Large Populations Are Not Always The Best Choice In Genetic Programming

Matthias Fuchs
Automated Reasoning Project, RSISE
Australian National University
Canberra ACT 0200, AUSTRALIA
Email: fuchs@arp.anu.edu.au, URL: http://arp.anu.edu.au:80/~fuchs

Abstract

In genetic programming a general consensus is that the population should be as large as practically possible or sensible. In this paper we examine a batch of problems of combinatory logic, previously successfully tackled with genetic programming, which seem to defy this consensus. Our experimental data gives evidence that smaller populations are competitive or even slightly better. Moreover, hill-climbing appears to exhibit the best performance. While these results are in a way unexpected, theoretical considerations provide a possible explanation in terms of a special constellation rather than a general misconception as to the benefits of large populations or genetic programming as such.

1 Introduction

Genetic programming (GP) [Koza, 1992] is an evolutionary search technique based on the genetic algorithm [Holland, 1992]. Given the problem-dependent *fitness measure*, the search behaviour of GP depends on the *genetic operators*, the *selection strategy*, and the *size of the population* (and, to a lesser extent, on how the initial random generation G_0 is produced).

From the beginning, *crossover* was the predominant genetic operator. Recent work, however, has shown that alternative genetic operators, in particular variants of *mutation*, can perform just as well or even better [Angeline, 1997; Harries and Smith, 1997; Hooper et al., 1997; Luke and Spector, 1998; Fuchs, 1998]. A variety of selection strategies is being utilised in numerous implementations of GP (e.g., fitness-proportionate selection, various forms of tournament selection, elite selection, etc.). Similar to the genetic operators, the way the selection strategy affects the search behaviour is hard to predict, and its appropriateness depends on the given circumstances.

As for the population size, a general rule in GP is to use a population which is as large as practically possible or sensible. Results presented in [Gathercole and Ross, 1997] showed that large populations can be outperformed by smaller ones. Thus, a suitable population size also appears to depend on the problem to be solved (and the choices made for all the other components of GP).

[Fuchs et al., 1997] demonstrated a very successful application of GP to theorem proving in the context of *combinatory logic* (CL). The experiments given in [Fuchs et al., 1997] showed that the search effort of GP (i.e., the number of individuals that need to be processed to obtain a successful individual with a probability of 0.99) was roughly the same for population sizes 500, 1000, and 2000. In this paper, we present empirical evidence which indicates that the performance of GP does not deteriorate even when reducing the population size to 50 (in the context of crossover) or when using the most extreme form of mutation, namely *random hill-climbing*.

While the experimental results are interesting in themselves, we also examined these problems theoretically to shed some light on the somewhat peculiar and in a way unexpected behaviour of GP. The theoretical considerations provide possible explanations as to why smaller populations can perform as well as, or even better than, larger populations. This is particularly helpful and reassuring in view of the possibly inadequate and consequently unreliable experimental data obtained in connection with GP [Daida et al., 1997]. While isolated experimental data may entice people to jump to conclusions ([Lang, 1995] countered by [Koza, 1995]), theoretical investigations provide a safer ground for drawing valid conclusions, and often make it possible to identify disturbances of general trends as particular constellations rather than evidence of misconception or basic flaws. Thus, the main purpose of this paper is to demonstrate that, on the one hand, there are situations and problems that do not conform with common believes. On the other hand, the paper shows that these situations do not constitute a reasonable cause for discrediting common believes regarding GP or even GP itself.

We begin with a short introduction to CL and the way it is tackled with GP.

2 Combinatory Logic and GP

Problems of CL are fundamental for studying theoretical and practical aspects of functional programming [Peyton Jones, 1987]. Such problems are typically tackled with automated theorem provers. [Fuchs et al., 1997] introduced a method for solving certain problems of CL with GP and examined its performance on 30 problems. The method performed very well, solving 27 of the 30 problems. As a matter of fact, GP clearly outperformed theorem provers regarding these problems, which was also acknowledged by the theorem proving community [Fuchs, 1997].

In [Barendregt, 1981] CL is defined by equational axioms for the combinators S and K: $Sxyz = xz(yz)$ and $Kxy = x$. Expressions are implicitly left-associated, i.e., $xyz \stackrel{\text{def}}{=} (xy)z$. x, y, and z denote variables. Besides combinators S and K various other combinators can be studied. The general idea is that combinators represent (functional) programs that are specified by defining equations. For an arbitrary instantiation of the variables on the left-hand side of the equation these programs produce the correspondingly instantiated expression on the right-hand side as output. A given set of combinators is called a *basis*.

