A Queue-Based Genetic Algorithm (QGA)

Asanobu KITAMOTO
Research and Development Department
National Center for Science Information Systems (NACSIS)
3-29-1, Otsuka, Bunkyo-ku, Tokyo 112-8640, JAPAN
TEL: +81-3-3942-8590 FAX: +81-3-5395-7064
E-mail: kitamoto@rd.nacsis.ac.jp

Abstract

This paper proposes a novel GA called "a queue-based genetic algorithm." This algorithm is unique in that the data structure of the population is based on a first-in-first-out queue, and it is the key idea to realize asynchronous structure of the algorithm. This algorithm is in particular developed in relation to interactive evolutionary computation (IEC) framework, and the characteristics of QGA comes from the solution to the problem of IEC framework. Various novel genetic operations is developed, and some of them are analyzed in a quantitative manner. Finally QGA is compared with SGA in terms of royal road test function, and behavior of the algorithm is illustrated in terms of population and fitness.

1 INTRODUCTION

Traditional genetic algorithms (GA) [1, 2] is based on a generation model, which, in practice, is used in the form of two extremes; namely a model that replaces the entire population at once (traditional GA) and that replaces only a small portion (usually less than a few individuals) of the population (steady-state GA). In this paper, the author proposes another new type of GA whose generational model is fundamentally different from those ones stated above, namely a non-generational model where all the individuals in the populations are maintained in a set of *queues*.

In fact, this idea is not totally new; a similar idea was already proposed by De Jong et al.[3] in relation to a steady-state GA. Instead of uniform random deletion scheme, they proposed an alternate deletion scheme called *FIFO deletion* to reduce the expected variance of growth curves. The advantage of FIFO deletion is that the variance in individual lifetimes is reduced to zero, and the implementation of GA is as simple as a first-in-first-out queue with new individuals added to one end and deleted individuals removed from other end. However, to the author's knowledge, the usage of a queue as a population model was not investigated so far. The fixed size population model has been popular for a long period.

However, the author's interest in a queue comes from his interest in the interactive evolutionary computation (IEC) framework. Here the IEC framework is a kind of human computer interaction mechanism to take user's preference into systems through the interactive optimization of system parameters (structures) by means of evolutionary computation algorithms. Interactivity in this case refers to the idea of relating relevance feedback given by a user to the fitness of individuals. The author has been working in this research field, and realized that asynchronous operations and rapid convergence becomes important issues [4]. To meet such requirements, a queue is a suitable data structure to maintain the population because of the reasons described later. Hence a new genetic algorithm proposed in this paper is called a queue-based genetic algorithm (QGA), although it should be emphasized that the QGA is not a specialized algorithm for IEC applications.

In the QGA, a *FIFO queue* is used to represent a structured population model instead of a flat population model by an *array*. Furthermore, the application of a queue requires a new type of algorithmic techniques on genetic operations, such as selection, reproduction, crossover and mutation. This paper mainly deals with issues on the population of individuals, not on issues on individuals, such as representation (binary / real coding, diploidy, ...)[1], bit-string recombination, and so on. The comparison of QGA with traditional GA is presented later to demonstrate the capability of QGA.

[1]Hereafter I use binary coding (gray coding) representation not to invite too much complexities in the paper, in spite of my preference for real coding on function optimization problems.

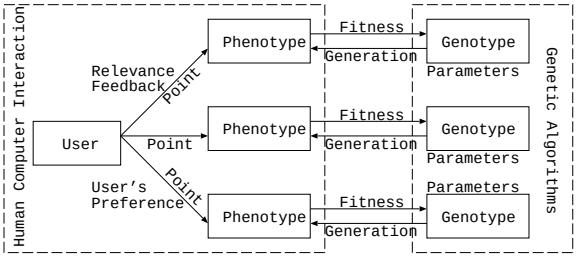


Figure 1: A schematic diagram of interactive evolutionary computation (IEC) framework.

