Exploring a Two-Population Genetic Algorithm

Steven Orla Kimbrough[1], Ming Lu[1], David Harlan Wood[2], and D.J. Wu[3]

[1] University of Pennsylvania, 3730 Walnut Street, Philadelphia, PA 19104-6340
{kimbrough,milu}@wharton.upenn.edu
[2] University of Delaware, CIS Dept., Newark, DE 19716
wood@cis.udel.edu
[3] DuPree College of Management, Georgia Institute of Technology, Atlanta, GA
30332
wudj@drexel.edu

Abstract. In a two-market genetic algorithm applied to a constrained optimization problem, two 'markets' are maintained. One market establishes fitness in terms of the objective function only; the other market measures fitness in terms of the problem constraints only. Previous work on knapsack problems has shown promise for the two-market approach. In this paper we: (1) extend the investigation of two-market GAs to nonlinear optimization, (2) introduce a new, two-population variant on the two-market idea, and (3) report on experiments with the two-population, two-market GA that help explain how and why it works.

1 Introduction

Evolution programs (EPs), and more specifically genetic algorithms (GAs), are optimizing or optimum–seeking procedures. They are used routinely to maximize or minimize an objective function of a number of decision variables. Because the EPs/GAs are general–purpose procedures (aka: weak methods) they can be used in the computationally more challenging cases in which the objective function is nonlinear and/or the decision variables are integers or mixtures of integers and reals. However, when the objective function is constrained by one or more constraints use of GAs (and EPs) is problematic. Considerable attention has been paid to the subject (see [4,9,10,12,16] for excellent reviews and treatments), but no consensus approach has emerged for encoding constrained optimization problems as GAs.

A natural—and the most often used—approach is to penalize the objective function in proportion to the size of the constraint violations. A feasible solution will impose no penalty on the objective function, an infeasible solution will impose an objective function penalty (negative if maximizing, positive if minimizing) 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.

$$\max_{x_i} z(\boldsymbol{x}), subject\ to\ E(\boldsymbol{x}) \geq \boldsymbol{a}, F(\boldsymbol{x}) \leq \boldsymbol{b}, G(\boldsymbol{x}) = \boldsymbol{c}, x_i \in \mathcal{S}_i \qquad (1)$$

Here, $z(\boldsymbol{x})$ is the objective function value produced by the candidate solution $\boldsymbol{x}$.[1] E, F and G each yield zero or more constraint inequalities or equalities. Taken as functions, in the general case z, E, F, G can be any functions at all (on $\boldsymbol{x}$) and in particular need not be linear. $\mathcal{S}_i$ is the set of permitted values for the $\boldsymbol{x_i}$ (the components of the vector $\boldsymbol{x}$), and are often called the *decision variables* for the problem. $\mathcal{S}_i$ may include reals, integers, or mixtures. Problems of the form of expression (1) are not directly translatable into the linear encodings normally used for 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 a GA. To indicate a penalty function representation, let $\boldsymbol{x}$ be a (candidate) solution to a maximization constrained optimization problem. Its absolute fitness, $W(\boldsymbol{x})$, in the presence of penalties for constraint violation is measured as:

$$\max_{x_i} W(\boldsymbol{x}) = z(\boldsymbol{x}) - P(\boldsymbol{x}) \tag{2}$$

where $P(\boldsymbol{x})$ is the total penalty (if any) associated with constraint violations by $\boldsymbol{x}$. Problems representable as in expression (2) are directly and naturally encoded as GAs. If a GA finds a solution $\boldsymbol{x}$ that is feasible ($P(\boldsymbol{x}) = 0$) and has a high value for $W(\boldsymbol{x})$, then we may congratulate ourselves on the successful application of the GA heuristic.

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 GA population. This may be undesirable and it may be responsible for the generally recognized weak performance of GAs for constrained optimization problems.