The problems we are dealing with here can be characterised as follows: Given a basis $\mathcal{B}$, the question is whether the *elementary* combinators in $\mathcal{B}$ can be combined so that the resulting *compound* combinator exhibits a desired input-output behaviour. (Henceforth, elementary and compound combinators will not be distinguished and simply referred to as combinators.) Consider, for instance, $\mathcal{B} = \{B, W\}$ where $Bxyz = x(yz)$ and $Wxy = xyy$. The question is whether there is a combinator Θ composed of Bs and Ws that satisfies $\Theta xy = x(yy)$. As a matter of fact, $\Theta = BWB$ is a solution of this problem since $BWBxy = W(Bx)y = Bxyy = x(yy)$.

In order to solve such problems of CL, GP is employed to search for combinators with the desired input-output behaviour. To this end we represent combinators with "proper" trees by introducing a binary function symbol a ("apply"). E.g., the combinators BWB and $B(WW)$ given in short notation then read as $a(a(B, W), B)$ and $a(B, a(W, W))$, respectively. Thus, the set of function symbols $F = \{a\}$ and the set of terminal symbols $T = \mathcal{B}$. The search space is the set of all (complete) binary trees with internal nodes labelled by a and leaves labelled by elements of $\mathcal{B}$.

We employed the following variant of GP: The initial random population or generation G_0 is generated with the *grow method* (cp. [Koza, 1992]). More precisely, a random binary tree is created by placing either the symbol a at the root with probability p_a or a combinator from $\mathcal{B}$ with probability $(1 - p_a) \cdot p_C$, where $p_C = 1/|\mathcal{B}|$. If symbol a is put at the root, we recursively proceed for both subtrees. The maximal depth of random trees is 10 (cp. [Fuchs et al., 1997] or [Fuchs, 1997]). In all our experiments we used $p_a = 0.5$.

A successor generation G_{i+1} is computed by using *elite* or *truncate selection*. In our experiments, the 30% fittest individuals of generation G_i are copied to G_{i+1}. These *surviving* individuals $\mathcal{S}$ then serve as parents for producing offspring to restock G_{i+1}. Offspring can be produced with crossover or mutation. We apply "standard" *one-point subtree crossover*. That is, crossover selects two (not necessarily distinct) parents from the pool of surviving individuals $\mathcal{S}$. (Note that all individuals in $\mathcal{S}$ have the same chance to become a parent.) In each parent a subtree is chosen at random, and two "children" are produced by copying the parents and exchanging the subtrees.

The mutation operator performs a simple random mutation (*one-point mutation*). A parent is drawn from $\mathcal{S}$, and a child is produced by copying the parent and replacing a randomly chosen subtree with a random tree. This random tree is created exactly like the random trees in generation G_0. The maximal depth of trees produced by crossover or mutation is 17.

The *fitness measure* Φ checks how well a combinator Θ realizes the desired input-output behaviour. To this end, the actual output of Θ is compared with the desired output using a simple tree-difference measure. If $\Phi(\Theta) = 0$ then Θ is a solution. Consequently, the success predicate is $\exists\Theta \in G_i : \Phi(\Theta) = 0$. (Due to space limitations we have to refer the reader to [Fuchs et al., 1997] or [Fuchs, 1997] for more details on Φ.)

3 Experimental Results

In [Fuchs et al., 1997] (and [Fuchs, 1997]) 30 problems of CL were investigated. (These problems can be found in the domain COL of the public TPTP library [Sutcliffe et al., 1994], a large collection of problems for theorem provers.) In this paper we concentrate on 18 of those 30 problems, omitting problems `COL006-1`, `COL037-1`, `COL042-1`, `COL043-1`, `COL057-1`, `COL067-1`, and COL072-1, because the success rate for these problems is so low (even 0 for three of these problems) that they cannot provide useful data for statistical and comparative analyses regarding search effort. We also omitted problems COL029-1 through COL033-1, because they are mostly solved by producing a random generation G_0, and therefore do not contribute useful information to comparative studies concerning population size.