2 BASIC MODEL OF QGA

2.1 Interactive Evolutionary Computation

Now let me briefly introduce the background and motivation to reach the idea of QGA. As addressed in the Introduction, the motivation of developing QGA comes from the author's interest in IEC framework, which refers to a human computer interaction system utilizing evolutionary computation framework to introduce the adaptive behavior of the system. In some systems, the model of user's preference or user's subjectivity play an important role, however in those cases it is extremely difficult or impossible to express the evaluation function of problems being considered. One good example is the aesthetic evaluation of computer generated images based on user's preference. Since human beings are especially good at visual task compared to computers, the recognition of beautiful images, or a rough assignment of evaluation scores to images based on some criteria is something human beings should do. On the contrary, a computer is extremely good at computation. Then the role of a computer is to accept relevance feedback given by a user, extract information from relevance information, and determine what to do next to best meet user's preference. Such complimentary relationship between human begins and computers leads to a mutual system such as IEC framework.

Thus this research area is attracting more and more attention recently. One of the early works is a "biomorph system" by Dawkins [5], in which patterns that look like creatures are evolved using mutation operations. The application of IEC framework to computer graphics are further investigated by Sims [6]. Another impressive application of IEC framework is on criminology [7]; others are evolvable line drawings [8], knowledge discovery [9], and so on. The author also has been working in this area to construct an interactive image browsing system based on a QGA [10, 4].

However, although traditional GAs can be applied to those applications, the author thinks that we need more convenient algorithm that solves problems addressed below.

2.1.1 Raw fitness value is imprecise

The schematic diagram of IEC is illustrated in Figure 1. System parameters or structures are coded in the genotype of each individual, and the genotype generates the phenotype which serves as the target of evaluation by a user. In turn, a user gives relevance feedback to each target, and this relevance feedback is fed back to genetic algorithm in which evaluation scores are related to the fitness of individuals. However, the fitness value, in this case, is actually not a precise value, since the result of relevance feedback is "subjective" in nature. We should not expect that, even if relevance feedback is given on semi-continuous scale in the interval, say $[0, 100]$, the raw evaluation score of relevance feedback is imprecise as it seems to be. In some cases, relevance feedback is given through questions like "please rate this target on five point scale," where evaluation score is given in a discrete value. It may be more convenient for a user to answer such questions like "which one you like better," by comparing two targets. In this case, the evaluation score is given only in the order of preference.

The evaluation score thus obtained is next related to the fitness of each individual. However, as we addressed above, we should again be aware that the raw evaluation score may be a nonliear approximation of the true value, so the fitness of individual also becomes an approximate value. Hence the absolute magnitude of fitness is of less importance; rather the relative order of fitness values is important. Content-based selection scheme, which will be later proposed in this paper, will remove this effect to some degree.

2.1.2 Reducing waiting time is important

Another unique issue in IEC framework is that "waiting time" plays an important role. Unlike usual evolutionary algorithms, the characteristic point of IEC framework is that a human user gets involved in the cycle of evolutionary algorithms. This involvement introduces two problems in terms of waiting time as follows.

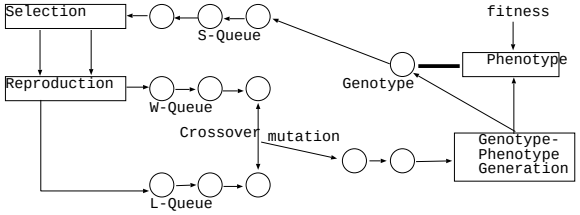


Figure 2: A schematic diagram of a queue-based genetic algorithm.

Waiting time for a human user A human user is required to wait until a computer finishes the generation of phenotype from genotype, because a human user gives relevance feedback on *phenotype* not on genotype. This generation stage sometimes requires intensive computation cost; however long waiting time invites discomfort in user interface.

