea, shows the total number of nearly feasible solutions (Infeasibility ≥ -200.0, a close call for this problem) during the indicated period. These data reinforce the picture of the infeasible population exploring the nearly feasible region, close to the feasible-infeasible boundary. It is instructive in this regard to view Table III. In the 100 infeasible generations before the discovery of the first feasible solution there were 38 nearly feasible solutions. In the 100 generations before that, there were none. And in the 100 generations after there were 1,239.

Column 6, labeled $\sigma_{z_{\mathrm{InF}}}$, shows the standard deviation of the objective function values during the indicated periods. Notice that it is nearly constant before the feasible population is created and is nearly constant at half the level after the feasible population is created. This indicates that the infeasible population is exploring the feasible-infeasible boundary rather broadly before the feasible population is created (recall: selection is on infeasibility, not on objective function values), but that after the feasible population is discovered immigration to the infeasible population begins and is focused on narrower regions of the feasible-infeasible boundary, i.e., those near to the feasible solutions.

Finally, SumSlacks shows the total slack in solution constraints per solution, averaged over the indicated period. This is a measure of how much *unused* resources are associated with

TABLE III

Generations	N-Fea
5420–5519	0
5520–5619	38
5620–5719	1239

Count of Near-Feasible Solutions (≥ -200.0).

the solutions. The measure is a crude one, because the various constraints may be (and are) stated in different dimensions and scales. Even so, it is remarkable that this figure in the infeasible population moves towards and eventually approximates the corresponding number in the feasible population. See Table II.

Considering Table II now, the column labeled z^+ shows the objective function value of the best solution per generation, averaged over the indicated generations. Early on, by the 400–499 period, the feasible population has found at least one very good solution, at $z = 66570.98$. It is unable to find a better one until much later, at generation 1534 as we saw. (This is a difficult optimization problem.) Notice that the feasible population generates a rather high level of out migration throughout the run (column Fea→InF) and that the variance in the feasible population (column $\sigma_{z_{\mathrm{Fea}}}$) stabilizes, apparently avoiding over-convergence. Recall in this regard the trickle of in-migration from the infeasible population.

VI. Discussion & Conclusion

The Financial Product (FP) model and its Soofi instance constitute a realistic, and very challenging, optimization problem, for which standard solvers, including GENOCOP, leave much to be desired. The untuned two-population GA described here has been remarkably successful in finding good, feasible and nearly optimal solutions. Recalling our three motivating questions—

1) Charactristic properties. What are the characteristic properties of this search process?
2) Infeasible contributions. How does the infeasible population contribute to the search process?
3) Uses for decision making. Does the infeasible population contain information that is useful for decision making based on the model?

—we can summarize our findings as follows. The previous section was mainly devoted to answering the first and second questions. The picture that emerges is that selection drives both the feasible and the infeasible populations towards the boundary between the feasible and the infeasible region. That boundary may be discontinuous and even disconnected; the GA is not fundamentally affected by this. As selection proceeds, the solutions—feasible and infeasible—increasingly hover near the boundary. The feasible population is directed towards improvements in the objective function and eventually collides with the boundary. The infeasible population is directed by selection towards the boundary, regardless of objective function values. Over time, the two populations approach one another, to the benefit of solving the problem.

We conclude with a comment on the third question. It was remarked above that there are a fair number of nearly feasible solutions and there is substantial variance among the feasible solutions. This body of solutions is potentially of great value for practical decision making. As explored previously in the context of a conventional single-population GA [9], [10], [11], alternate solutions can be used to estimate shadow prices and reduced costs, quantities that are normally only available for linear programming problems. In particular, the feasible population can be used to answer the following question (and abstractions of it): In the z^+ solution, debt 6, at d_6, is not paid off; what is the best solution available to me if I decide to pay it off with this loan and how much will the objective function value increase? The answer to this question, which may be estimated from the trace of the run, is called the *reduced cost* of the decision variable, here x_6. Also, the infeasible population can be used to answer the following question: If I am able to relax a particular constraint by a certain amount, how much better can I do in the objective function, and what is the associated solution? The answer to this question is called the *shadow price* on the constraint.