The search effort is determined based on the (empirically obtained) number $I(M, G, z)$ of individuals that need to be processed to produce a successful individual by generation G with a probability z, given a population of size M (cp. [Koza, 1992]). Following [Koza, 1992], we let $z = 0.99$. Furthermore, $M \in \{50, 100, 500, 1000, 2000\}$. For a given problem, 100 runs of GP are executed which provide us with the (empirical) probabilities $P(M, G)$ to succeed by generation G. Each run is limited to $G_{max} = 50{,}000/M$ generations. The number of runs $R(M, G, z)$ required to produce at least one successful individual by generation G with probability z, given a population of size M, can be determined with the help of

$$1 - (1 - P(M, G))^{R(M,G,z)} = z$$

which gives us

$$R(M, G, z) = \frac{\log(1 - z)}{\log(1 - P(M, G))}$$

and

$$I(M, G, z) = M \cdot R(M, G, z) \cdot (G + 1).$$

The search effort is measured by

$$I^*(M, z) = \min\{I(M, G, z) \mid 0 \le G \le G_{max}\}.$$

(For $M = 50$, we let $G_{max} = 500$ for practical reasons: The experiments in particular with $M = 100$ and $M = 500$ had shown that the generation G^* accounting for the minimal search effort, that is $I^*(M, z) = I(M, G^*, z)$, occurred clearly before G_{max}. Hence, $G_{max} = 1000$ appeared to be an unnecessarily large value, which was sustained by $G^* \le 51$ in our experiments with $M = 50$. See also Section 5.) In all our experiments we used elite selection with a 30% survival rate, and the random generation G_0 was produced with the grow method as described in Section 2.

Table 1 shows the search effort required when employing one-point crossover. The first column lists the problem names, and the remaining columns show the respective value of $I^*(M, 0.99)$, where the value of M is given in the head of the columns. (The results for $M \in \{500, 1000, 2000\}$ are taken from [Fuchs et al., 1997].) Note the entry '—' in column 50 (i.e., $M = 50$) for problem COL004-1 which indicates that no successful individual could be produced in 100 runs.

Table 1: Search effort required when using crossover.

Problem	50	100	500	1000	2000
COL003-1	90,000	304,000	140,000	200,000	196,000
COL004-1	—	258,400	787,500	360,000	476,000
COL034-1	45,900	96,800	135,000	225,000	180,000
COL035-1	34,200	27,000	37,500	40,000	34,000
COL036-1	129,200	688,500	684,000	784,000	704,000
COL038-1	285,000	240,000	704,000	667,000	880,000
COL039-1	10,850	6,400	8,000	10,000	12,000
COL041-1	45,200	47,500	62,500	72,000	90,000
COL044-1	62,000	67,500	81,000	75,000	100,000
COL046-1	582,800	172,500	241,000	220,000	320,000
COL049-1	56,500	52,500	87,000	113,000	168,000
COL060-1	11,250	12,800	11,000	10,000	12,000
COL061-1	9,100	10,000	12,000	8,000	12,000
COL062-1	38,850	45,900	57,000	64,000	60,000
COL063-1	33,350	32,500	35,000	30,000	40,000
COL064-1	101,700	76,800	201,500	320,000	208,000
COL065-1	57,000	456,000	560,000	720,000	680,000
COL066-1	56,350	55,800	63,000	66,000	48,000

Table 2 lists the results for one-point mutation analogously to Table 1. In connection with mutation, we omitted the experiments with population size $M = 50$. Instead, we ran experiments with *hill-climbing* (HC) (see column 'HC' in Table 2). Hence, we pushed things to the limit since, essentially, hill-climbing corresponds to mutation with $M = 1$: In the beginning, a random individual is generated. Then the following process is iterated until a maximal number of iterations, also denoted by G_{max}, is reached, or a successful individual is produced. In each iteration, one offspring is obtained by applying one-point mutation. The parent is replaced with the offspring if the offspring is fitter than the parent. If the fitness is the same, the parent is replaced with a probability of 0.5.

For a given problem, the probability $P(G)$ for HC to succeed by iteration G was determined based on 1000 runs of HC. Analogously to the search effort of GP (for HC we have the special case $M = 1$) we obtain

$$R(G, z) = \frac{\log(1 - z)}{\log(1 - P(G))}, \quad I(G, z) = R(G, z) \cdot (G + 1)$$

and thus the search effort is

$$I^*(z) = \min\{I(G, z) \mid 0 \le G \le G_{max}\}.$$

In our experiments, we let $G_{max} = 2000$. Similar to $M = 50$ in connection with crossover, we preferred this

Table 2: Search effort required when using mutation.

