International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:7, 2007

The Rank-scaled Mutation Rate for Genetic Algorithms

Mike Sewell, Jagath Samarabandu, Ranga Rodrigo, and Kenneth McIsaac

***Abstract*—** A novel method of individual level adaptive mutation rate control called the rank-scaled mutation rate for genetic algorithms is introduced. The rank-scaled mutation rate controlled genetic algorithm varies the mutation parameters based on the rank of each individual within the population. Thereby the distribution of the fitness of the papulation is taken into consideration in forming the new mutation rates. The best fit mutate at the lowest rate and the least fit mutate at the highest rate. The complexity of the algorithm is of the order of an individual adaptation scheme and is lower than that of a self-adaptation scheme. The proposed algorithm is tested on two common problems, namely, numerical optimization of a function and the traveling salesman problem. The results show that the proposed algorithm outperforms both the fixed and deterministic mutation rate schemes. It is best suited for problems with several local optimum solutions without a high demand for excessive mutation rates.

***Keywords*—** Genetic algorithms, mutation rate control, adaptive mutation.

I. Introduction

GENETIC algorithms are computational models that mimic the evolutionary process [1], [2], [3]. Small genetic 'mistakes' or mutations give rise to genetic traits which are not present in the overall population [2]. The near-optimum mutation rate is problem domain dependant and is usually found by trial and error [4], [5], [6]. Successful approaches of applying genetic algorithms to solve real-world problems with fixed mutation rates are numerous (e.g. Zhang *et al.* [7], Lutton and Véhel [8]). The use of such a fixed parameter throughout the run is called parameter tuning and differs from parameter control. Controlling the parameters optimize the algorithm to the problem. Interested readers are referred to the surveys by Eiben *et al.* [9] and by Angeline [10] which discuss the issues surrounding parameter control and parameter tuning. There are three classes of methods of controlling the mutation rate (or any other parameter), namely, deterministic, adaptive and self-adaptive parameter control. The first class, deterministic control, changes the mutation rate with time according to some deterministic rule with no feedback from the search [4], [5], [6]. The deterministic rule is devised as a function of time (or evolution or generation) based on some heuristics. It can also be dependent on one or more parameters such as the string length, the population size and the total number of generations [5]. While most of these schemes vary the mutation rate based on time, there are others based on quantities such as distance to the optimum [11]. The second class, adaptive parameter control, on the other hand uses feedback from the search to modify the parameters [12]. The third class, self-adaptive parameter control uses a 'meta-evolution' scheme where the parameters themselves are evolved along with the search variables.

Manuscript received December 23, 2005.

M. Sewell is with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON., N6A 5B9, Canada. (e-mail: msewell@canada.com).

J. Samarabandu is with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON., N6A 5B9, Canada (e-mail: jagath@uwo.ca).

R. Rodrigo is with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON., N6A 5B9, Canada (phone: 519-661-2111 ext. 87615; fax: 519-850-2436; e-mail: brodrigo@uwo.ca).

K. McIsaac is with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON., N6A 5B9, Canada (e-mail: kmcisaac@engga.uwo.ca).

Adaptive parameter control is preferred due to its adaptability to the search and the controllability, taking the intrinsic dynamic and adaptive nature of a genetic algorithm run [9], [13]. A classic example is the 1/5 rule of Rechenberg (see Eiben [9] for details) which adapts the mutation step size depending on the frequency of successful mutations. Mutation survival rate is also exploited by Aguirre and Tanaka [14] by reducing the mutation rate by a coefficient each time the survival rate falls under a predefined constant. Similar approaches of varying mutation rate are found in Uyar [15] and Acan [16]. Djurisic *et al.* [17] adapt the mutation rate depending on the average of rates of the current generation and narrowing the range of the of mutation rate using a predetermined parameter.