The reader will no doubt be able to think of questions useful for decision making that may be answered from the trace of two-population GA runs on a constrained optimization problem. This is a subject to be explored in detail in the future, and in a longer paper, but it is safe to say that the two-population GA shows much promise for practical application.

Appendix
Specifics of the 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 [4]. 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 formula [1, pages 103, 111]) 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 for 5,000 generations, or rather 10,000 generations in total for the two populations.

THE MODEL INSTANCE: SOOFI

Solution for Soofi, run 260: Random number seed, 260; $z^+ = 66482.98660989686$. Decision variable settings at this solution: $c = 13.390103138320057$, $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$, $x_6 = 0$, $x_7 = 1$, $x_8 = 1$, $x_9 = 0$, $x_{10} = 1$, $x_{11} = 1$, $x_{12} = 0$, $x_{13} = 0$, $x_{14} = 1$, $x_{15} = 0$, $x_{16} = 0$, $\lambda_1 = 0$, $\lambda_2 = 0$, $\lambda_3 = 0$, $\lambda_4 = 0$, $\lambda_5 = 0$, $\lambda_6 = 0$, $\lambda_7 = 0$, $\lambda_8 = 0$, $\lambda_9 = 1$, $\lambda_{10} = 0$, $\gamma_1 = 0$, $\gamma_2 = 0$, $\gamma_3 = 0$, $\gamma_4 = 0$, $\gamma_5 = 1$, $\gamma_6 = 0$, $\gamma_7 = 0$, $\gamma_8 = 0$, $\gamma_9 = 0$, $\gamma_10 = 0$, $\tau_1 = 1$, $\tau_2 = 0$, $\tau_3 = 0$, $\omega_1 = 1$, $\omega_2 = 0$, $\omega_3 = 0$.

This parameterized optimization problem, called *Soofi* is shown below.

Objective function (Minimization): $\min z = 100(1 - x_1) + 512(1 - x_2) + 1022(1 - x_3) + 2563(1 - x_4) + 3567(1 - x_5) + 3267(1 - x_6) + 4562(1 - x_7) + 6532(1 - x_8) + 8569(1 - x_9) + 4562(1 - x_{10}) + 9856(1 - x_{11}) + 256(1 - x_{12}) + 15632(1 - x_{13}) + 4562(1 - x_{14}) + 236(1 - x_{15}) + 5269(1 - x_{16}) + 100 + 512 + 1022 + 2563 + 3567 + 3267 + 4562 + 6532 + 8569 + 4562 + 9856 + 256 + 15632 + 4562 + 236 + 5269 - (100x_1 + 512x_2 + 1022x_3 + 2563x_4 + 3567x_5 + 3267x_6 + 4562x_7 + 6532x_8 + 8569x_9 + 4562x_{10} + 9856x_{11} + 256x_{12} + 15632x_{13} + 4562x_{14} + 236x_{15} + 5269x_{16}) + \frac{25000 - c}{1000}$

Constraints:

$$L \leq 150000(0.95\omega_1 + 0.85\omega_2 + 0.70\omega_3) \tag{35}$$

$$\begin{aligned}\frac{L \cdot R}{12} &+ 25(1 - x_1) + 86(1 - x_2) + 51(1 - x_3) \\ &+ 102(1 - x_4) + 125(1 - x_5) + 253(1 - x_6) \\ &+ 126(1 - x_7) + 256(1 - x_8) + 356(1 - x_9) \\ &+ 562(1 - x_{10}) + 989(1 - x_{11}) + 28(1 - x_{12}) \\ &+ 563(1 - x_{13}) + 111(1 - x_{14}) + 236(1 - x_{15}) \\ &+ 366(1 - x_{16}) \leq 7500(0.5\tau_1 + 0.5\tau_2 + 0.5\tau_3)\end{aligned} \tag{36}$$