Problem	HC	100	500	1000	2000
COL003-1	116,596	105,000	149,000	208,000	240,000
COL004-1	345,681	292,500	392,000	468,000	456,000
COL034-1	65,600	91,200	147,000	160,000	238,000
COL035-1	16,107	30,000	28,000	36,000	38,000
COL036-1	185,504	179,200	1,024,000	735,000	864,000
COL038-1	158,912	262,400	229,500	459,000	840,000
COL039-1	4,603	9,300	11,000	10,000	10,000
COL041-1	23,548	45,900	80,000	84,000	120,000
COL044-1	45,436	86,400	96,000	112,000	126,000
COL046-1	118,286	126,900	204,000	260,000	380,000
COL049-1	59,527	64,000	87,000	135,000	144,000
COL060-1	6,929	10,800	13,500	11,000	14,000
COL061-1	6,027	9,600	10,000	9,000	16,000
COL062-1	39,196	74,800	82,500	76,000	84,000
COL063-1	22,270	34,100	45,000	36,000	40,000
COL064-1	63,270	157,500	237,500	234,000	240,000
COL065-1	232,517	1,596,000	798,000	551,000	714,000
COL066-1	32,221	61,600	85,000	78,000	80,000

value to $G_{max} = 50,000$ for practical reasons, and used the released time for more runs (1000 instead of 100). See also Section 5 in this context.

When comparing the search effort for the various population sizes and HC we can detect no advantage on the parts of larger populations. As a matter of fact, HC appears to perform better than GP, which in turn appears to have a slightly better performance for smaller populations in connection with both crossover and mutation. (Note that our results here sustain the results presented in [Fuchs, 1998] as to a comparable performance of crossover and mutation for problems of CL.) A few runaways can be attributed to the statistical inadequacy of our data due to a restricted number of runs to obtain $P(M,G)$ or $P(G)$. The restriction is of course necessary for practical reasons. (Executing 100 runs with a certain populations size M for all 18 problems took ca. 24 hours on a SPARCstation Ultra.) Thus, the fact that no successful individual was found by crossover in 100 runs with $M = 50$ for problem COL004-1, or the superior performance of the same setting for problem COL065-1, should not be overrated.

The entirety of the statistical data for all 18 problems, however, draws a rather clear picture in favour of HC. We underline this observation with a brief statistical analysis. The search effort of HC is compared with the search effort of GP using mutation and $M \in \{100, 500, 1000, 2000\}$ (cp. left-hand side of Table 3). Given a certain M (see the first column of Table 3), the data pairs $(I_1^*(z), I_1^*(M,z)), \ldots, (I_{18}^*(z), I_{18}^*(M,z))$ representing the search effort of HC and mutation for the 18 problems are subject to linear regression resulting in a slope and offset value given in the respective columns of Table 3, expressing the search effort of mutation as a linear function of the search effort of HC. We also computed the correlation coefficient for the data pairs. Especially for $M \geq 500$ we have a rather good positive correlation, a slope of approximately 2, and a positive offset. The same comparison between crossover with $M = 50$ and crossover with $M \in \{100, 500, 1000, 2000\}$ is not as impressive (cp. right-hand side of Table 3), but it does not reveal a clearly inferior performance for $M = 50$. (We omitted COL004-1 from this comparison.)

Table 3: Comparing HC with mutation, and crossover ($M = 50$) with crossover, for various population sizes $M \in \{100, 500, 1000, 2000\}$.

	HC			crossover with $M = 50$		
pop. size	corr.	slope	offset	corr.	slope	offset
100	0.57	2.24	-11,965	0.25	0.34	107,783
500	0.72	2.14	23,028	0.42	0.69	116,532
1000	0.86	1.96	35,166	0.33	0.61	154,221
2000	0.78	2.37	54,595	0.45	0.88	135,199

In view of our experimental results the benefits of large populations are questioned. In the following section we try to shed some light on these in a way unexpected observations with theoretical considerations.

4 Theoretical Considerations

In this section we give possible explanations for the observations made in conjunction with our experiments in Section 3. According to these experiments (see also [Fuchs, 1998]) crossover and mutation do not cause GP to have a considerably different performance. Therefore, we shall concentrate on the effects population size has on GP when using mutation as the sole genetic operator, and we shall later comment on crossover.

