Setting the Mutation Rate: Scope and Limitations of the $1/L$ Heuristic

Gabriela Ochoa
Universidad Simon Bolivar, Computacion y TI
Sartenejas, Miranda, Apartado 89000, Caracas 1080-A, Venezuela
E-mail: gabro@ldc.usb.ve
Tel.: +58 212 906 32 63, Fax.: +58 212 906 32 43

Abstract

An important decision to make when designing a GA is how to set the evolutionary parameters. Among these parameters, the mutation rate has been acknowledged as the most sensitive one. All approaches so far for a near-optimal setting of the mutation rate have intrinsic limitations. A promising guideline is, however, the heuristic suggesting $p_m = 1/L$ where L is the string length. This paper is a first attempt to explore the scope and limitations of this heuristic on GAs with bit-string representation. Specifically, we select two real-world domains as test problems and explore (i) whether optimal mutation rates change with time; and (ii) the interactions between the mutation rate and other evolutionary parameters. Results suggest that a constant mutation rate of $1/L$ is useful for a GA with a controlled 'moderate' selection pressure. It should be, however, revised for a weak or extremely strong selection pressure, and for a small population size.

1 INTRODUCTION

It has been suggested that the most sensitive of GA parameters is the mutation rate [Schaffer et al., 1989, Bäck, 1996]. Several studies in the literature look for 'optimal' mutation rates, or optimal schemes for varying the mutation rate over a single run [Fogarty, 1989, Davis, 1989, Bäck, 1991, Mühlenbein, 1992, Julstrom, 1995, Tuson and Ross, 1998].

We believe that the most useful guideline so far for an effective and general setting of the mutation rate in GAs is the heuristic suggesting $p_m = 1/L$ (per bit), where L is the string length. This figure has appeared several times in the evolutionary computation literature. The earliest appearance we can trace back was due to [Bremerman et al., 1966] as quoted by [Bäck, 1996]. Also, in his dissertation [DeJong, 1975] suggested this value as quoted by [Hesser and Männer, 1991]. The work of [Mühlenbein, 1992] states that $p_m = 1/L$ is optimal for general unimodal functions. This setting has also produced good results for several NP-hard combinatorial optimization problems such as the multiple knapsack problem [Khuri et al., 1994], the minimum vertex cover problem [Khuri and Bäck, 1994], the maximum independent set problem [Bäck and Khuri, 1994], and others [Bäck and Khuri, 1994]. The work of [Smith and Fogarty, 1996] found $1/L$ as the best fixed setting for the mutation rate, giving results comparable to their best self-adaptive method. Other authors have found a dependence of effective mutation rates upon the string length L, although they had not explicitly suggested $p_m = 1/L$ [Schaffer et al., 1989, Hesser and Männer, 1991, Bäck, 1992, Bäck, 1993].

Thus, there may well be some true principle underlying this heuristic. In previous work, we argued that this principle is related to the notion of error threshold from molecular evolution [Ochoa et al., 1999, Ochoa et al., 2000]. The error threshold is the minimal replication accuracy that still maintains genetic information in the population.

This paper is a first attempt to explore the scope and limitations of the $1/L$ heuristic on GAs with bit-string representation. Specifically, we select two real-world domains as test problems and explore (i) whether optimal mutation rates change with time; and (ii) the interactions between the mutation rate and other evolutionary parameters (the selection pressure and the population size).

The remainder of this document is organized as follows. Section 2 describes the test problems used in this paper: a combinatorial optimization problem — the Multiple Knapsack problem, and an engineering problem — the design of an optimal aircraft Wing-Box. Sections 3 and 4 describe our methods and results respectively, and, finally, Sections 5 and 6 summarizes our findings.

2 TEST PROBLEMS

Two real-world domains were selected for study, namely, a combinatorial optimization problem — the Multiple Knapsack problem, and an engineering problem — the design of an optimal aircraft Wing-Box. The Multiple Knapsack is a maximization problem, whereas the Wing-Box is a minimization problem. This selection is somewhat arbitrary, but is consistent with the following criteria. First, both are complex problems: the Wing-Box is an engineering design problem based on real data and constraints, and the Multiple Knapsack is a highly constrained combinatorial optimization problem known to be NP-hard. Second, both problems were available and relatively easy to implement, and third, both have a natural bit string encoding which was a requirement for the present study. Additionally, these two problems are completely unrelated, so common results have a good chance to convey some generality. It is worth noting, however, that other real-world problems may have very different characteristics from these two test problems.