$$15000 \leq L \leq 1000000 \tag{37}$$

$$\begin{aligned}\frac{L}{150000} &\leq 0.60\lambda_1 + 0.65\lambda_2 + 0.70\lambda_3 + 0.75\lambda_4 \\ &+ 0.80\lambda_5 + 0.85\lambda_6 + 0.90\lambda_7 + 0.95\lambda_8 + 1.00\lambda_9\end{aligned} \tag{38}$$

$$\begin{aligned}\frac{L}{150000} &\geq 0.60\lambda_2 + 0.65\lambda_3 + 0.70\lambda_4 + 0.75\lambda_5 \\ &+ 0.80\lambda_6 + 0.85\lambda_7 + 0.90\lambda_8 + 0.95\lambda_9 + 1.00\lambda_{10}\end{aligned} \tag{39}$$

$$\begin{aligned}L &\leq 25000\gamma_1 + 50000\gamma_2 + 75000\gamma_3 + 100000\gamma_4 \\ &+ 150000\gamma_5 + 400000\gamma_6 + 500000\gamma_7 \\ &+ 750000\gamma_8 + 1000000\gamma_9\end{aligned} \tag{40}$$

$$\begin{aligned}L &\geq 25000\gamma_2 + 50000\gamma_3 + 75000\gamma_4 + 100000\gamma_5 \\ &+ 150000\gamma_6 + 400000\gamma_7 + 500000\gamma_8 \\ &+ 750000\gamma_9 + 1000000\gamma_{10}\end{aligned} \tag{41}$$

$$L \leq 400000\tau_1 + 500000\tau_2 + 1000000\tau_3 \tag{42}$$

$$L \geq 15000\tau_1 + 400000\tau_2 + 500000\tau_3 \tag{43}$$

$$L \leq 400000\omega_1 + 500000\omega_2 + 1000000\omega_3 \tag{44}$$

$$L \geq 15000\omega_1 + 400000\omega_2 + 500000\omega_3 \tag{45}$$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 + \lambda_8 + \lambda_9 + \lambda_{10} \leq 1 \tag{46}$$

$$\gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 + \gamma_5 + \gamma_6 + \gamma_7 + \gamma_8 + \gamma_9 + \gamma_{10} \leq 1 \tag{47}$$

$$\tau_1 + \tau_2 + \tau_3 \leq 1 \tag{48}$$

$$\omega_1 + \omega_2 + \omega_3 \leq 1 \tag{49}$$

$$c \leq 25000 \tag{50}$$

where

$$\begin{aligned}L =&100x_1 + 512x_2 + 1022x_3 + 2563x_4 + 3567x_5 + 3267x_6 \\ &+ 4562x_7 + 6532x_8 + 8569x_9 \\ &+ 4562x_{10} + 9856x_{11} + 256x_{12} \\ &+ 15632x_{13} + \\ &4562x_{14} + 236x_{15} + 5269x_{16} + 2137.50 + 2484.17 \\ &+ 100000 + c\end{aligned} \tag{51}$$