The adaptive parameter control schemes cited above tend to vary the mutation rate based on the frequency of successful mutants. The distribution of the fitness of the current generation is not directly exploited in coming up with the new mutation rate. This concept can be characterized by the scope of the mutation parameter. The scope refers to the level at which the change is made: population level or individual level. If the scope is population level, as in the cited schemes above, the fitness distribution is disregarded. Altering the parent solution guided by a probability model to generate new offspring has been successfully applied for the maximum clique problem [18]. The purpose of the probability model is to characterize the promising area in the solution space while the similarity between the offspring and the parents are controlled. It is important to note that both feasible and infeasible solutions can affect the algorithm [19]. The feasibility of the offspring of the set of feasible solutions should be maintained to explore the promising area. On the other hand a compromise should be made between the the exploration of the whole space and eliminating the potentially harmful infeasible individuals.

The adaptive mutation approach presented in this paper makes use of the knowledge of the fitness distribution of the current generation to adapt the individual mutation rates.

The metric used is the fitness rank of each individual. The rank refers to how fit an individual is as determined by the evaluation function. The mutation rate is made a function of the fitness rank. Consequently, the mutation rate is lowest for the best fit and highest for the least fit. This can be used to vary the degree of retaining feasible solutions and eliminating infeasible solutions. The scope of the algorithm is therefore at the individual level. As a result the complexity of the proposed scheme is much lower than that of a self adaptive scheme and is of the order of an individual adaptation scheme.

II. Adaptive Mutation Control

There are three methods of adapting the mutation rate as indicated in the Section I, namely, deterministic, adaptive and self-adaptive control. Deterministic mutation control changes the mutation rate with generation or time according to some heuristic rule. An example of a deterministic scheme is the one in Bäck and Schüts [5]. The individuals are represented as a fixed length binary vector $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \{0,1\}^n$ and the time dependent mutation rate is $p_t = (a + b \cdot t)^{-1}$ with initial rate $p_0 = 1/2$ and final rate $p_{T-1} = 1/n$. t is the time, epoch or the generation. The resulting function is

$$p_t = \left(2 + \frac{n-1}{T-1}t\right)^{-1} \quad . \tag{1}$$

More examples are found in Thierens [12] and Eiben [9]. The mutation rate is changed deterministically over time and is independent of any feedback from the search. The question is whether the deterministic rule of the fixation of parameters of the mutation rate is the optimum for the algorithm or not. The alternative is to use feedback from the search itself.

Methods with adaptive mutation rate use feedback from the search process to to modify the mutation rate. For example, the well known Rechenberg's "1/5 success rule" (see Eiben [9] for details) aims at maintaining at least 1/5 of the mutants being successful. If t is the generation, n represents the frequency of mutation rate change and p_s is the relative frequency of successful mutation, the mutation rate $\sigma(t)$ is found as
if $(t \bmod n = 0)$ then

$$\sigma(t) = \begin{cases} \sigma(t-n)/c & if p_s > 1/5 \\ \sigma(t-n) \cdot c & if p_s < 1/5 \\ \sigma(t-n) & if p_s = 1/5 \end{cases} \tag{2}$$

else

$$\sigma(t) = \sigma(t-1) \tag{3}$$

$c < 1$ is a constant close to 1. These schemes ensure that fit mutants are retained. The proposed scheme refines the retention property to the extent that the fitness of the individuals is taken into account in the process of mutation.

III. Rank-scaled Mutation Control Algorithm

The method for mutation selection being proposed here is the "rank-scaled mutation rate". This method involves increasing the chance that the solutions mutate depending on their viability. The solutions become more likely to mutate as their viability decreases. The optimum solutions are less likely to mutate, allowing them to recombine to produce better offspring. The least viable solutions are more likely to mutate, inserting new codes into the population. This method has an advantage in the way that since the least viable solutions are more likely to mutate, the system has the ability to explore potential solutions other than the solution that the "optimum" solution is converging on. This would make it less likely to converge to a local maximum and more likely to converge to a global maximum.

IV. Implementation Details

The implementation is carried out to reflect the essence of various forms of mutation rates. The problems selected for testing the algorithm are optimization of a function and the traveling salesman problem (TSP). TSP favors higher mutation rates than the function optimization problem. It will be shown in results (Section V) that the proposed algorithm performs well with problems requiring low mutation rates. As the implementation slightly differs depending on the problem, the section outlines the details.