In view of the excellent performance of HC we want to examine if we can gain anything with respect to search effort if we have a population of M parallel and *independent* "hill-climbers". Given the probability $P(G)$ for one hill-climber to succeed by generation G, the probability $Q(M,G)$ that at least one of the M hill-climbers succeeds by generation G is

$$Q(M,G) = 1 - (1 - P(G))^M.$$

Hence we have

$$R(M,G,z) = \frac{\log(1-z)}{\log(1-Q(M,G))} = \frac{\log(1-z)}{M \cdot \log(1-P(G))}$$

and consequently

$$\begin{aligned} I(M,G,z) &= M \cdot R(M,G,z) \cdot (G+1) \\ &= R(G,z) \cdot (G+1) \\ &= I(G,z). \end{aligned}$$

That is, we cannot expect any gains w.r.t. search effort if we deploy M parallel and independent hill-climbers.

The search conducted by both HC and GP is characterised by *exploration* and *exploitation*. Exploration is a more dynamic process that samples large areas of the search space to open up areas of high fitness, whereas exploitation is a more static process that examines the search space in the "vicinity" of individuals with high fitness. In connection with HC, the distinction between an initial (and quick) exploration phase ("rushing up or down a hill") and the subsequent (much longer lasting) exploitation phase ("slowly wandering around on a plateau") is rather clear, with a kind of sudden transition between the two phases.

For GP, the transition from exploration to exploitation is more of a continuous nature, emphasising exploration and massive changes in the population during the initial stages of the search. The exploration aspect gradually diminishes, giving way to more and more exploitation with a more stable population. However, assuming that convergence has not occurred, there is always a certain amount of exploration going on that is missing in HC once a plateau has been reached. These remaining exploration capabilities are responsible for GP having a better chance to escape from local optima than HC, without sacrificing the benefits of exploiting high-fitness areas. For larger populations, this property becomes more and more notable and effective.

When searching on a plateau, which can be considered exploitation, GP (using mutation) and HC do not differ. This becomes particularly clear in the extreme case where the fitness landscape is one huge plateau with very few "holes" representing solutions. In this case the chance to find a solution increases with the number of trial solutions tested. ("After wandering around the plateau long enough one may eventually step into a hole.") That is, we basically have the same situation as in the case of M hill-climbers (GP searches in a different, but not more effective way), and consequently the population size does not matter. (Under these circumstances random search should be just as effective. However, our fitness measure does not produce such a monotonous fitness landscape, and as a result random search is not competitive as experiments presented in [Fuchs, 1997] have shown.)

HC gains an advantage over GP if the exploration aspect still present in GP later in the search is not useful. That is, focusing on the plateau the search has reached is the best way to continue the search. Under these circumstances, exploration results in wasted effort. (Preliminary experimental results indicate that this seems to play a role in our case, but a more thorough examination is required before definite statements can be made.)

Thus, there are situations conceivable where HC performs better than or as well as GP, and larger populations do not pay off. Consequently, we are able to provide possible explanations as to why GP with larger populations does not produce the results commonly expected. These explanations centre on certain unfavourable conditions, rather than general flaws of GP and the use of large populations.

In the context of crossover, larger populations are preferred not only because they promise advantages concerning the search behaviour as such, but also because larger populations provide more genetic material and thus the fundamental pieces for potential *building blocks*. For problems of CL, however, there is only one function symbol and usually two or three terminal symbols. Furthermore, [Fuchs, 1998] showed that, in order to create a successful individual, the subterms exchanged by crossover or replaced by mutation are, on average, rather small (mostly of size 7 or less). Consequently there is no need for a large population in order to supply sufficient genetic material. This, and the fact that crossover and mutation seem to perform equally well for problems of CL (cp. [Fuchs, 1998]), help to explain the experimental results regarding crossover in a way that is consistent with what was said in connection with mutation above.

5 Discussion

In this paper we have given experimental evidence that large populations (the "default rule" in GP) sometimes do not perform better than small populations. Theoretical considerations have provided possible explanations for this behaviour in that the shortcoming of GP can be attributed to certain unfavourable conditions, which is not a sign of weakness of GP, but rather an inevitable fact of search-based problem-solving techniques ("no free lunches").

Our main performance criterion was the search effort expressed by the (minimal) number of individuals that need to be processed to produce a solution with a probability of 0.99. To obtain reliable data for the search effort requires a rather large number of runs. Conclusions drawn on the basis of few runs run the risk of being misleading or just plain wrong. In this context we want to recant the statement made in [Fuchs et al., 1997] which basically stated that the performance of GP (employing crossover) degrades significantly for a population size of 100 (or less). According to our now available more reliable data this is not the case. Similarly, we advise to be cautious with "one or two exploratory runs" suggested in [Gathercole and Ross, 1997] to determine whether or not a smaller population is more appropriate than a larger one.