Waiting time for a computer A computer (processor) is required to wait until a human user finishes relevance feedback to targets. Even if a human user can give evaluation scores at a glance, it requires at least a second or so. If a human user need to scrutinize given targets, then it requires much more time; however long waiting time brings the deterioration of computational efficiency because of the idling period of processors.

2.2 Overview of QGA

To solve the problem of mismatch in waiting time, one promising approach is to reconfigure evolutionary algorithms into an asynchronous version, where modules for a human user and for a computer can overlap to allow simultaneously processing like a pipeline processing of CPU. The introduction of a queue is a natural solution to this problem as illustrated in Figure 2. Here all individuals are maintained in a queue, and all modules can be processed simultaneously. If it is possible to remove any synchronization points and reconfigure the algorithm so that it does not require any global information on the population, then this algorithm can be easily extended to asynchronous version of GA. The characteristics of QGA can be summarized as follows:

1. Selection is based on a contest-based FIFO selection and after selection all individuals are categorized into either winners or losers.

2. In reproduction, winners have more than one offspring, while losers, less than one offspring. Selective pressure is called by two parameters.

3. In crossover, different probability is assigned for multiple types of crossover. Since the queue of winners and the queue of losers are maintained in QGA, different type of crossover plays a different role.

Thus the main data structure in QGA is a queue. Hereafter the author will explain in more detail the genetic operations proposed in this paper.

2.3 Selection

The selection operator in QGA can be called "contest-based FIFO selection." Individuals waiting for selection operation is maintained by a queue called "S-Queue (SQ)." The selection operator picks up two individuals at the head of the SQ, and compares their fitness. Then the individual with larger fitness is named as a "winner," while the other, a "loser." This selection scheme is similar to binary tournament selection in terms of fitness comparison between two individuals. However, in contrast to probabilistic selection scheme of tournament selection, all individuals are always picked up for selection in contest-based FIFO selection.

The advantage of this scheme is that it does not require any global information on the population. Except for tournamet selection schemes, traditional selection schemes require global information on the population such as the average fitness of the population, which information is required in proportionate selection schemes, or the ranking of each individual in the population, which information is required in ranking-based schemes; however among traditional schemes. The author conjectures that selection schemes relying on global information is not only inappropriate for asynchronous operation, but also inappropriate in terms of biology; instead selection schemes based on local interaction between individuals should be a more natural formulation. In the selection scheme used in the paper, selection is done by the local comparison of two individuals; therefore as soon as two individuals appear in the SQ, selection schemes can be immediately started.

Another advantage, which is a natural consequence of characteristics stated above, is that this selection scheme does not require any scaling or ranking techniques. In practice, choosing appropriate scaling or ranking

techniques is an important research issue to achieve good performance; however there appear to be no theoretical framework for choosing appropriate schemes for a particular problem [3]. Thus the removal of scaling or ranking techniques is advantageous in terms of the reduction of complexity in GA. However, this merit is obtained at the sacrifice of discarding information on absolute fitness value. This scheme may sound to be ineffective, but Whitley states as "however in many real applications the evaluation function is likely to be a heuristic that simply indicates which strings are better than others. In most cases it may not be realistic to use the value generated by the evaluation function to judge relative differences in fitness." [11] Following this argument, this scheme may not be effective as it sounds to be. Moreover, as explained in Section 2.1.1, raw fitness values are not precise in IEC applications.

Finally let me briefly describe the quantitative analysis of this contest-based selection. Let the current population size (the number of individuals in the population) be N, and the current ranking of the individual be S. Then the probability of this individual being a winner s and a loser t can be simply represented as:

$$\begin{aligned} s &= S/N \\ t &= (N-S)/N = 1-s \end{aligned} \tag{1}$$

where it is assumed that the competitor is randomly selected from the population. Equation (1) shows that a winner is not always an individual with "above average" performance. An individual with high performance has a high chance of being a winner, but an individual with low performance also has a chance of being a winner in a low-level contest.