Kimbrough et al. [8] have noted that in constrained optimization a GA will drive the population of solutions to the neighborhood of the Pareto frontier.[2] Under a penalty regime, GA-driven probing of the Pareto frontier is inhibited because infeasible solutions are heavily penalized, leading to likely loss of useful aspects (properties, genetic material) that placed them near the frontier. Armed with this intuition, Kimbrough et al. [8] investigated a "two-market GA" in which a single population of solutions to a constrained optimization problem undergoes two distinct GA procedures. Recalling expression (2), in phase 1("optimality improvement") the population is evaluated with the fitness function $\max_{x_i} z(\boldsymbol{x})$ and a new generation is created via the usual GA mechanisms. Then, in phase 2

---

[1] By *candidate solution* or just *solution* we mean any instance of $\boldsymbol{x}$. Some candidate solutions are feasible and some not. An optimal solution is a candidate solution, which need not be unique, satisfying the expression; that is, it is feasible and no feasible candidate solution yields a better value of $z(\boldsymbol{x})$.

[2] Or rather the effective Pareto frontier. While it is often true, it is not true in general that the solution to a constrained optimization problem lies on the Pareto frontier of the constraint set. In this paper when we speak of the Pareto frontier we mean the effective Pareto frontier, that is the set of solutions such that each is feasible (is within the constraint set) and such that no decision variable in the solution can be singly changed so as to improve the objective function and have the solution remain feasible.

Table 1. Summary of Group 1 Problems

Problem	Min/Max	Objective	# Variables	# Linear Inequalities	# Nonlinear Inequalities
1	min	Linear	8	3	3
2	min	Polynomial	7	0	4
3	min	Quadratic	10	3	5
4	min	Polynomial	2	0	2
5	min	Linear	2	0	2
6	min	Polynomial	2	0	2
7	min	Quadratic	2	0	2
8	min	Quadratic	2	1	1
9	max	Nonlinear	20	1	1
10	max	Nonlinear	50	1	1

("feasibility improvement") the population is evaluated with the fitness function $\min_{x_i} P(\boldsymbol{x})$ and a new generation is created in the usual way. The procedure continues, one phase following the other, until a stopping condition is met.

Kimbrough et al. [8] obtained encouraging results for a specialized class of constrained optimization problems (knapsack problems), demonstrating that (as measured by the number of fitness evaluations employed) their two-market GA outperformed generally accepted GA penalty function approaches. Knapsacks are simple, linear integer programming problems. They are known to be NP-hard, yet in practice they have proved tractable and there exist excellent polynomial time heuristics.

The investigations we describe in §2 examined the effectiveness of the two-market GA on a collection of standard nonlinear constrained optimization problems. This is a challenging class of problems, for which improved solution techniques are actively being sought. Further, we introduce a variation of the two-market GA, which we call a *two-population* (and two-market) GA. The two-population GA is essentially equivalent to the single-population, two-market version, but it affords insight into how the GA is exploring the solution space to these constrained optimization problems. We discuss this in some detail in §3.

2   Results

Genocop II/III [11] is a sophisticated, highly-developed product, and may be taken to represent the state of the art for constrained optimization using GAs. We use it as our benchmark for evaluating the performance of our two-market GAs. It is convenient to separate the test problems we studied, and our discussions of them, into distinct groups.

Table 2. Summary of Group 1 Results

Problem	Min/Max	Best Known or Optimal*	Genocop II/III	Two-Market GA		
				Best of 10	Median	Std.
1	min	7049.330923*	7268.650	∅	∅	∅
2	min	680.6300573*	680.640	680.6374	680.7566	0.085123
3	min	24.3062091*	25.883	25.18437	25.61935	0.825277
4	min	0.25*	close	0.25	0.25	0
5	min	-5.5079*	close	-5.50773	-5.50708	0.001042
6	min	-6961.81381*	close	-6960.95	-6957.44	1.908346
7	min	5*	close	5	5	0
8	min	1*	close	1	1	0
9	max	0.80351067	0.80351067	0.80288	0.7888	0.12853
10	max	0.8348	0.83319378	0.809211	0.743844	0.054728

2.1 Group 1

Group 1 contains 10 test problems described by Michalewicz in [10].[3] Table 1 summarizes these problems.