A. Traveling Salesman Problem

Traveling salesman problem (TSP) is to find the most efficient path to visit each of n cities, passing through a city only once. Finding the optimum path in the TSP is done, in general, by swapping cities in the processes of crossover and mutation. In particular, mutation takes place once two cities selected at random in the current route of a random salesman is swapped. The random selection of a salesman is tied to the rank of the salesman. A predefined number (elite count) of best ranking salesman is retained. A salesman is selected to be mutated at random in the following fashion. If $x \in [0,1]$ is a random variable the probability of mutation p_m is

$$p_m = P\left(x < b + k\frac{i}{N}\right) \tag{4}$$

where i is the rank, N is the population size and k is a small positive integer tuned to increase the mutation dependency on the rank. b is used to act as a bias. If k is large, then the probability of mutation is increasingly high if the rank i if the individual is high. We selected $k = 3$ in our experiments and as a result the mutation tendency is kept at a higher rate.

B. Optimizing a Curve

The objective is to find the global maximum (or minimum) of a function. That is to find

$$x^\star = \arg\max_x f(x) \tag{5}$$

subject to some constraints. However, we follow a slightly different approach in optimizing the function. Both the domain x and the value $y = f(x)$ are evolved. The function selected in this example is $f(x) = \sin(x/3) \times x^2 + 10$ where $x \in [0, 50]$. Each individual contains both x and y and they are initialized randomly. Single point cross over is used and the elitism count is 1. Based on the predefined crossover point, each of x^t and y^t at time t can be expressed as $x^t = x^t_U + x^t_L$

and $y^t = y_U^t + y_L^t$, respectively. Rank scaled mutation is carried out depending on a random number $\alpha \in \{0, 1, \ldots, N\}$ related to the mutation rate σ and a random number $\beta \in \{1, 2, \ldots, 8\}$ as shown below.

$$\text{if} \quad \alpha - i \times 3 < c, \quad x_U^{t+1} = \begin{cases} x_U^t - 1 & \text{if } \beta = 1 \\ x_L^t + 1 & \text{if } \beta = 2 \\ x_L^t - 1 & \text{if } \beta = 3 \\ x_U^t & \text{otherwise} \end{cases} \qquad (6)$$

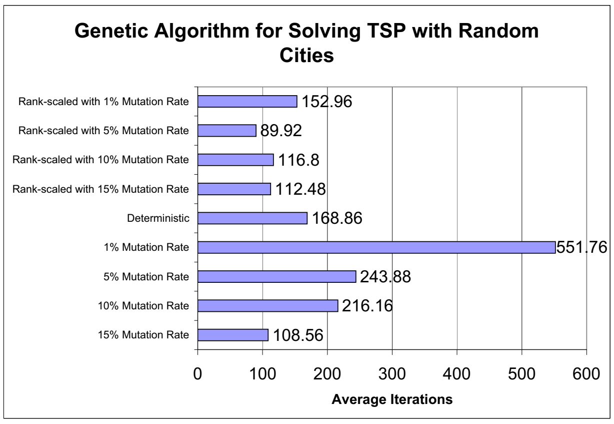
Here i is the rank of the individual in the current population and since the probability of mutation is controlled by i, the scheme is a rank-scaled mutation scheme. c is a constant that acts as a threshold controlling the mutation rate. The values of x_L^{t+1},y_L^{t+1} and y_U^{t+1} are also found in a similar fashion depending on ϕ.