2.1 THE WING-BOX PROBLEM

The Wing-Box problem was formulated as part of the Genetic Algorithms in Manufacturing Engineering (GAME) project at COGS, University of Sussex [1]. An industrial partner, British Aerospace, provided data from a real Airbus wing box.

A common problem faced in the design of aircraft structures, is to define structures of minimum weight that can withstand a given load. Figure 1 sketches the elements of a wing relevant to this problem. The wing is supported at regular intervals by slid ribs which run parallel to the aircraft's fuselage. On the upper part of the wing, thin metal panels cover the gap separating adjacent ribs. The objective is to find the number of panels and the thickness of each of these panels while minimizing the mass of the wing and ensuring that none of the panels buckle under maximum operational stresses. More details, and the equations for calculating the fitness function, can be found in [McIlhagga et al., 1996].

[1] http://www.cogs.susx.ac.uk/projects/game/

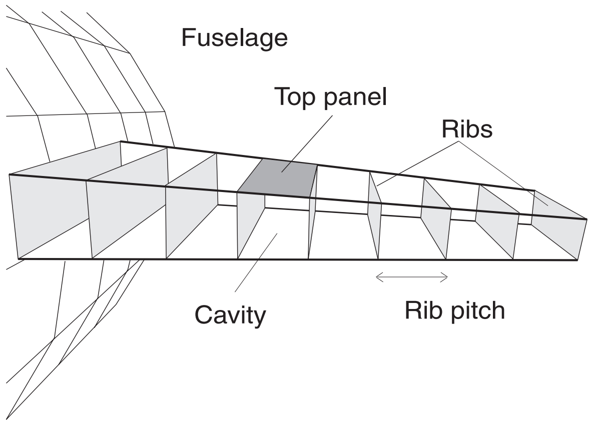


Figure 1: Relevant elements of a wing. Wing dimensions are fixed. The variable elements are the number of ribs and the thickness of the top panels.

A full description of a potential solution to the Wing-Box problem requires the definition of the number of ribs N and the thickness of the $N-1$ panels. There is a constraint on the thickness of these panels which is that adjacent panels should not differ in thickness by more than 0.25 mm. The simplest way to accomplish this, is to encode the differences in thickness between adjacent panels rather than the absolute thickness of the panels. If we know the difference in thickness $\delta th(i)$ between panels i and $i+1$ for $i \in (1, N-1)$, the absolute thickness of the first panel is enough to define everything else.

Originally, the Wing-Box parameters were encoded following the order described by Figure 2. For the experiments in this paper we fixed the number of panels in 50 (i.e $N = 51$ ribs, since the number of ribs is 1 + the number of panels), thus our genetic encoding is the same, but excluding the first gene. The thickness of the first panel was allowed to vary between 10 and 15 mm by steps of 10^{-3} mm. This requires 5×10^3 values which can be represented with a minimum of 13 bits. For all subsequent $N-2$ panels the difference in thickness with the previous panel is encoded. According to manufacturing tolerance considerations, only five values were allowed for these differences in thickness: $\{-0.25, -0.125, 0.0, 0.125, 0.25\}$. Three bits are needed to encode these five values. Notice that a change in $\delta th(i)$ leads to changes in the thickness of panel $i+1$, and of all subsequent panels up to the tip of the wing. Notice also that in both the encoding of the first section, and the remainder $N-2$ sections, there is an amount of redundancy in the genotype to phenotype mapping. To sum up, the number of bits needed for encoding an individual is 13 for the first panel, and 3 for each of the others 49 panels, that is $13 + 3 \times 49 = 160$.

N	th(1)	Δ th(1)= th(2)-th(1)	. . .	Δ th(i)= th(i+1)-th(i)	. . .	Δ th(N-2)= th(N-1)-th(N-2)

N: Number of ribs

th(i): Thickness of i$^{\text{th}}$ panel

Figure 2: Genetic representation of the wing parameters.