Table 2 compares results reported by Michalewicz with results obtained by our two-market, single-population GA. In these experiments we used a float-ing point representation for real number variables, fitness-proportional selection, arithmetical crossover [10, page 112] with probability 0.4, non-uniform mutation with probability 0.4 [10, page 103].[4] The results reported for Genocop are for the best solution found from 10 runs. In problems 1–3, the Genocop and the two-market GA runs are for 5,000 generations with a population of 70. (In the two-market GA case, this means 2,500 generations of phase 1 and 2,500 generations of phase 2, alternating). For problems 4–8 Michalewicz reports that Genocop achieved "convergence" to the known optimal solution within 1,000 generations for a population of 70. The two-market GA results reported are also for 1,000 generations—or rather half-generations (500 each, phase 1 and phase 2)—with a population of 70. The Genocop results for problems 9–10 are for 10,000 gen-erations on a population of 70. In the two-market GA, the population size was 50 and the total number of half generations was 10,000 (5,000 phase 1; 5,000 phase 2). We report the best solution found, the median best solution, and the standard deviation of the best solution found over 10 runs. In problem #1, the two-market GA failed to find a feasible solution. We note that Genocop requires a feasible solution at inception; it is not clear how Michalewicz obtained the necessary solution(s) for Genocop. On the remaining problems, the performance of this simple, two-market, single-population GA compares well with Genocop.

---

[3] The correspondence between our numbering and his is, using format (our-number, page-ref, his-identifier): (1, 145, test case #2), (2, 145, test case #3), (3, 146, test case #5), (4, 137, test case #1), (5, 137, test case #2), (6, 138, test case #3), (7, 138, test case #4), (8, 140, test case #5), (9, 156, Keane (n=20)), (10, 156, Keane (n=50)).

[4] In all our experiments we employed elitest selection [10, page 59].

Table 3. Summary of Group 2 Problems

Problem	Min/Max	Objective	# Variables	# Linear Inequalities	# Nonlinear Inequalities
11 (test11)	max	Nonlinear	2	0	2
12 (chance)	min	Linear	3	2	1
13 (circle)	min	Linear	3	0	10
14 (ex3_1_4)	min	Linear	3	2	1
15 (ex7_3_1)	min	Linear	4	6	1
16 (ex7_3_2)	min	Linear	4	6	1
17 (ex14_1_1)	min	Linear	3	0	4
18 (st_e08)	min	Linear	2	0	2
19 (st_e12)	min	Nonlinear	3	3	0
20 (st_e19)	min	Polynomial	2	1	1
21 (st_e41)	min	Nonlinear	4	0	2

Problems 4–8 are relatively easy, while 1–3 and 9–10 are much more challenging. On problems 9–10, the Genocop results are superior to the two-market GA, while on problems 2–3 Genocop falls behind. We would also draw the reader's attention to the reasonably low standard deviations for the two-market GA.

2.2 Group 2

Group 2 consists of 11 nonlinear programming problems. Ten are from the GLOBAL Library at GAMS World [6]. The Library is "a web site aiming to bridge the gap between academia and industry by providing highly focused forums and dissemination services in specialized areas of mathematical programming." As such it is an important source of mathematical programming (constrained optimization) problems that have proved valuable and interesting to the operations research and management science community (cf., INFORMS [5]). Demonstrable success on these problems with evolution programming would cause the INFORMS community to take notice. An eleventh problem, "test11", appeared in a paper by Schoenaur and Xanthakis [17]. Table 3 summarizes the Group 2 problems, all of which have only inequality constraints (rather than equality constraints), perhaps after algebraic transformation by us.[5]

We obtained the Genocop results by executing Genocop III [11] for 10,000 generations, using the default settings (as recommended), including a population of size 70. The two-market GA was run for 10,000 half generations (5,000 phase 1, 5,000 phase two, alternating), with a population size of 50. Again, we used a floating point representation for real number variables, fitness-proportional selection, arithmetical crossover with probability 0.4, non-uniform mutation with probability 0.4. Because we have not tuned the crossover or mutation rates, and we are using a smaller population, it has to be concluded that the two-market

---

[5] We so transformed problems 12 and 19.

Table 4. Summary of Group 2 Results. † = 29.8943939275064, ‡ = **0.324275369**.