V. Experiments and Results

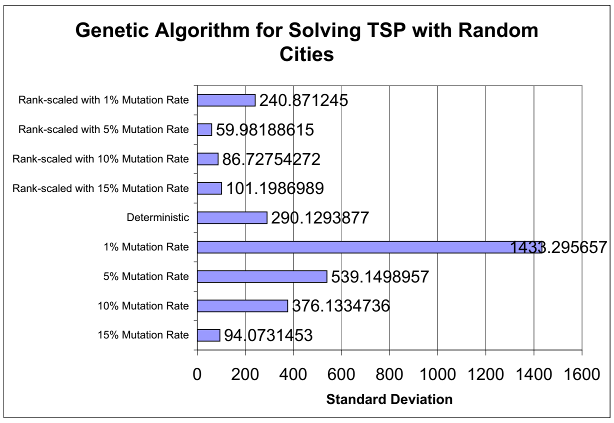
Experiments were carried out to solve the TSP and to find the global maximum of a function. Performance of the proposed rank-scaled mutation rate is compared against a standard GA with a set of fixed mutation rates as well as using a GA with deterministic mutation rate. As can be seen from Figure 1, for the TSP implementation the rank-scaled method is superior to both the set and deterministic mutation rate methods, for both the cases of fixed cities and random cities. This is most likely due to the nature of the method of mutation. Since two random cities are swapped, this creates the possibility of having a large difference prior to and post mutation, and a much smaller possibility of having minute decrements in path length. This type of system equally favors the recombination and mutation methods for approaching the optimum solution. This type of problem also has several local optimum solutions for the algorithm to become confused by, whereas the curve fitting problem only has two. This type of problem therefore favors the strengths of the rank-scaled mutation rates, which is having the greater likelihood of finding the global optimum. On the other hand, in the curve fitting problem, as the system approaches the optimum solution, small increments and decrements of solutions are favored. The method that was chosen for mutation was to add or subtract from the coordinates, 50% of the time at the decimal level. Therefore, the recombination stage for this problem is only effective for the first few iterations, after which mutation is favored. Since the deterministic method, as it approaches the optimum solution, is more likely to mutate, it is more likely to quickly converge than the other methods.

In the first implementation of the rank-scaled mutation rate, the random city TSP, we can see that the rank-scaled mutation rate with a rate of 5% was the optimum solution, showing an average of 89.9 iterations before convergence (Figure 1(a)). This beat the next best solution, the deterministic mutation rate, by 17%, and the best set mutation rate by 46%. The rank-scaled mutation rate also had the smallest standard deviation of 59.98 iterations, ensuring the most consistency between implementations (Figure 1(b)).

In the second implementation of the rank-scaled mutation rate, the fixed city TSP, we can see that the rank-scaled



(a) Average iteration

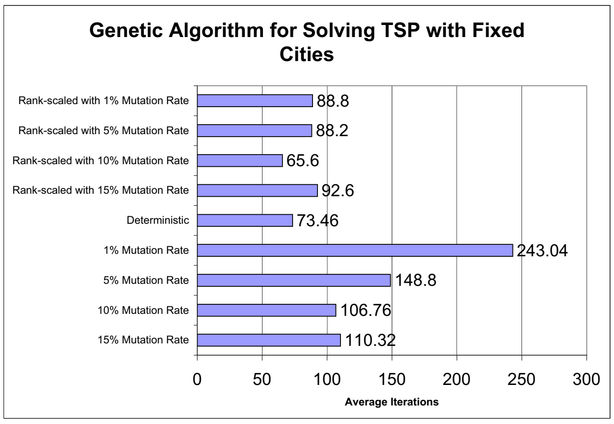


(b) Standard deviation

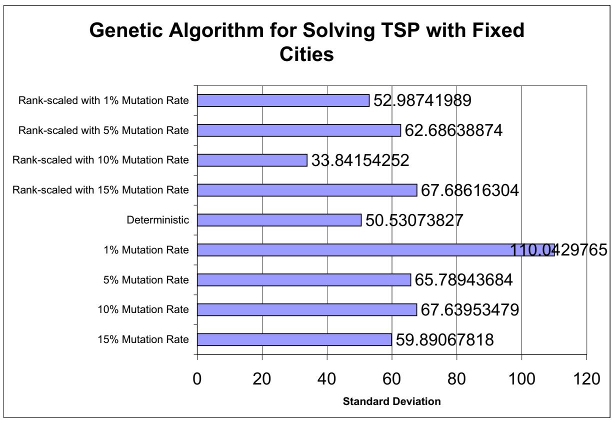
Fig. 1. Traveling salesman problem with random cities

mutation rate with a rate of 10% was the optimum solution, showing an average of 65.6 iterations before convergence. (Figure 2(a)) This beat the next best solution, the deterministic mutation rate, by 10%, and the best set mutation rate by 38.5%. The rank-scaled mutation rate also had the smallest standard deviation of 33.8 iterations, ensuring the most consistency between implementations. (Figure 2(b))