2.2 THE MULTIPLE KNAPSACK PROBLEM

The combinatorial optimization problem described here, called the 1/0 multiple knapsack problem, follows the specifications given by [Khuri et al., 1994]. This problem is a generalization of the 0/1 simple Knapsack problem where a single knapsack of capacity C, and n objects are given. Each object has a weight w_i and a profit p_i. The objective is to fill the knapsack with objects producing the maximum profit P. In other words, to find a vector $x = (x_1, x_2, \ldots, x_n)$ where $x_i \in \{0, 1\}$, such that $\sum_{i=1}^{n} w_i x_i \leq C$ and for which $P(x) = \sum_{i=1}^{n} p_i x_i$ is maximized.

The multiple version consists of m knapsacks of capacities $c_1, c_2, \ldots, c_m$ and n objects with profits $p_1, p_2, \ldots, p_n$. Each object has m possible weights: object i weighs w_{ij} when considered for inclusion in knapsack j ($1 \leq j \leq m$). Again, the objective is to find a vector $x = (x_1, x_2, \ldots, x_n)$ that guarantees that no knapsack is over-filled: $\sum_{i=1}^{n} w_{ij} x_i \leq c_j$ for $j = 1, 2, \ldots, m$; and that yields maximum profit $P(x) = \sum_{i=1}^{n} p_i x_i$.

This problem leads naturally to a binary encoding. Each string $x_1 x_2 \ldots x_n$ represents a potential solution. If the ith position has the value 1 (i.e. $x_i = 1$) then the ith object is in all knapsacks; otherwise, it is not. Notice that a string may represent an infeasible solution. A vector $x = (x_1, x_2, \ldots, x_n)$ that over-fills at least one of the knapsacks; i.e., for which $\sum_{i=1}^{n} w_{ij} x_i > c_j$ for some $1 \leq j \leq m$, is an infeasible string. Rather than discarding infeasible strings and thus ignore infeasible regions of the search space, the approach suggested by [Khuri et al., 1994] is to allow infeasible strings to join the population. A penalty term reduces the fitness of infeasible strings. The farther away from feasibility, the higher the penalty term of a string. Thus, the following fitness function was defined (s is the number of over-filled knapsacks):

$$f(x) = \sum_{i=1}^{n} p_i x_i - s \times max(p_i) \tag{1}$$

Hence, the fitness function uses a graded penalty term $max(p_i)$. The number of times this term is subtracted from the fitness of a infeasible solution is equal to the number of over-filled knapsacks that the solution produces.

A Multiple Knapsack instance, taken from the literature (termed Weish 30), was used as test problem. It has 90 objects and 5 sacks. This (and several other) problems are available online from the OR-library by [Beasley, 1990]. Weish 30 is among the biggest and more complex Multiple Knapsack instances available in the library.

3 METHODS

For estimating optimal mutation rates in GAs we need to define what an optimal or near-optimal mutation rate is. The working definition used here is: an optimal mutation rate is that producing optimal performance. But then, we need a good way of measuring GA performance. Given the randomized nature of GAs, conclusions can never be drawn from a single run. Instead, the common practice is to consider statistics from a sufficiently large number of independent runs. So, the standard performance measures for GAs are the average and best fitness values attained after a prefixed termination criterion, averaged over several runs. Within a given run, the best fitness could be either the current best in the population, or the best fitness attained so far. These measures are considered after a fixed termination criterion, or over fixed intervals throughout the GA run. For the experiments in this paper, we will consider the best fitness attained so far after a fixed termination criterion. This criterion will be carefully selected in each case to be long enough to stabilize the best and average fitness of the population. The average of several runs will be considered (typically 50) and the standard deviation will be shown in most cases. The first empirical section, however, studies the time dependency of the mutation rate. In this case best-so-far fitness values are reported at fixed intervals.

To study the applicability of the $1/L$ heuristic, we explore the effect of modifying some relevant evolutionary parameters on the magnitude of optimal mutation rates. Specifically, we explore the effects of modifying the selection pressure and population size. Unless otherwise stated, experiments use a generational GA with tournament selection (tournament size = 2), a population of 100 members, and both mutation and recombination (two-point with a rate of 1.0). Table 1 summarizes these default settings. Further details on the experiments and departures from the default settings are given in the respective results subsections.