Problem	Best Known or Optimal*	Genocop III			Two-Market GA		
		Best of 10	Median	Std.	Best of 10	Median	Std.
11	?	0.115047	0.11504	4.87E-05	0.115047	0.115047	1.12E-07
12	29.8943781591	29.89549	29.94807	0.034976	†	29.90093	0.003931
13	4.57424778502	4.574318	4.575747	0.027615	4.574249	4.574257	9.01E-06
14	-4.0000	-4	-4	0.032482	-4	-4	3.01E-05
15	0.341739553124	0.3558	0.416292	0.141607	‡	0.328268	0.010168
16	1.08986397147	1.09145	1.113644	0.033588	1.089928	1.089985	0.000111
17	0.0000	1.44E-10	1.19E-06	2.85E-06	6.18E-06	0.000172	0.00031
18	?	0.741782	0.741782	9.93E-09	0.741782	0.741782	1.09E-07
19	?	-4.5099	-4.49564	0.02252	-4.50691	-4.49855	0.004872
20	?	-118.705	-118.704	0.021621	-118.705	-118.705	0.001853
21	?	645.626	648.9749	4.398663	641.8242	642.0479	1.827517

GA is not operating at any advantage over Genocop III in these experiments. Table 4 summarizes our results.

Note that for problem 15, the two-market GA has found a solution (shown in **bold**) that is superior to the best solution otherwise known to us.

2.3   Group 2 with a Two-Population GA

The two-market GA introduced and explored for knapsack problems by Kimbrough et al. [8], and further explored by us, is not the only possible form for a two-market GA. Theirs is a *single-population* two-market GA. We introduce here a *two-population* two-market GA.

In the two-population GA, the *feasible population* contains only feasible solutions to the constrained optimization problem to hand, while the *infeasible population* contains only infeasible solutions. The process is simple. The *population-size* parameter is chosen in the customary manner (we use 50). The *initialization-pool* parameter is set (we used 100). At initialization we randomly generate at most *initialization-pool× population-size* solutions. When a solution is created we test it for feasibility. If it is feasible and the feasible population is not complete (has less than *population-size* solutions), we add the solution to it. Similarly, if the infeasible population is incomplete and the solution is infeasible, we add it to the infeasible population. After initialization, each population has at most *population-size* solutions or chromosomes. Either population may have fewer, if a sufficient number were not found.

In phase 1 of the two-population (+two-market) GA, the fitnesses of all solutions in the feasible population are calculated based only on the objective function, $z(\boldsymbol{x})$, and a new generation is created under the chosen genetic regime. The regime is any standard GA with the exceptions that (a) the highest fitness solution is placed directly into the next generation and (b) any infeasible daughter solutions created by the genetic operations are placed in the infeasible

population. A total of *population-size* daughter solutions are created each generation, but the size of the succeeding generation may be considerably smaller.

In phase 2 of the two-population GA, fitnesses of all the solutions in the infeasible population are calculated. (Because of contributions resulting from phase 1, more than *population-size* solutions may be present.) The fitness calculations are based only on the employed penalty function for the constraint set, $P(\boldsymbol{x})$, and not on the objective function.[6] The best *population-size* infeasible solutions are candidates to be parents for the next generation. As in phase 1, (a) the highest fitness solution is placed directly into the next generation and (b) any feasible daughter solutions created by the genetic operations are placed in the feasible population.

Phase 1 and 2 are processed in alternation until a stopping condition is reached. Table 5 shows our results with Group 2 using the two-population GA for 10,000 half generations (5,000 phase 1, 5,000 phase 2). Otherwise, the GAs employed are exactly as in the single-population, two-market version discussed above. Note that for problem 15 the two-population GA has found solution

Table 5. Summary of Group 2 Results for a Two-Population GA. † = 29.8943786178232, ‡ =**0.319777729979837**.

Problem	Best Known or Optimal*	Genocop III			Two-Population GA		
		Best of 10	Median	Std.	Best of 10	Median	Std.
11	?	0.115047	0.11504	4.87E-05	0.115047	0.115047	4.16E-08
12	29.8943781591	29.89549	29.94807	0.034976	†	29.91486	0.032017
13	4.57424778502	4.574318	4.575747	0.027615	4.574248	4.574257	6.67E-05
14	-4.0000	-4	-4	0.032482	-4	-3.99991	0.000131
15	0.341739553124	0.3558	0.416292	0.141607	‡	0.320127	0.00033
16	1.08986397147	1.09145	1.113644	0.033588	1.089952	1.089994	2.56E-05
17	0.0000	1.44E-10	1.19E-06	2.85E-06	4.69E-05	6.63E-05	1.36E-05
18	?	0.741782	0.741782	9.93E-09	0.741782	0.741782	5.58E-08
19	?	-4.5099	-4.49564	0.02252	-4.51347	-4.51319	0.000995
20	?	-118.705	-118.704	0.021621	-118.705	-118.705	0.000187
21	?	645.626	648.9749	4.398663	641.8244	641.8283	2.663347

(shown in **bold**) that is superior to the best solution otherwise known to us. Note further that the two-population GA has found a solution[7] to problem 15 that is superior to the best solution[8] that the single-population GA found, which itself is better than the best known solution we are otherwise aware of.