For the third implementation, the rank-scaled mutation rate came in second to the deterministic mutation rate. The rank-scaled mutation rate at 10% had 3663.08 average iterations, while the deterministic mutation rate had 2517.98, beating it by 45.47%. The rank-scaled mutation rate still beat the nearest set mutation rate by 21.56%. (Figure 3(a)) The deterministic mutation rate also had the smallest standard deviation of

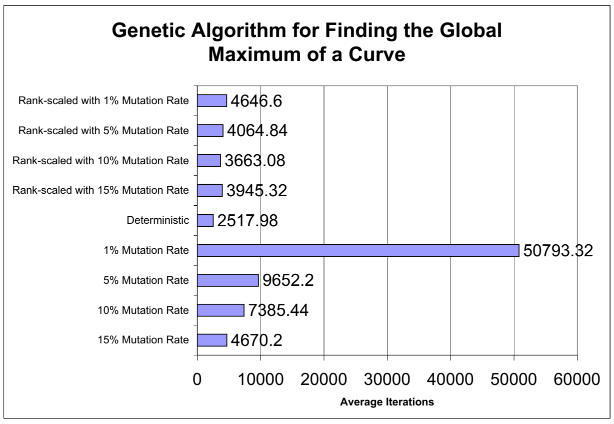


(a) Average iteration

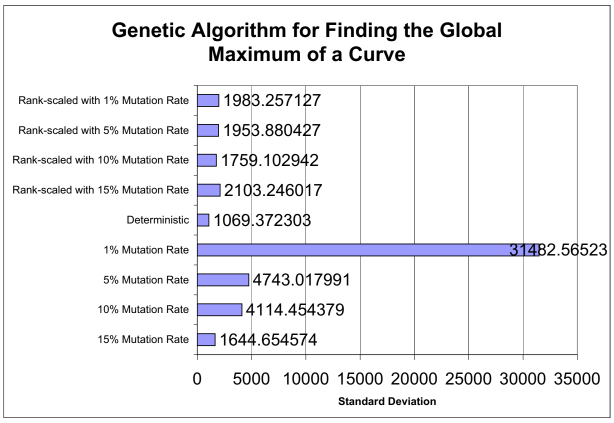


(b) Standard deviation

Fig. 2. Finding global maximum of a curve



(a) Average iteration



(b) Standard deviation

Fig. 3. Finding global maximum of a curve

1069.37 iterations. (Figure 3(b))

VI. Conclusion

This paper proposed an individual level adaptive mutation scheme called rank-scaled mutation. In this scheme the mutation rate is adaptively changed depending on the rank of each individual. Overall, the rank scaled mutation rate was found to be an effective method of mutating a population of solutions in a genetic algorithm. Although not as effective in problems where mutation is favored, it is extremely effective in problems with several local optimum solutions, beating its nearest competitor by 10% – 17% fewer iterations.

A. Improvements

The implementation assumes a linear rank scaled mutation scheme. Further refinements can be introduced if the mutation rate is defined as a function of the rank. Assume that an individual $i(= 1, 2, \ldots, n)$ where n is the current population size less the elicit count, is represented by x_i. Ranks 1 to n are assigned to each individual depending on the fitness as the best fit individual assumes rank 1 and the least fit assumes rank n. The mutation rate for the individual i is found as

$$p_{mi} = \beta \times \gamma \times \sigma(0, 1) \tag{7}$$

where

$$\gamma = \left(\frac{p_{max} - p_{min}}{n} \times i \right)^{\alpha} \tag{8}$$

β is a constant used to tune the algorithm and $\sigma(0,1)$ is the zero mean Gaussian random variable with variance 1. α is a parameter which allows selection of the nature of the rank scaling. For example $\alpha = 1$ yields a linear scaling scheme and so on. Additional control is available through the lower and upper bounds p_{max} and p_{min} respectively.