2.4 Reproduction

Unlike traditional GAs, fitness proportionate reproduction scheme cannot be applied to QGA because of the reasons addressed above. Instead, to bias sampling toward better performing structures, the following reproduction schemes is proposed.

- Assign two offsprings to a winner with probability $p_w \geq 0$, otherwise assign one offspring.
- Assign zero offspring to a loser with probability $p_l \geq 0$, otherwise assign one offspring.

As a result, the expected number of offspring for a winner N_w and a loser N_l can be calculated as:

$$\begin{aligned} N_w &= 1 + p_l \\ N_l &= 1 - p_w \end{aligned} \tag{2}$$

In this scheme, the number of offsprings for each individual is restricted to integer value, namely 0, 1 or 2. However, the winner can produce at least one offspring, and at least a winner can survive in reproduction. The offsprings of a winner and a loser are put at the tail of "winner-queue (WQ)" and "loser-queue (LQ)" respectively.

Here, two parameters p_w and p_l in Equation (2) have strong relationship with population size. If two parameters are set to $p_w = p_l$, expected population size (the length of queues) is constant; however, due to the effect of stochastic processes, actual population size for a specific GA run fluctuates (random walks) along time, even if parameters are set to $p_w = p_l$. Moreover, expected population size grows monotonically over time when $p_w > p_l$. Population size in QGA is therefore *not constant*. This nature characterizes QGA from many other traditional GAs, and a queue is a very effective data structure to manage variable population size. In consequence, population size in QGA is implicitly expressed as the length of a queue.

Moreover, two parameters are also related to selective pressure. High p_w and p_l values quickly delete out losers and increase winners in the population; which might result in premature convergence. Thus the appropriate settings of two parameters are important issues in this scheme to keep optimal selective pressure, or optimal balance between exploration and exploitation.

Now, it is time to have a brief quantitative analysis on two parameters above. Combining Equation (1) and Equation (2) yields the expected number of offsprings O of an individual with the rank S as follows:

$$\begin{aligned} O &= s(1+p_w) + t(1-p_l) \\ &= 1 + sp_w - tp_l \qquad (3) \\ &= (p_w + p_l)s + 1 - p_l \qquad (4) \end{aligned}$$

From Equation (3), it is clear that if $sp_w > tp_l$, the expected number of offsprings is more than one. In other words, the number of offspring is exactly equal to one if

$$s_{\text{th}} = \frac{p_l}{p_w + p_l} \tag{5}$$

unless $p_w = p_l = 0$. This shows that if the rank of an individual is above this threshold s_{th}, then this individual is expected to increase in the next reproduction. Another important point Equation (4) shows is that O increases monotonically with s; hence high performance individuals are expected to have more offsprings.

Further development of Equation (4) in the framework of schema may lead to a variant of well-known "Schema Theorem" [1], but this development is left for future works.

2.5 Crossover

Crossover is a bit-string recombination operator also in QGA. However crossover has another important role; that is, by assigning high priority to crossover that involves WQ, crossover also serves as another operator to to increase or decrease selective pressure. In this scheme, a winner not only increases the number of copies in the population through reproduction, but also enjoys a "short" lifetime in the sense that it is expected to have a high change of experiencing more genetic operations, and achieve faster feedback of their genetic material to the population.

To realize such selective pressure, three types of crossover is proposed with probability C_i:

Type 1 Crossover within WQ (prob. C_1)

Type 2 Crossover between WQ and LQ (prob. C_2)

Type 3 Crossover within LQ (prob. C_3)

where a set of probability satisfies the following equation:

$$\sum_{i=1}^{3} C_i = 1 \tag{6}$$

In crossover stage, each crossover operator is selected in accordance with probability assigned to each crossover type.

