Parameterless Genetic Algorithms: Review and Innovation

by F. Daridi, N. Kharma and J. F.N. Salik

Concordia University, Montreal, QC

Abstract

We present a brief review of Genetic Algorithms (GAs) that do not require the manual tuning of their parameters, and are thus called Parameterless Genetic Algorithms (pGAs). There are three main categories of Parameterless GAs: Deterministic, Adaptive and Self-Adaptive pGAs. We also describe a new parameterless Genetic Algorithm (nGA), one that is easy to understand and implement, and which performs very well on a set of five standard test functions.

Sommaire

Nous présentons un bref examen des algorithmes génétiques (GA) qui n'exigent pas l'accord manuel de leurs paramètres, et nous appelons ainsi les algorithmes de "Parameterless Genetic Algorithms" (pGA). Il y a trois catégories principales des pGAs: PGAs déterministes, adaptatifs et individu-adaptatifs. Nous décrivons également un nouvel algorithme génétique parameterless (nGA), un il est facile comprendre qu'et instrument, et qui exécute très bien sur un ensemble de cinq fonctions standard d'essai.

1.0 Introduction and Review

Introduction. Holland [16] invented the Genetic Algorithm (GA) as an easy-to-use general method for a wide range of optimization problems. The performance of a GA is dependant on a number of factors, including candidate solution representation, and fitness evaluation and manipulation (via crossover and mutation). Both crossover and mutation have parameters (probability of crossover P_c and probability of mutation P_m) that require initialization and adjustment. For a given problem, these parameters, as well as the size of the population of candidate solutions (S), require careful manual optimization, often done through trial and error. Naturally, this diminishes the autonomy of GAs, and renders them much less attractive to potential users, such as engineers, that are not experts in GAs. Parameterless GAs (pGAs) represent an attempt (not yet complete or widely used) to eliminate the need for manual tuning of GA parameters.

1.1 Review. There are two main approaches to the elimination of parameters in GAs: a) Parameter Tuning, and b) Parameter Control.

Parameter tuning involves finding good values for the parameters before the GA is run and then using these values during the GA run. In an empirical study, De Jong [8] discovered a set of parameter values, which were good for the classes of test functions he used ($P_c = 0.6$, $P_m = 0.001$, $S = 60$). Using the same test functions as De Jong, Grefenstette [11] ran a (meta-) GA to find a set of parameter values for another GA ($P_c = 0.95$, $P_m = 0.001$, $S = 30$). As shown, both studies suggest the same low value for P_m (0.001), proposed double-digit values (< 100) for S, and used high (> 0.5) values for P_c. Although none of these two researchers were unable to prove that their sets were optimal for every optimizational task, their results were viewed by many GA users as sound empirically-founded guidelines.

In Parameter Control, one starts with certain initial parameter values; possibly the De Jong or Grefenstette's sets or some amalgamation thereof. These initial values are then adjusted, during run-time, in a number of ways. The manner in which the values of the parameters are adapted at run-time is the basis of Eiben's classification of Parameter Control into three different sub-categories (Eiben et al. [9]). These sub-categories are: (a) Deterministic, (b) Adaptive and finally (c) Self-adaptive. A brief review of published work in these three areas of Parameterless GAs follows.

A. Deterministic Parameterless GAs. In this type of Parameterless GAs, the values of the parameters are changed, during a run, according to a heuristic formula, which usually depends on time (i.e. number of generations or fitness evaluations).

Fogarty et al. [10] change the probability mutation in line with equation (1) - t is the generation number.

$$P_m = \frac{1}{240} + \frac{0.11375}{2^t} \tag{1}$$

Hesser et al. [14] derived a general formula for probability of mutation using the current generation number, in addition to a number of constants used to customize the formula for different optimization problems. Unfortunately, these constants are hard to compute for some optimization problems. In equation (2) n is the population size, l is the length of a chromosome (in bits), and t is the index of the current generation.

$$P_m = \sqrt{\frac{C_1}{C_2}} \frac{\exp((-C_3 t)/2)}{n\sqrt{l}} \tag{2}$$

Both Back [3] and Muhlenbein [17] discovered, experimentally, that $1/l$ is the best value for P_m for (1+l) GAs. A (1+l) GA is an algorithm that sees single parent chromosomes each producing a single child by means of mutation. Hence, the best of parent and child is passed to the next generation. In other studies [4], Back proposes a general formula for P_m, one that is a function of both generation number (t) and chromosome length (l). The formula is presented as equation (3); T is the maximum number of generations allowed in a GA run.

$$P_m = \left\{2 + \frac{l-2}{T-1}t\right\}^{-1} \tag{3}$$

All formulae presented above are variations on a single theme presented symbolically by $1/t$, where t is the generation number. In this theme the probability of mutation is initially very high, but is quickly reduced to a low and reasonably stable value. This agrees with common sense, as most GAs go through a short and frantic period of locating areas of interest on the fitness surface, followed by a lengthy and deliberate exploration of those locales (mainly via crossover). Naturally, random search (and hence mutation) are ineffective methods of exploration of large spaces. This simple fact leads to the incorporation of $1/l$ (and variants) into many formulae for P_m - l is the length of the chromosome which is linked to the dimensionality of the search space.