---

[6] Throughout our two-market runs we used the sum of absolute violations on constraints as our penalty function. We leave for future research, on tuning the two-market GAs, the problem of finding optimal penalty functions.

[7] $x_1 = 1055.688727, x_2 = 3.360444871, x_3 = 5.040713598, x_4 = 0.319777719.$

[8] $x_1 = 1053.464266, x_2 = 3.361225273, x_3 = 5.029994467, x_4 = 0.324275369.$

2.4   Summary and Comments on Results

We have investigated a number of constrained optimization problems that are nonlinear in their objective functions or in their constraints (or both). This class of problem is interesting because it arises often in practice and because standard OR/MS solution methods are often defeated by practical problems of this sort. The problems we discuss above are the problems we have investigated. Specifically, none have been left out, e.g., because results were disappointing.[9]

Although it is imperative to study many more problems in this class, these results are broadly quite encouraging. The simple, untuned, and in many ways naïve, single-population, two-market GA of Kimbrough et al. is apparently robust for nonlinear optimization. It compares favorably with the performance of Genocop, an excellent, well-tuned, state-of-the-art conventional GA package. In addition, we have introduced a two-population variant of a two-market GA, and shown that it too performs well (at least on the Group 1 and Group 2 problems we have investigated).

3   Discussion

We conclude with a discussion focusing on understanding why the two-market GAs should perform well for constrained optimization. This ought to be useful in designing future investigations.

Here is an easy question: Why, in a constrained optimization problem attacked with a GA (or EP) should we have *any* penalty associated with constraint violation? The answer, of course, is that the constraints are part of the problem and unless we provide information from them to the GA solving procedure there is no reason to think that the right problem will be addressed. Something has to be done with the constraints.

A more interesting question: Why not impose prohibitive penalties on constraint violation, why not simply eliminate infeasible solutions from consideration? Answer: it's been tried and doesn't work very well.[10] Why not? Expanding on suggestions in Kimbrough et al. [8], we should think of the GA as creating a population of solutions that is driven towards and then explores the Pareto frontier[11] of the constrained optimization problem.

During this stochastic exploration process solutions will inevitably be created that "step over the line" from the feasible region to infeasibility. Are these

---

[9] More carefully, we are not reporting experience with nonlinear problems having equality constraints for which it is not straightforward to eliminate a variable and thereby convert to an inequality constraint. We examined three such problems with mixed results.

[10] Repair of infeasible solutions is a more modest variant of this approach and it sometimes does work reasonably well. We defer discussion of repair to future work. We note further that the "death penalty" method, which liquidates solutions found to be infeasible, has been reported to perform poorly for constrained optimization problems, e.g., [10].
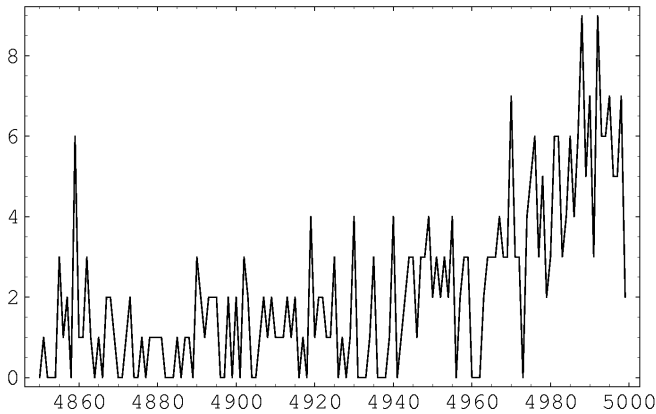
[11] See earlier comments qualifying this terminology.
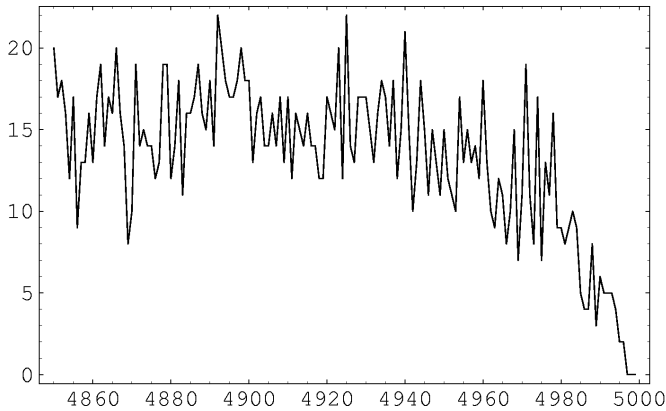
Fig. 1. For problem ex7_3_2 with population size 50: count of number of infeasible solutions created from the feasible population, by generation, 4850–4999.