Here mating is also a FIFO-based mating[2]. Then in crossover of Type 1 and Type 3, an individual at the head of each queue is first selected, and another individual is searched along the queue to find the first individual that has different genotype from the head. On the other hand, in crossover of Type 2, an individual at the head of WQ is first selected, and another individual in LQ is searched in the same manner.

Now relationship between parameters appeared in reproduction and crossover becomes another interesting issue. For the optimal assignment of crossover probabilities for winners, those probabilities should be set so that the offsprings produced from a winner at the reproduction stage is all consumed in crossover stage. Then we can have the following equation:

$$N_w = 1 + p_w = 2C_1 + C_2 \tag{7}$$

where the coefficient "2" with C_1 is necessary since in Type 1 crossover two individuals are involved. Referring to Equation (6), we can determine parameter values if we set a value to one of three parameters. Perhaps the most interesting parameter is C_2, since it controls the degree of mixing between winners and losers. If we set C_2, other parameters can be obtained with the following equations.

$$C_1 = \frac{1 + p_w - C_2}{2} \tag{8}$$

$$C_3 = 1 - C_1 - C_2 \tag{9}$$

The recombination of bit-strings can be done in any ways, such as one-point crossover, two-point crossover, uniform crossover, and so on. The selection of appropriate recombination operator is independent of the overall structure of QGA.

2.6 Mutation

The same argument like above applies to the mutation operator. Any mutation schemes proposed so far can also be applied to QGA. However, the biggest difference of QGA is that you can set different mutation probabilities to different crossover types. For example, a crossover within WQ deserves low mutation probability because it is a recombination of promising genes. On the contrary a crossover within LQ deserves high mutation probability to explore search space. Hence in QGA, mutation probability M_i is associated to each crossover type.

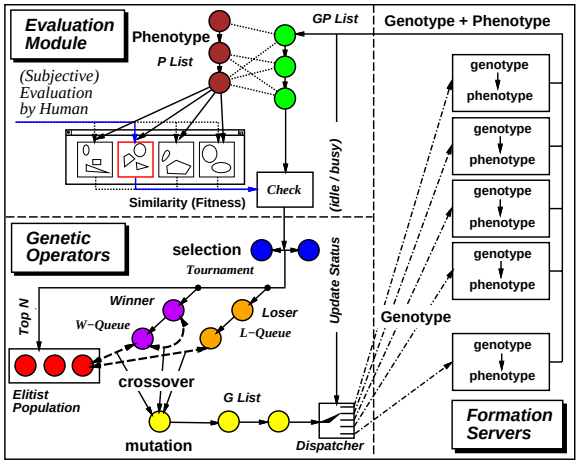


Figure 3: Extended model of QGA used for image browsing.

3 EXTENDED MODEL OF QGA

3.1 Elite Population

The basic structure of QGA is explained thus far. However, the author has several ideas on the extension of QGA. One extension is the introduction of another fixed-size population called "elite population." The role of this population is to maintain individuals with particularly high performance. An insertion scheme such as keeping the best individuals found so far is closely related to a population model used in GENITOR algorithm [11] or CHC algorithm [12]. In the framework of QGA, queue-based populations keep diversity, while elite population keep individuals that could guide the search to promising search areas.

3.1.1 Insertion into Elite Population

There should be many variants of insertion schemes other than keeping best individuals, and now an insertion scheme that focuses on niching is proposed as follows.

1. Define the size of the elite population.
2. Fill this array-based population by randomly generated initial individuals in queue-based populations.
3. A winner is compared to one of individuals in the elite population which is closest to the winner. Compare fitness between two individuals, and if newly appeared individual has high fitness, then replace an individual in the elite population with a winner.
4. If distance between arbitrary two individuals in the elite population is below threshold D_t, then leave the higher individual and remove the lower one to make make a room for a new comer. Goto 3.

This simple insertion scheme for niching addressed above is similar to the voronoi tessellation of the search space with the representation point being updated based on the algorithm above. Although this algorithm is relatively prone to be captured in local optimal, and one parameter D_t is critical in this algorithm in spite of lacking practical method to estimate an appropriate value.