Not only is the need for manual tuning of P_m eliminated, but the performance of GAs is much improved by the use of time-dependant formulae for P_m. This conclusion is supported by a many studies, including [5].

B. Adaptive Parameterless GAs. In this mode of parameter control, information fed-back from the GA is used to adjust the values of the GA parameters, during runtime. However, (as opposed to self-adaptive control) these parameters have the same values for all individuals in the population.

Adaptive control was first used by Rechenberg [18]. He asserted that 1 every 5 mutations should lead to fitter individuals. As such, he enforced a variable mutation probability that was controlled by the rate of successful mutations in a population. If, at one point in time, the fraction of successful mutations was more than 1/5 then the probability of mutation is decreased and visa versa. Similarly, Bryant et al. [7] increased or decreased the probabilities of crossover and mutation (from initial values) as a function of how much or little those probabilities contributed to the generation of new fitter individuals in a given population: an elaborate credit allocation system was employed and is detailed in their paper.

Schlierkamp et al. [19] focused their efforts on adapting the size of the population. Indeed, they simultaneously evolved a number of populations with different sizes. After each generation, the population with the best maximum fitness is stored in a quality record. After a number of generations, the population with the highest record is increased; all other populations are decreased. In similar fashion, Hinterding et al. [15] ran 3 populations simultaneously. These populations had an initial size ratio of 1:2:4. After a certain pre-specified time interval the populations are halved doubled, or maintained as is, depending on the relative fitness values of their fittest individuals: the best population is doubled in size, while the worst one is halved; the last one is maintained as is. Along the same theme, Harik et al. [13] ran races between multiple populations of different sizes, allocating more time to those populations with higher maximum fitness, and firing new populations whenever older populations had drifted towards suboptimal (search) subspaces.

On a different note, Annunziato et al. [2] asserted that an individual's environment contains useful information that could be used as a basis for parameter tuning. They used a trip-partite scheme in which a new parameter (meeting probability) influences the likelihood of meeting between any two individuals, which (if they meet) can either mate or fight - see section 3.1.3 for details.

C. Self-adaptive Parameterless GAs. These GAs use parameter control methods that utilize information fed back from the GA, during its run, to adjust the values of parameters attached to each and every individual in the population. It was first used by Schwefel [20] in an Evolutionary Strategy (similar to a GA, but using real numbers and matching operators, instead of bit strings, for chromosomes), where he tried to control the mutation step size. Each chromosome in the population is combined with its own mutation variance, and this mutation variance is subjected to mutation and crossover (as is the rest of the chromosome). Back [6] extended Schwefel's [20] work to GAs. He added extra bits at the end of each chromosome to hold values for the mutation and crossover probabilities. At first, the mutation and crossover probability values were chosen at random. Then, these bits were subjected (again, with the rest of the chromosome) to the processes of evolution until, gradually, chromosomes with better probabilities (and better candidate solutions) appeared, and hence dominated the population.

Another way of self-adapting GA parameters, described by Srinivas et al. [21], involves assigning mutation and crossover probabilities to each chromosome, based on its own current fitness and the fitness of the population at large. On the other hand, Arabas et al. [1] defined a new quantity called remaining life time (or RLT). Every new individual is assigned a RLT variable. Each time a new generation is created, the RLT of every individual is updated using a bi-linear formula; how an individual's RLT is updated depends on whether its fitness is less than the average fitness of the current population (or not). Once the RLT of an individual reaches 0, it dies (i.e. is removed from the population).

2.0 Test Functions and Evaluation Measures

2.1 Test Functions. We use exactly the same test functions used in [6] - we restate them here for convenience. This set of test functions have:

- Problems resistant to hill-climbing,
- Nonlinear non-separable problems,
- Scalable functions,
- A canonical form,
- A few uni-modal functions,
- A few multi-modal functions of different complexity with many local optima,
- Multi-modal functions with irregularly arranged local optima, and
- High-dimensional functions.

All test functions have 10 dimensions and use 20 bits/variable except for *f5*, which uses 6 bits/variable.

$$f_1(\bar{x}) = \sum_{i=1}^{n} x_i^2 \tag{4}$$

$$f_2(\bar{x}) = \sum_{i=1}^{n-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2 \right) \tag{5}$$

$$f_3(\bar{x}) = -20\exp\left\{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right\} - \exp\left\{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right\} + 20 + e \tag{6}$$

$$f_4(\bar{x}) = 10n + \sum_{i=1}^{n}\left\{x_i^2 - 10\cos(2\pi x_i)\right\} \tag{7}$$

$$f_5(\bar{x}) = n - \sum_{i=1}^{n}\left\{\begin{array}{lll} \frac{0.92}{4}(4 - x_i) & if & x_i \le 4 \\ \frac{2.00}{4}(x_i - 4) & if & x_i > 4 \end{array}\right\} \tag{8}$$

where x_i is the number of 1 bits in gene i.