solutions now worthless? Surely not, one would conjecture. Even a randomly generated pool of solutions will typically contain useful genetic material, which after recombination, mutation, etc. may prove valuable indeed. One would expect that infeasible solutions, descending from feasible parents and pressured by selection to minimize infeasibility, might have a fruitfully dense presence of useful genetic material. The two-market GAs, in both the single- and double-population forms, present ways of preserving genetic material in infeasible solutions so that it might be used by future generations.

The two-population GA is convenient for studying the exchange dynamics between feasible and infeasible solutions. For the two-population GA we described above and for the problems we applied it to, we found a robust pattern of behavior. First, of the 50 daughter solutions created from the feasible population each generation, roughly 10-20 are infeasible and get put into the infeasible population. This number tends to tail off to nearly zero as the population gets very close to finding an optimal solution. See Figure 1 for a representative example. Second, of the 50 daughter solutions created from the infeasible population each generation, roughly 0-2 are feasible and get put into the feasible population. This number typically increases, roughly to 0-4 then 2-8, as the number of generations increases beyond 5,000. See Figure 2 for a representative example. These two findings indicate that the infeasible population is steadily and increasingly sending genetic material to the feasible population. Our third finding confirms this. If we track for each feasible solution whether it has an infeasible ancestor we find that very quickly, before 20 generations have elapsed, *every* solution in the feasible population is descended from one infeasible solution or another! See Figure 3 for a representative example.

Fig. 2. For problem ex7_3_2 with population size 50: count of number of feasible solutions created from the infeasible population, by generation, 4850–4999.

These findings help us understand the dynamics of constrained optimization with GAs. Yet as good findings should, they raise more questions than they answer. How can these simple two-market GAs (both single- and double-population GAs) be improved and tuned? How broadly will they work? By characterizing the structure of the effective Pareto frontier can we gain useful information for configuring the GAs? How does the richness of the genetic material in the infeasible population vary over generations and how does it compare to that in the feasible population? What are the most effective forms of penalty functions? Should scaling be introduced on constraints? There will be an exchange rate in computational value between maintaining an incrementally larger feasible population and an incrementally larger infeasible population. At some point—it would appear from our results—an extra infeasible solution is worth more for optimization than an extra feasible solution. When does this happen? Specifically, we have maintained identical target sizes for both the feasible and the infeasible populations. Is this the right tradeoff or might it be improved? What can be done to maximize, or even diagnose, the value of the genetic material in the infeasible population? And so forth.

In addition, GAs and EPs are members of a larger family of *metaheuristics* for constrained optimization [13,15,18]. This larger family includes simulated annealing, neural networks, tabu search, extremal optimization [2], and other forms of local search heuristics. We are especially intrigued by parallels in the concepts that different heuristics attempt to instantiate. For example crossover in GAs and annealing in simulated annealing arguably are parallel approaches to avoid greedy hill-climbing in favor of a less myopic form of local search. The underlying concept for both forms of the two-market GA is to maintain a diversity of useful partial solutions (genetic material) and to facilitate subsequent use of this information. A tantalizing parallel concept, called *demon algorithms,*
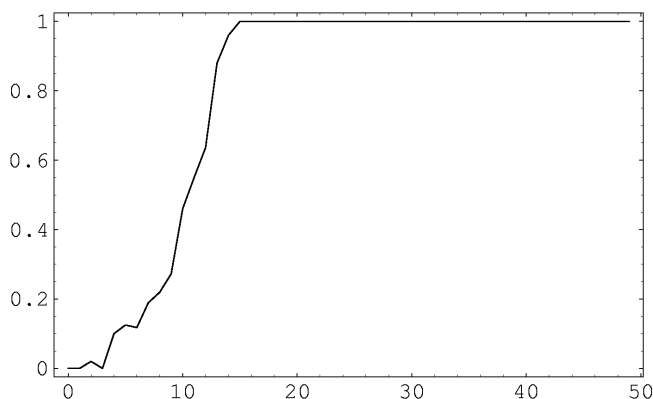
Fig. 3. For problem ex7_3_2 with population size 50: fraction, by generation, of feasible individuals with an infeasible ancestor.