3.1.2 New Types of Crossover

After the introduction of the elite population, it is natural to add two types of crossover in the list such as the following.

Type 4 Crossover between Elite and WQ (prob. C_4)

Type 5 Crossover between Elite and LQ (prob. C_5)

where the following equation is satisfied:

$$\sum_{i=1}^{5} C_i = 1 \tag{10}$$

[2]Other schemes such as incest prevention mating or niching can also be implemented.

In these crossovers, one of two offsprings produced from crossover is appended at the tail of an appropriate queue. The introduction of this crossover hopefully speeds up convergence guided by individuals stored in the elite population. In this case, Equation (7) can be rewrited as

$$N_w = 1 + p_w = 2C_1 + C_2 + C_4 \tag{11}$$

Since the degree of freedom is three, we have to determine three parameters among five parameters. Again C_2 controls the degree of mixing between winners and losers. Perhaps more interesting parameters are C_4 and C_5. The role of these crossovers are to inject promising genetic material into queue-based population. If C_4 and C_5 are set to a higher value, then these crossovers have a strong effect on directing search areas. That is because these crossovers have an effect on the whole population in the same way as having many copies in the population. In other words, we can think of a "virtual population" in which the percentage of C_4 is the copies of elite population. It can be achieved without actually having many copies in the population, instead having another population that preserves good individuals. In short, new types of crossover works like a "crossover-frequency-controlled selection mechanism," although this idea requires rigorous experimental validation.

Thus the setting of C_4 of C_5 is important to control the convergence and diversity of populations. In particular C_4 is the probability of combining an elite with a individual in WQ, expectedly an individual with high performance, so this combination may succeed in local search around optimum points, or may fail in premature convergence.

3.2 Parallelism

Parallel GA has been an active research area with the purpose of, for example, achieving fast computation time, evolving spatially structured population, simulating mutual reaction between individuals or populations. Asynchronous configuration of QGA is also an advantageous trait in parallel GA. QGA does not require global information nor global synchronization points, so the extension to parallel QGA is straightforward. Paralleized QGA will be useful in two ways.

Overlapped Modules The first parallel version is a solution to the problem addressed in Section 2.1.2. To reduce waiting time for both a human user and a computer, one solution is to overlap modules so that modules for a human user such as evaluation module, and modules for a computer such as genetic operations, can overlap and work simultaneously like a pipeline processing of CPU. This is not a parallel GA in a usual sense, but this is an important direction in IEC framework.

Multiple (Sub)Populations Another version is much more typical in GA research. Although the author still did not test this type of parallel GA, the extension of QGA to this type of parallel GA is natural and straightforward. By treating QGA as a module that serves as a (sub)population, and connect multiple QGA to form parallelized QGA. This is a promising area of research, but remains as a future work.