$$
\begin{aligned}
R = \lambda_1\gamma_1 * 0.07125 + \lambda_2\gamma_1 * 0.07125 + \lambda_3\gamma_1 * 0.07125 + \\
\lambda_4\gamma_1 * 0.07375 + \lambda_5\gamma_1 * 0.07625 + \lambda_6\gamma_1 * 0.07875 + \\
\lambda_7\gamma_1 * 0.08125 + \lambda_8\gamma_1 * 0.08625 + \lambda_9\gamma_1 * 0.09125 + \\
\lambda_{10}\gamma_1 * 0.09125 + \lambda_1\gamma_2 * 0.07125 + \lambda_2\gamma_2 * 0.07125 + \\
\lambda_3\gamma_2 * 0.07125 + \lambda_4\gamma_2 * 0.07375 + \lambda_5\gamma_2 * 0.07625 + \\
\lambda_6\gamma_2 * 0.07875 + \lambda_7\gamma_2 * 0.08125 + \lambda_8\gamma_2 * 0.08625 + \\
\lambda_9\gamma_2 * 0.09125 + \lambda_{10}\gamma_2 * 0.09125 + \lambda_1\gamma_3 * 0.06875 + \\
\lambda_2\gamma_3 * 0.06875 + \lambda_3\gamma_3 * 0.06875 + \lambda_4\gamma_3 * 0.07125 + \\
\lambda_5\gamma_3 * 0.07375 + \lambda_6\gamma_3 * 0.07625 + \lambda_7\gamma_3 * 0.07875 + \\
\lambda_8\gamma_3 * 0.08375 + \lambda_9\gamma_3 * 0.08875 + \lambda_{10}\gamma_3 * 0.08875 + \\
\lambda_1\gamma_4 * 0.06625 + \lambda_2\gamma_4 * 0.06625 + \lambda_3\gamma_4 * 0.06625 + \\
\lambda_4\gamma_4 * 0.06875 + \lambda_5\gamma_4 * 0.07125 + \lambda_6\gamma_4 * 0.07375 + \\
\lambda_7\gamma_4 * 0.07625 + \lambda_8\gamma_4 * 0.08125 + \lambda_9\gamma_4 * 0.08625 + \\
\lambda_{10}\gamma_4 * 0.08625 + \lambda_1\gamma_5 * 0.06375 + \lambda_2\gamma_5 * 0.06375 + \\
\lambda_3\gamma_5 * 0.06375 + \lambda_4\gamma_5 * 0.06625 + \lambda_5\gamma_5 * 0.06875 + \\
\lambda_6\gamma_5 * 0.07125 + \lambda_7\gamma_5 * 0.07375 + \lambda_8\gamma_5 * 0.07875 + \\
\lambda_9\gamma_5 * 0.07875 + \lambda_{10}\gamma_5 * 0.08375 + \lambda_1\gamma_6 * 0.06125 + \\
\lambda_2\gamma_6 * 0.06125 + \lambda_3\gamma_6 * 0.06125 + \lambda_4\gamma_6 * 0.06375 + \\
\lambda_5\gamma_6 * 0.06625 + \lambda_6\gamma_6 * 0.06875 + \lambda_7\gamma_6 * 0.07125 + \\
\lambda_8\gamma_6 * 0.07625 + \lambda_9\gamma_6 * 0.08125 + \lambda_{10}\gamma_6 * 0.08125 + \\
\lambda_1\gamma_7 * 0.06875 + \lambda_2\gamma_7 * 0.06875 + \lambda_3\gamma_7 * 0.06875 + \\
\lambda_4\gamma_7 * 0.07125 + \lambda_5\gamma_7 * 0.07375 + \lambda_6\gamma_7 * 0.07625 + \\
\lambda_7\gamma_7 * 0.07875 + \lambda_8\gamma_7 * 0.08375 + \lambda_9\gamma_7 * 0.08875 + \\
\lambda_{10}\gamma_7 * 0.08875 + \lambda_1\gamma_8 * 0.07125 + \lambda_2\gamma_8 * 0.07125 + \\
\lambda_3\gamma_8 * 0.07125 + \lambda_4\gamma_8 * 0.07375 + \lambda_5\gamma_8 * 0.07625 + \\
\lambda_6\gamma_8 * 0.07875 + \lambda_7\gamma_8 * 0.08125 + \lambda_8\gamma_8 * 0.08625 + \\
\lambda_9\gamma_8 * 0.09125 + \lambda_{10}\gamma_8 * 0.09125 + \lambda_1\gamma_9 * 0.07125 + \\
\lambda_2\gamma_9 * 0.07125 + \lambda_3\gamma_9 * 0.07125 + \lambda_4\gamma_9 * 0.07375 + \\
\lambda_5\gamma_9 * 0.07625 + \lambda_6\gamma_9 * 0.07875 + \lambda_7\gamma_9 * 0.08125 + \\
\lambda_8\gamma_9 * 0.08625 + \lambda_9\gamma_9 * 0.09125 + \lambda_{10}\gamma_9 * 0.09125 + \\
\lambda_1\gamma_{10} * 0.07125 + \lambda_2\gamma_{10} * 0.07125 + \lambda_3\gamma_{10} * 0.07125 + \\
\lambda_4\gamma_{10} * 0.07375 + \lambda_5\gamma_{10} * 0.07625 + \lambda_6\gamma_{10} * 0.07875 + \\
\lambda_7\gamma_{10} * 0.08125 + \lambda_8\gamma_{10} * 0.08625 + \lambda_9\gamma_{10} * 0.09125 + \\
\lambda_{10}\gamma_{10} * 0.09125
\end{aligned}
\tag{52}
$$