Difficulties with empirical comparisons in connection with GP arise on many levels (cp. [Daida et al., 1997]). One difficulty is the choice of G_{max}. Cutting off the evolutionary process at that point may prevent us from obtaining the "true" (minimal) search effort. For our experiments, this concern is rather weak, since the generation associated with the search effort I^* mostly occurs well before G_{max}. Naturally, this empirically established fact cannot guarantee that the search effort does not become even smaller if we go beyond G_{max}, but at least it is good evidence that this contingency is very unlikely.

An interesting topic for future research is to put the theoretical considerations given in this paper, which were partly given in "prose", on a more solid mathematical basis. That is, from a theoretical point of view, it is an interesting subject to find ways to analyse the interplay between fitness measure and the resulting fitness landscape as well as the selection strategy and the genetic operators in order to be able to explain more formally the consequences for the search behaviour of GP. From a practical point of view, these studies may eventually provide the possibility to choose an appropriate selection strategy and suitable genetic operators based on the given fitness measure, or even to suggest alternative, better suited, search methods.

References

Angeline, P. (1997). Subtree crossover: Building block engine or macromutation? In *Proc. 2nd Annual Conference on Genetic Programming (GP-97)*, pages 9–17. Morgan Kaufmann.

Barendregt, H. (1981). *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, Amsterdam.

Daida, J., Ross, S., McClain, J., Ampy, D., and Holczer, M. (1997). Challenges with verification, repeatability, and meaningful comparisons in genetic programming. In *Proc. 2nd Annual Conference on Genetic Programming (GP-97)*, pages 64–70. Morgan Kaufmann.

Fuchs, M. (1997). Evolving combinators. In *Proc. 14th Conference on Automated Deduction (CADE-14)*, LNAI 1249, pages 416–430. Springer.

Fuchs, M. (1998). Crossover versus mutation: An empirical and theoretical case study. In *Proc. 3rd Annual Conference on Genetic Programming (GP-98)*, pages 78–85. Morgan Kaufmann.

Fuchs, M., Fuchs, D., and Fuchs, M. (1997). Solving problems of combinatory logic with genetic programming. In *Proc. 2nd Annual Conference on Genetic Programming (GP-97)*, pages 102–110. Morgan Kaufmann.

Gathercole, C. and Ross, P. (1997). Small populations over many generations can beat large populations over few generations in genetic programming. In *Proc. 2nd Annual Conference on Genetic Programming (GP-97)*, pages 111–118. Morgan Kaufmann.

Harries, K. and Smith, P. (1997). Exploring alternative operators and search strategies in genetic programming. In *Proc. 2nd Annual Conference on Genetic Programming (GP-97)*, pages 147–155. Morgan Kaufmann.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems.* Ann Arbor: Univ. of Michigan Press, 2nd edition.

Hooper, D., Flann, N., and Fuller, S. (1997). Recombinative hill-climbing: A stronger search method for genetic programming. In *Proc. 2nd Annual Conference on Genetic Programming (GP-97)*, pages 174–179. Morgan Kaufmann.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

Koza, J. (1995). A response to the ML-95 paper entitled "Hill-climbing beats genetic search on a Boolean circuit synthesis problem of Koza's". `http://www-cs-faculty.stanford.edu/~koza/hill_climb.html`. Distributed at ML-95.

Lang, K. (1995). Hill-climbing beats genetic search on a Boolean circuit synthesis problem of Koza's. In Prieditis, A. and Russell, S., editors, *Machine Learning: Proceedings of the Twelfth International Conference (ML-95)*, pages 340–343. Morgan Kaufmann.

Luke, S. and Spector, L. (1998). A revised comparison of crossover and mutation in genetic programming. In *Proc. 3rd Annual Conference on Genetic Programming (GP-98)*, pages 208–213. Morgan Kaufmann.

Peyton Jones, S. (1987). *The Implementation of Functional Programming Languages.* International Series in Computer Science. Prentice-Hall.

Sutcliffe, G., Suttner, C., and Yemenis, T. (1994). The TPTP problem library. In *Proc. 12th Conference on Automated Deduction (CADE-12)*, LNAI 814, pages 252–266. Springer. See also `http://www.cs.jcu.edu.au/~tptp`.