4 Experimental Results

4.1 Royal Road Experiments

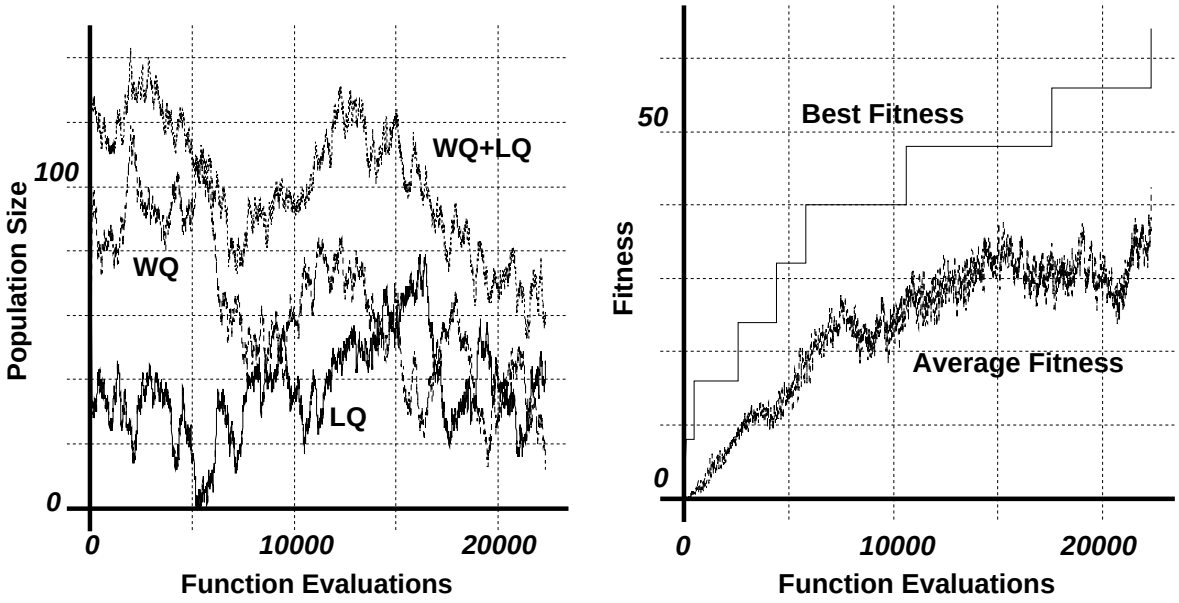
4.1.1 Parameter Setting

The first test functions used in this paper are royal road (RR) functions. For the detail of the function, readers should refer to [13, 14]. These test functions are advantageous because these functions are rather simple and well studied. Among RR functions, $R1$, $R2$, $R2_{flat}$ was used. The parameters used in QGA refers to the parameters used in [13], namely the length of bit-string = 64 ($K = 8$, $N = 8$), population size = 128, single-point crossover, sigma scaling with maximum fitness 1.5, crossover probability 0.7, mutation probability $m = 0.005$ per bit.

In QGA, initial population size was also 128, but after that population size fluctuated because of the effect of stochastic process. Another consideration is the counterpart of sigma scaling. As stated above, QGA does not use any scaling mechanism. However, Equation (4) shows that mostly best performing individual has the expected offsprings of $1 + p_w(s = 1)$, and mostly worst performing individual, $1 - p_l(s = 0)$. Then parameter values $p_w = p_l = 0.5$ is expected to have a similar effect on the convergence speed as sigma scaling used in [13]. Crossover probability for each crossover type is set to $C_2 = 0.4$. Mutation probability for each crossover type is set to $M_1 = 0.0$, $M_2 = 5m$, $M_3 = 10m$. M_2 and M_3 has relatively high mutation rate, but M_1 is zero probability of mutation; namely Type 1 crossover is not affected by mutation.

Table 1: The number of mean and median function evaluations taken to find the optimum over 200 runs on test functions.

	$R1$	$R2$	$R2_{flat}$
SGA Mean [13]	62099	73563	62692
SGA Median [13]	56576	66304	56448
QGA Mean	33796	79010	61427
QGA Median	27680	52484	43124



(a) Behavior of Population (b) Behavior of Fitness

Figure 4: Behavior of population and fitness of QGA.

4.1.2 Results

First, Table 1 gives the mean and median function evaluations taken to find the optimum over 200 runs on each function. Since I cannot assure that the experiments are carried out on the same ground, it is difficult to compare directly these results, but QGA seems to be an effective algorithm for $R1$. The author is thinking of more intensive comparison of two GAs.

Next a rough sketch on the behavior of QGA is presented in Figure 4. As stated before, the population changes over time, and fluctuates like a random walk. However, this fluctuation is handled by the queue-based population structure. The behavior of fitness is also shown in Figure 4. Average fitness is calculated as the moving average of 128 individuals. It shows steady improvement of average fitness, and at the final stage, best individual with fitness 64 was found, which terminated this GA run.