Population replacement	Generational
Selection scheme	Tournament (T. Size = 2)
Population size	100
Recombination rate	1.0
Recombination operator	Two-point
Termination criterion	2,000 Generations
Number of runs	50

Table 1: GA default parameters used in the experiments.

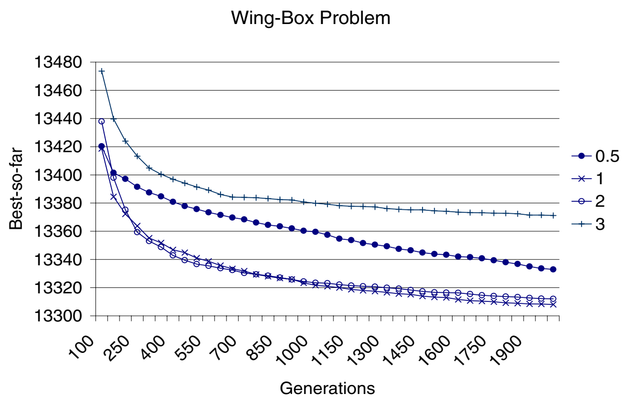
4 RESULTS

Three groups of experiments were performed with the aim of exploring: (i) the time-dependency of the mutation rate, (ii) the effect of modifying the selection pressure, and (iii) the effect of modifying the population size. Experiments were run on both test problems (Wing-Box and Knapsack). For analyzing the results, it is worth remembering that the Wing-Box is a minimization problem whereas the Knapsack is a maximization problem.

4.1 TIME-DEPENDENCY

The first set of experiments studies the behavior of different mutation values over the generations of a GA run. The evolutionary parameters used are those summarized in Table 1. Results are presented in three stages. First the "interesting" part of the search, from generation 100 to generation 2,000 (Figure 3). Then, the first stage of the search, the first 100 generations (Figure 4); and, finally, the last stages of the search, from generation 2,000 to 5,000 (Figure 5). The plots show the average best-so-far fitness attained over fixed intervals throughout the GA run on both test problems (the Wing-Box and Knapsack problems). Four mutation values were explored: 0.5, 1.0, 2.0, and 3.0 mutations per genotype. Standard deviations are not shown in these plots for the sake of clarity.

For the intermediate stage of the search, on the Wing-box problem the mutation rates of $1/L$ and $2/L$ produced the best results and performed similarly (Figure 3, top). On the Knapsack problem, a mutation rate of $1/L$ seems to produce the best performance in this stage (Figure 3, bottom).



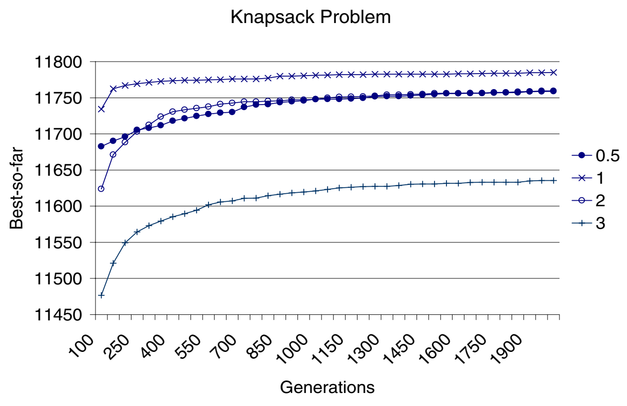


Figure 3: Comparing the performance of different mutation rates over a GA run on both test problem. The curves show the average best-so-far fitness over fixed intervals throughout the GA run for various mutation rates (expressed as mutations per genotype).

The initial stage of the search is rather similar for both test problems (Figure 4, recall that the Wing-Box problem is a minimization problems whereas the Multiple Knapsack is a maximization problem) . All the mutation values explored performed similarly. However, the mutation values of 0.5 and 1.0 mutations per genotype seem to produce the best results in this stage.

Again, the final stage of the search is rather similar for both test problems. A mutation rate of $1/L$ produced the best performance in both cases (Figure 5).