has recently been brought to our attention. (There are various forms; see [3] for an overview.) Demons act rather like lines of credit for a search process. If the process is minimizing, it is always permitted to go down hill without drawing on its line of credit (its demon). Faced with a solution that would take the process up hill, however, it may draw on its available line of credit and undertake a local upwards climb. The draw may be deterministic or stochastic, depending on the variety of algorithm in play. Convergence may be realized by gradually reducing the available credit.

We see demon algorithms as motivated by an intuition or concept similar to that underlying the two-market GAs. Related ideas appear elsewhere, e.g. [14]. What other forms might this concept take? How is it best employed and when will work or not work well? Our two-population GA would seem to be a GA analog of primal-dual algorithms [1,7], with two bodies of solutions—one primal feasible, the other dual feasible—being driven towards each other. These and many other questions are now on the table, due in part to the encouraging results obtained, here and earlier, with two-market GAs.

References

1. Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms.* John Wiley & Sons, Inc., New York, NY, 1993.
2. Stefan Boettcher and Allon G. Percus. Extremal optimization: An evolutionary local-search algorithm. In Hemant K. Bhargava and Nong Ye, editors, *Computational Modeling and Problem Solving in the Networked World: Interfaces in Computer Science and Operations Research*, Operations Research/Computer Science Interface Series, pages 61–77. Kluwer, Boston, MA, 2003.

3. Bala Chandran, Bruce Golden, and Edward Wasil. A computational study of three demon algorithm variants for solving the traveling salesman problem. In Hemant K. Bhargava and Nong Ye, editors, *Computational Modeling and Problem Solving in the Networked World: Interfaces in Computer Science and Operations Research*, Operations Research/Computer Science Interface Series, pages 155–76. Kluwer, Boston, MA, 2003.

4. Carlos Artemio Coello Coello. A survey of constraint handling techniques used with evolutionary algorithms. Technical report Lania-RI-99-04, Laboratorio Nacional de Informática Avanzada, Veracruz, México, 1999.
http://www.lania.mx/~ccoello/constraint.html.

5. Institute for Operations Research and Management Science. Informs online. Pages on the World Wide Web, Accessed 2003-01-19. URL: http://www.informs.org.

6. GAMS World. Global world. Pages on the World Wide Web, Accessed 2003-01-19. URLs: http://www.gamsworld.org/global/index.htm,
http://www.gamsworld.org/global/globallib.htm.

7. Arthur M. Geoffrion. Duality in nonlinear programming: A simplified applications-oriented development. *SIAM Review*, 13(1):1–37, January 1971.

8. 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.

9. Zbigniew Michalewicz. 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.

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

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

12. Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, Germany, 2000.

13. I. H. Osman and J. P. Kelly, editors. *Meta-Heuristics: Theory and Application*. Kluwer, Boston, MA, 1996.

14. Carlos-Andres Pena-Reyes and Moshe Sipper. Fuzzy CoCo: Balancing accuracy and interpretability of fuzzy models by means of coevolution. *IEEE Transactions on Fuzzy Systems*, 9(5):727–737, October 2001.

15. Colin R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, New York, NY, 1993.

16. Ruhul Sarker, Masoud Mohammadian, and Xin Yao, editors. *Evolutionary Optimization*. Kluwer, Boston, MA, 2002.

17. M. Schoenauer and S. Xanthakis. Constrained GA optimization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 573–80. Morgan Kaufmann, 1993.

18. S. Voß. Metaheuristics: The state of the art. In A. Nareyek, editor, *Local Search for Planning and Scheduling*, volume 2148 of *Lecture Notes in Computer Science*, pages 1–23. Springer, Heidelberg, Germany, 2001.