4.2 Computational Complexity

It requires more computation cost due to the management of data structures, especially queues. However, in real problems, usually computation cost required for function evaluation becomes the most important part of computational complexity [15, 14]. So I believe, although without quantitative evaluation, that the additional computation cost required by the management of these data structures can be well compensated if this algorithm can considerably reduce the number of function evaluation.

5 CONCLUSION

This paper proposed a novel GA called "a queue-based genetic algorithm." This algorithm is unique in that the data structure of the population is based on a first-in-first-out queue, and it is the key idea to realize asynchronous structure of the algorithm. This algorithm is in particular developed in relation to interactive evolutionary computation (IEC) framework, and the characteristics of QGA came from the solution to the problem of IEC framework. Various novel genetic operations have developed, and some of them were analyzed

in a quantitative manner. Finally QGA was compared with SGA in terms of royal road test function, and behavior of the algorithm was illustrated in terms of population and fitness.

There remains many topics for future works, including intensive comparison of QGA with other GA methods, adaptation of parameters during search, finding out better parameter settings, and extension of QGA into several direction as addressed in Section 3.

References

[1] Holland, J.H. Adaptation in Natural and Artificial Systems. MIT Press, 1992.

[2] Goldberg, D.E. Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, 1989.

[3] DeJong, K.A. and Sarma, J. Generation Gaps Revisited. In Whitley, L.D., editor, *Foundations of Genetic Algorithms 2*, pp. 19–28. Morgan Kaufmann Publishers, 1993.

[4] Kitamoto, A. and Takagi, M. Interactive Image Browsing Using Queue-Based Genetic Algorithm. *The Journal of the Japanese Society of Artificial Intelligence*, Vol. 13, No. 5, pp. 728–738, 1998. (in Japanese).

[5] Dawkins, R. The Blind Watchmaker. Longman, 1986.

[6] Sims, K. Artificial Evolution for Computer Graphics. *Computer Graphics*, Vol. 25, No. 4,, 1991.

[7] Caldwell, C. and Johnston, V.S. Tracking a Criminal Suspect through Face-Space with a Genetic Algorithm. In *Proc. of the Fourth International Conference on Genetic Algorithms*, pp. 416–421, 1991.

[8] Baker, E. Evolving Line Drawings. In *Graphics Interface*, 1994.

[9] Venturini, G., Slimane, M., Morin, F., and Beauville, J.P. Asselinde. On Using Interactive Genetic Algorithms for Knowledge Discovery in Databases. In Bäck, T., editor, *Proc. of the Seventh International Conference on Genetic Algorithms*, pp. 696–703. Morgan Kaufmann Publishers, 1997.

[10] Kitamoto, A. and Takagi, M. Learning Criteria for Similarity-Based Image Retrieval Using Simulated Breeding by Pipeline-Type Genetic Algorithms. *Technical Report of the Institute of Electronics, Information and Communication Engineers*, Vol. HIP96-4, pp. 17–22, 1996. (in Japanese).

[11] Whitley, D. The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In Shaffer, J.D., editor, *Proc. of the Third International Conference on Genetic Algorithms*, pp. 116–121. Morgan Kaufmann Publishers, 1989.

[12] Eshelman, L.J. The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In Rawlins, G.J.E., editor, *Foundations of Genetic Algorithms*, pp. 265–283. Morgan Kaufmann, 1991.

[13] Forrest, S. and Mitchell, M. Relative Building-Block Fitness and the Building-Block Hypothesis. In Whitley, L.D., editor, *Foundations of Genetic Algorithms 2*, pp. 109–126. Morgan Kaufmann Publishers, 1993.

[14] Mitchell, M. An Introduction to Genetic Algorithms. The MIT Press, 1996.

[15] Davis, L. Handbook of Genetic Algorithms. Van Nostrand Reinhold, 1990.