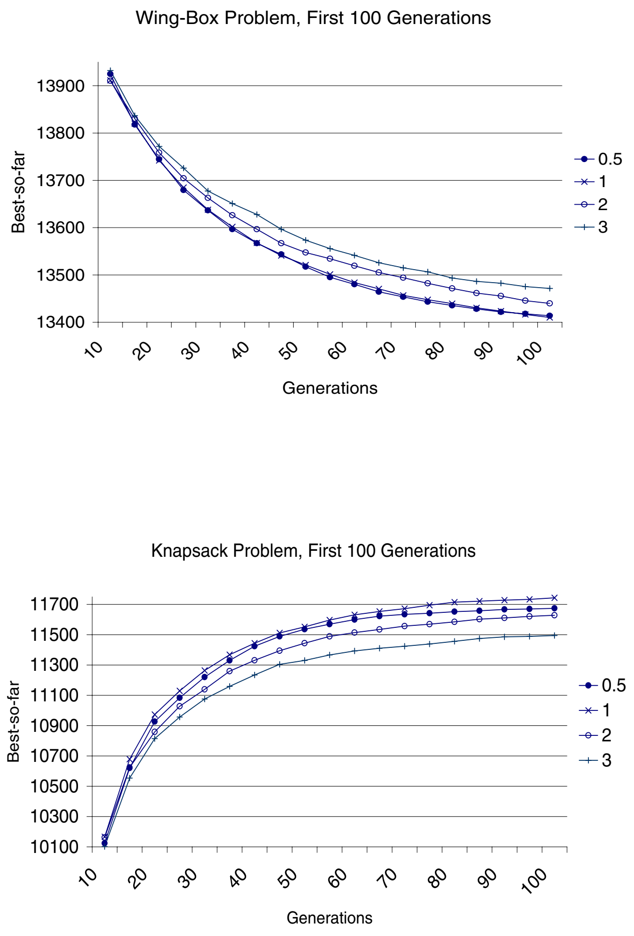


Figure 4: Comparing the performance of different mutation rates over a GA run on both test problems for the first 100 generations. The curves show the average best-so-far fitness over fixed intervals throughout the GA run for various mutation rates (expressed as mutations per genotype).

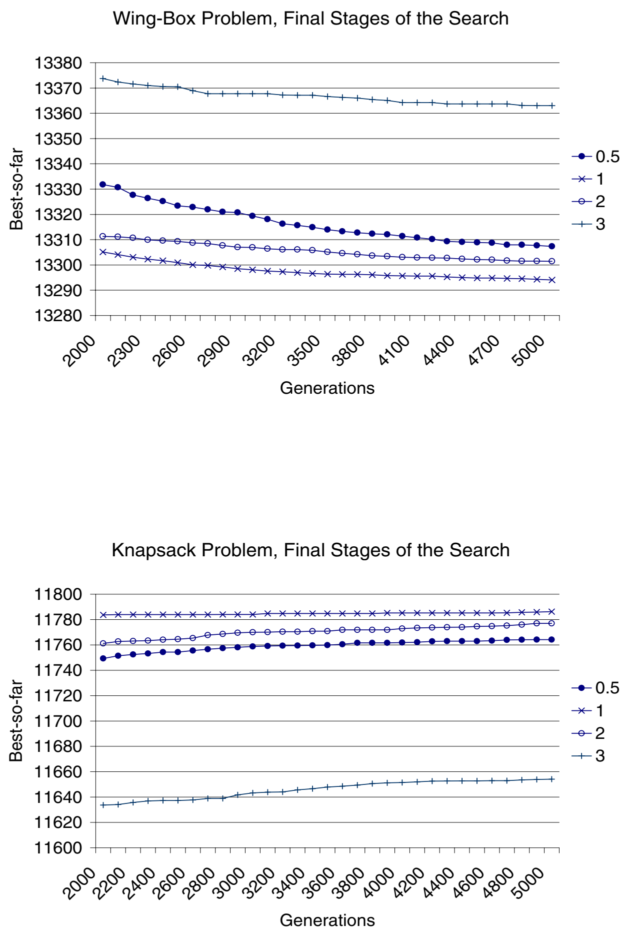
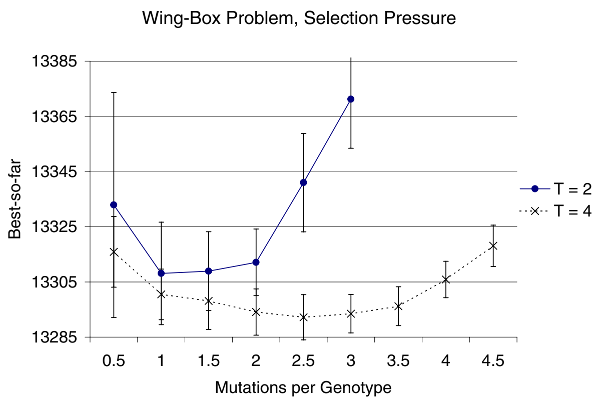