REFERENCES

[1] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, pp. 65–85, 1994.

[2] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3–17, April 1997.

[3] X. Yao, "Evolutionary computation," in *Evolutionary Optimization*, R. Sarker, M. Mohammadian, and X. Yao, Eds. Kluwer Academic Publishers, 2002, ch. 2, pp. 27–53.

[4] T. Bäck, "Optimal mutation rates in genetic search," in *Proceedings of the 5th International Conference on Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA, USA: Morgan Kaufmann, 1993, pp. 2–8.

[5] T. Bäck and M. Schütz, "Intelligent mutation rate control in canonical genetic algorithms," in *Proceedings of the Ninth International Symposium on Foundations of Intelligent Systems*, ser. LNAI, Z. W. Rás and M. Michalewicz, Eds., vol. 1079. Berlin: Springer, June 1996, pp. 158–167.

[6] G. Ochoa, I. Harvey, and H. Buxton, "On recombination and optimal mutation rates," in *Proceedings of the Genetic and Evolutionary Computation Conference*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. 1. Orlando, FL: Morgan Kaufmann, 1999, pp. 488–495.

[7] F. Zhang, Y. F. Zhang, and A. Y. C. Nee, "Using genetic algorithms in process planning for job shop machining," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 4, pp. 278–289, November 1997.

[8] E. Lutton and J. L. Véhel, "Hölder functions and deception of genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 2, pp. 56–71, July 1998.

[9] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124 – 141, July 1999.

[10] P. J. Angeline, "Adaptive and self-adaptive evolutionary computations," in *Computational Intelligence: A Dynamic Systems Perspective*, M. Palaniswami and Y. Attikiouzel, Eds. IEEE Press, 1995, pp. 152–163.

[11] T. Bäck, "The interaction of mutation rate, selection, and self-adaption within a genetic algorithm," in *Parallel Problem Solving from Nature, 2: Proceedings of the Second Conference on Parallel Problem Solving from nature*. Brussels: North-Holland, 1992, pp. 85–94.

[12] D. Thierens, "Adaptive mutation rate control schemes in genetic algorithms," in *Proceedings of the 2002 Congress on Evolutionary Computation CEC '02*, vol. 1. Honolulu, HI: IEEE, 2002, pp. 980–985.

[13] A. Tuson and P. Ross, "Adapting operator settings in genetic algorithms," *Evolutionary Computation*, vol. 6, no. 2, pp. 161–184, 1998.

[14] H. E. Aguirre and K. Tanaka, "Genetic algorithms on nk-landscapes: Effects of selection, drift, mutation, and recombination," *Lecture Notes in Computer Science*, vol. 2611, pp. 131–142, January 2003.

[15] S. Uyar, S. Sariel, and G. Eryigit, "A gene based adaptive mutation strategy for genetic algorithms," *Lecture Notes in Computer Science*, vol. 3103, pp. 271–281, January 2004.

[16] A. Acan, "Mutation multiplicity in a panmictic two-strategy genetic algorithm," *Lecture Notes in Computer Science*, vol. 3004, pp. 1–10, January 2004.

[17] A. B. Djurisic, A. D. Rakic, E. H. Li, M. L. Majewski, N. Bundaleski, and B. V. Stanic, "Continuous optimization using elite genetic algorithms with adaptive mutations," *Lecture Notes in Computer Science*, vol. 1585, pp. 365–372, January 1999.

[18] Q. Zhang, J. Sun, and E. Tsang, "An evolutionary algorithm with guided mutation for the maximum clique problem," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 192 – 200, April 2005.

[19] Z. Michalewicz and M. Schmidt, "Evolutionary algorithms and constrained optimization," in *Evolutionary Optimization*, R. Sarker, M. Mohammadian, and X. Yao, Eds. Kluwer Academic Publishers, 2002, ch. 3, pp. 57–86.