Variable bounds

$0 \leq c \leq 25000$

$x_i \in \{0, 1\}, i = 1, 2, ..., 16$

$\lambda_j \in \{0, 1\}, \gamma_j \in \{0, 1\}, j = 1, 2, ..., 10$

$\tau_s \in \{0, 1\}, \omega_s \in \{0, 1\}, s = 1, 2, ..., 3$

M_t and $m_t \in R^+$

Parameter Values

$d_i = \{100, 512, 1022, 2563, 3567, 3267, 4562, 6532, 8569, 4562, 9856, 256, 15632, 4562, 236, 5269\}$

$q_i = \{25, 86, 51, 102, 125, 253, 126, 256, 356, 562, 989, 28, 563, 111, 236, 366\}$

$\alpha_j = \{0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 1.00\}$

$\kappa_j = \{25000, 50000, 75000, 100000, 150000, 400000, 500000, 750000, 1000000\}$

$R_{jk} = \{0.07125, ..., ...0.09125\}$, $D_{B_t} = \{0.5, 0.5, 0.5\}$

$V_{B_t} = \{0.70, 0.85, 0.95\}$, $l_d = \{15000, 400000, 500000, 1000000\}$

$l_v = \{15000, 400000, 500000, 1000000\}$

$M_t = 100000$, $m_t = 898$, $c' = 25000$, $A = 150000$ $H = 1000$, $I = 7500$, $p = 2137.50$, $e = 2484.17$, $Q = 0$, $L_m = 1000000$, $L_U = 15000$, $p_{x_i} = 1$, $y_t = 1$

REFERENCES

[1] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, Germany, third edition, 1996.

[2] Steven O. Kimbrough, Ming Lu, and David H. Wood. Exploring the evolutionary details of a two-population genetic algorithm in the context of constrained optimization. Working paper, University of Pennsylvania, Department of Operations and Information Management, Philadelphia, PA, January 2004.

[3] Steven O. Kimbrough, Ming Lu, David Harlan Wood, and D. J. Wu. Exploring a two-market genetic algorithm. In W. B. Langdon, E. Cantú-Paz, and et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 415–21, San Francisco, CA, 2002. Morgan Kaufmann Publishers.

[4] Steven O. Kimbrough, Ming Lu, David Harlan Wood, and D. J. Wu. Exploring a two-population genetic algorithm. In Erick Cantú-Paz and et al., editors, *Genetic and Evolutionary Computation—GECCO 2003*, LNCS 2723, pages 1148–1159, Berlin, Germany, 2003. Springer.

[5] T. F. Cooper, D. E. Rozen, and R. E. Lenski. Parallel changes in gene expression after 20,000 generations of evolution in *e. coli*. *Proceedings of the National Academy of Sciences*, 100:1072–1077, 2003.

[6] F Taddei, M. Radman, J. Maynard-Smith, B. Toupance, P. H. Gouyon, and B. Goodelle. Role of mutator alleles in adaptive evolution. *Nature*, 387(6634):700–703, 1997. Cited by Paul Sniegowski.

[7] R. E. Lenski, J. A. Mongold, P. D. Sniegowski, M Travisano, F. Vasi F, P. J. Gerrish, and T. M. Schmidt. Evolution of competitive fitness in experimental populations of e. coli: what makes one genotype a better competitor than another? *Antonie van Leeuwenhoek*, 73(1):35–47, 1998. A review.

[8] C. O. Wilke, J. Wang, C. Ofria, R. E. Lenski, , and C. Adami. Evolution of digital organisms at high mutation rate leads to survival of the flattest. *Nature*, 412:331–333, 2003.

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

[10] S. O. Kimbrough 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.

[11] B. Branley, 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.