Figure 5: Comparing the performance of different mutation rates over a GA run on both test problems for the final stages of the search (from generation 2,000 until 5,000). The curves show the average best-so-far fitness over fixed intervals throughout the GA run for various mutation rates (expressed as mutations per genotype).

4.2 SELECTION PRESSURE

This subsection explores the effect of increasing the selection pressure on the magnitude of optimal mutation rates. The experiments use tournament selection because this scheme allows the selection pressure to be explicitly controlled. A common tournament size is 2, but selection pressure increases steadily for growing tournament sizes. Two tournament sizes, 2 and 4, were tested. Additionally, on the Knapsack problem, results using proportional selection are also presented for the sake of comparison. Figure 6 compares optimal mutation rates (per genotype) on the two selected test problems. The strength of selection had a noticeable effect on the magnitude of optimal mutation rates: on the Wing-box problem and for a tournament size of 2, the optimal mutation rate was around $1.0 - 2.0$ mutations per genotype, whereas for a tournament size of 4 it was around $2.5 - 3.0$ mutations per genotype. Similarly, on the Knapsack problem the optimal mutation values were around $1.5/L$ for tournament size of 2; and around $2.0 - 3.0$ for tournament size of 4. Moreover, the curve using proportional selection on the Knapsack problem (Figure 6, bottom), strikingly shows the difference in magnitude of optimal mutation rates for a weak selection pressure. In this case, the optimal mutation rate was as low as 0.05 mutations per genotype.



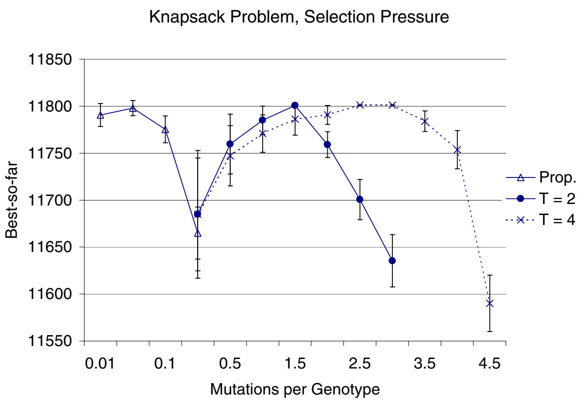


Figure 6: Comparing optimal mutation rates (per genotype) for different selection pressures on the two test problems. Tournament selection with two tournament sizes (2 and 4) was tested. Additionally, proportional selection was tested on Knapsack problem. The curves show the average best-so-far fitness attained after 2,000 generations for various mutation rates.

4.3 POPULATION SIZE

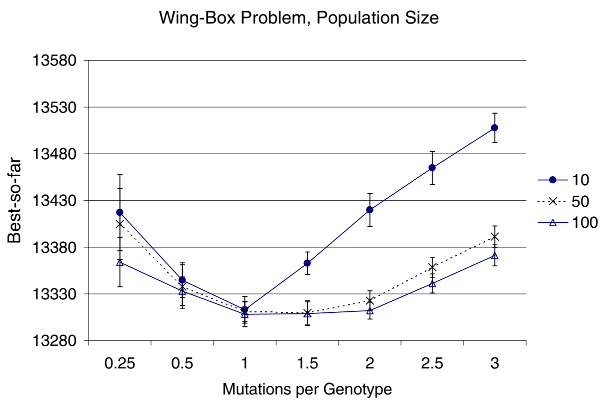
This subsection explores the effect of modifying the population size on the magnitude of optimal mutation rates. Three population sizes: 10, 50, and 100, were tested. The number of generations used as a stop criterion varied according to the population size since the smaller the population, the more generations were needed for equilibrating the best-so-far fitness. So the termination criteria used were 20,000, 4,000, and 2,000 generations for population sizes 10, 50, and 100 respectively. Figure 7 shows results on the two selected test problems. Optimal mutation rates tended to be smaller, the smaller the population size, this tendency was clearer on the Knapsack problem (bottom plot), where optimal mutation rates were around $0.5 - 1.0/L$ for a population size of 10, and around $1.0 - 1.5/L$ for population sizes of 50 and 100. Notice that for population sizes of 50 and 100, differences in performance for the various mutation rates tend to stabilize. This was also the case for preliminary experiments on larger populations.

5 DISCUSSION

This paper has been a first attempt to explore the validity of the heuristic suggesting a mutation rate of $1/L$ for GAs with bit-string encoding. Two completely unrelated and complex real-world domains were selected as test problems. Some common behaviors were found, so these findings may convey some generality. Three aspects were studied: (i) the time-dependency of the mutation rate, and the effect (on the magnitude of optimal mutation rates) of modifying (ii) the selection pressure, and (iii) the population size. Our main results are summarized below:

Time-Dependency: It has been suggested elsewhere that mutation rates should not be constant, but should decrease over the GA run. Results in this paper, however, suggest that a mutation rate of $1/L$ will produce optimal or near optimal results throughout the whole search process. So, on the specific but rather standard GA settings used here (generational GA, population size of 100, two-point recombination with a rate of 1.0, tournament selection of size 2, best-so-far fitness as performance measure), a constant mutation regime with a rate of $1/L$ would produce very competitive results.

Selection pressure: The strength of selection had a pronounced effect on optimal mutation rates. The stronger the selection pressure, the higher the magnitude of optimal mutation rates. The use of proportional selection (where there is no control over the selection pressure) may produce much smaller optimal mutation rates as compared to tournament selection. An interesting observation is that for tournament selection with tournament size of 2 (and a population of size 100), optimal mutation rates occurred between 1.0 and 2.0 mutations per genotype, whereas for tournament size of 4 they increased to 2.5 - 3.0 mutations per genotype (Figure 6). This result suggests that se-



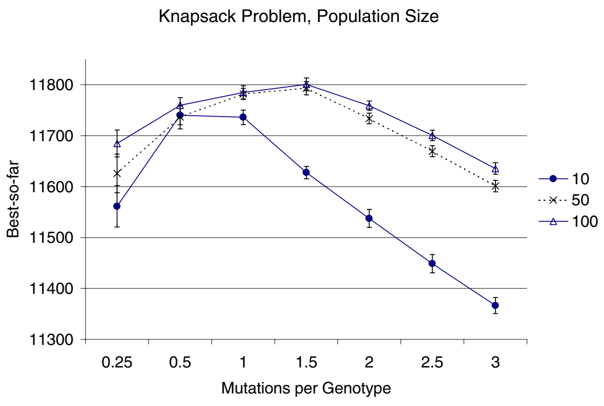


Figure 7: Comparing optimal mutation rates for various population sizes (see legends) on both test problems. The curves show the average best-so-far fitness attained after a fixed number of generations for various mutation rates. These fixed number of generations varied according the population size (20,000, 4,000 and 2,000 generations for population sizes 10, 50, and 100 respectively).

lection pressure is an important component in determining the magnitude of optimal mutation rates.

Population size: The effect of population size on the magnitude of optimal mutation rates was not found to be marked. However, the evidence suggests that optimal mutation rates are smaller, the smaller the population size. These differences in the magnitude of optimal mutation rates tend to stabilize for population sizes of 50 and larger.

6 CONCLUSION

It is very difficult to suggest general principles for setting evolutionary parameters. The evidence gathered in this paper, however, suggest that for a controlled selection pressure (tournament selection, with tournament size of 2), a mutation rate of $1/L$ throughout the whole GA run, will be a good setting, producing optimal or near-optimal results. In general, we suggest that mutation rates should be expressed as mutations per genotype instead of as mutations per bit.

The heuristic of setting a mutation rate of one mutation per genotype ($1/L$) has been proposed before within the evolutionary computation community. However, results in this paper set bounds to the validity of this heuristic. A mutation rate of $1/L$ would be sub-optimal in the following cases:

- a weak selection pressure,
- an excessively high selection pressure, and
- a very small population.

ACKNOWLEDGEMENTS

Many thanks to the anonymous GECCO reviewers for their useful comments, insight and critical reading.

References

[Bäck, 1991] Bäck, T. (1991). Self-adaptation in genetic algorithms. In Varela, F. J. and Bourgine, P., editors, *Proceedings of the First European Conference on Artificial Life*. MIT Press, Cambridge, MA.

[Bäck, 1992] Bäck, T. (1992). The interaction of mutation rate, selection, and self-adaption within a genetic algorithm. In und R. Manderick, B. M., editor, *Parallel Problem Solving from Nature 2*. North-Holland.

[Bäck, 1993] Bäck, T. (1993). Optimal mutation rates in genetic search. In Forrest, S., editor, *Proceedings of the 5th ICGA*. Morgan Kaufmann.

[Bäck, 1996] Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. The Clarendon Press Oxford University Press. Evolution strategies, evolutionary programming, genetic algorithms.

[Bäck and Khuri, 1994] Bäck, T. and Khuri, S. (1994). An evolutionary heuristic for the maximum independent set problem. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 531–535. IEEE Press.

[Beasley, 1990] Beasley, J. E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072. Also available at http://www.ms.ic.ac.uk/info.html.

[Bremerman et al., 1966] Bremerman, H., Rogson, M., and Salaff, S. (1966). Global properties of evolution processes. In *Natural Automata and Useful Simulations*, pages 3–41. Spartan.

[Davis, 1989] Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, George Mason University. Morgan Kaufmann.

[DeJong, 1975] DeJong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor, MI. Dissertation Abstracts International 36(10), 5140B, University Microfilms Number 76-9381.

[Fogarty, 1989] Fogarty, T. C. (1989). Varying the probability of mutation in the genetic algorithm. In Schaffer, J. D., editor, *Proceedings of the 3rd ICGA.* Morgan Kaufmann.

[Hesser and Männer, 1991] Hesser, J. and Männer, R. (1991). Towards an optimal mutation probability for genetic algorithms. In Schwefel, H.-P. and Männer, R., editors, *Parallel Problem Solving from Nature.* Springer-Verlag, Lecture Notes in Computer Science Vol. 496.

[Julstrom, 1995] Julstrom, B. A. (1995). What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. In Eshelman, L. J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 81–87, San Francisco, CA. Morgan Kaufmann.

[Khuri and Bäck, 1994] Khuri, S. and Bäck, T. (1994). An evolutionary heuristic for the minimum vertex cover problem. In Hopf, J., editor, *Genetic Algorithms within the Framework of Evolutionary Computation*, pages 86–90, Saarbrücken, Germany. Max-Planck-Institut für Informatik.

[Khuri et al., 1994] Khuri, S., Bäck, T., and Heitkötter, J. (1994). The zero/one multiple knapsack problem and genetic algorithms. In Deaton, E., Oppenheim, D., Urban, J., and Berghel, H., editors, *Proceedings of the 1994 ACM Symposium of Applied Computation*, pages 188–193. ACM Press.

[McIlhagga et al., 1996] McIlhagga, M., Husbands, P., and Ives, R. (1996). A comparison of search techniques on a wing-box optimisation problem. *Lecture Notes in Computer Science*, 1141.

[Mühlenbein, 1992] Mühlenbein, H. (1992). How genetic algorithms really work: I. mutation and hillclimbing. In Männer, B. and Manderick, R., editors, *Parallel Problem Solving from Nature 2.* North-Holland.

[Ochoa et al., 1999] Ochoa, G., Harvey, I., and Buxton, H. (1999). Error thresholds and their relation to optimal mutation rates. In Floreano, J., Nicoud, D., and Mondada, F., editors, *Proceedings of the Fifth European Conference on Artificial Life (ECAL'99).* Springer-Verlag.

[Ochoa et al., 2000] Ochoa, G., Harvey, I., and Buxton, H. (2000). Optimal mutation rates and selection pressure in genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 315–322.

[Schaffer et al., 1989] Schaffer, J., Caruana, R., Eshelman, L., and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer, J. D., editor, *Proceedings of the 3rd ICGA*, San Mateo CA. Morgan Kaufmann.

[Smith and Fogarty, 1996] Smith, J. E. and Fogarty, T. C. (1996). Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 318–323, New York. IEEE Press.

[Tuson and Ross, 1998] Tuson, A. and Ross, P. (1998). Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6(2):161–184.