2.2 Evaluation Measures. In this section we explain a number of statistical measures that we use to evaluate the performance of genetic algorithms. These measures are listed under three headings: reliability, speed and memory load; the measures are defined in the order of their appearance in Table 1.

A. Reliability. Reliability of convergence is essentially the likelihood that the GA is going to converge to an optimal value, within a given number (say 500,000) of fitness evaluations. This following statistic measures reliability.

Percentage of Runs to Optimal Fitness: Each GA was run 30 times. This measure reflects the percentage of runs that were successful in converging to the optimal solution at or before 500 thousand (fitness function) evaluations.

B. Speed. Speed of convergence is essentially the (average) number of fitness evaluations required for a GA to optimally converge. This may be assessed using the following statistical measures.

Ave. No. of Evaluations to Best Fitness, and C.V.: This measure represents the average number of evaluations that are required for a GA to achieve its best fitness value in a run. In cases where the best fitness is 1, it serves as a measure of convergence velocity. Every run produces a different number of evaluations to best fitness. C.V. (Coefficient of Variation) is equal to the standard deviation of that set of evaluations, divided by the average. It is a measure of reliability. Ave. No. of Evaluations to Near-Optimal Fitness: Near-Optimal fitness is defined as a fitness of 0.95. In cases where optimal fitness is not obtained, near-optimal fitness is the next best measure of convergence velocity. This measure is defined in the same way as the preceding measure, except that we substitute near-optimal for optimal.

Average Best Fitness (and S.D.): This is the average of the set of best fitness values achieved in all 30 GA runs. S.D. is standard deviation of that set. Naturally, this is a crucial measure; GAs that are able to achieve a best fitness of 1 (and reliably) are taken seriously; those that return best fitnesses of less than 1 (or 1 but inconsistently) are not as good. Ave. Mean Fitness (and S.D.): This is the average of the set of average fitness values, achieved at the end of the 30 GA runs. S.D. is the standard deviation of that set.

C. Memory Load. This is the amount of memory required, on average, for a GA to achieve optimal convergence. Since the amount of memory correlates with the number of individuals in a given population, we can use population size as a measure of memory load. The following set of measures tackle that issue.

Table 1: Results of Applying nGA to Test Functions f_1 - f_5

nGA	Function 1	Function 2	Function 3	Function 4	Function 5
Percentage of Runs to Optimal Fitness	100%	100%	100%	100%	100%
Ave. No. of Evaluations to Best Fitness	14345	145980	28873	94640	10413
C.V.[1]	18.85	18.34	16.8	40.85	40.64
Ave. No. of Evaluations to Near-Optimal Fitness	4912	15512	9175	90465	6793
Average Best Fitness	1	1	1	1	1
S.D.	0	0	0	0	0
Ave. Mean Fitness	0.7784	0.51	0.7307	0.6	0.5241
S.D.	0.0423	0.0252	0.03	0.04	0.0339
Ave. Mean Population Size to Optimal Fitness	111.78	132.2	121.3	178	114.7
Ave. Max. Population Size to Optimal Fitness	111.78	132.2	121.3	178	114.7
Ave. Mean Population Size to Near-Optimal Fitness	77	117	78.4	178	98.2
Ave. Max. Population Size to Near-Optimal Fitness	77	117	78.4	178	98.2

Ave. Mean Population Size to Optimal Fitness (~ Memory Requirements): In a given run, the size of the population may differ from one generation to the next until (and after) the GA converges to the optimal value (if ever). In one run, the average size of all the populations preceding optimal convergence is called Average Population Size to Optimal Fitness (or APSOF). Every one of the 30 runs may return a value for APSOF. The average value for the set of APSOF values is the Ave. Mean Population Size to Optimal Fitness. Ave. Max. Population Size to Optimal Fitness: For each GA run, the largest population size prior to optimal convergence is stored in a set. The mean of that set is the average maximum population size to optimal fitness. Ave. Mean Population Size to Near-Optimal Fitness: In a given run, the size of the population may differ from one generation to the next until (and after) the GA converges to the near-optimal value of 0.95 (if ever). In one run, the average size of all the populations preceding near-optimal convergence is called Average Population Size to Near-Optimal Fitness (or APSNOF). Every one of the 30 runs may return a value for APSNOF. The average value for the set of APSNOF values is the Ave. Mean Population Size to Near-Optimal Fitness. Ave. Max. Population Size to Near-Optimal Fitness: For each GA run, the largest population size prior to near-optimal convergence is stored in a set. The mean of that set is the average maximum population size to near-optimal fitness. These measures allow GA users to assess the memory requirements for a given GA. The smaller the size of the population required for getting an optimally fit individual the better. This is because smaller populations require less memory. And, memory is a serious concern, still, if one is using large populations for real-world optimization and design problems.

3.0 A New Parameterless GA and Results

The simple Genetic Algorithm (SGA) has been applied successfully in many applications. However, it is not a parameterless GA. In this section, we describe a number of elaborations of the SGA that a) enhance the performance of the SGA, and b) make it into a pGA.

3.1 Stagnation-Triggered-Mutation (STM). The idea behind STM is simple: older individuals stuck at a sub-optimal point on the fitness surface for a long time need to be given some kind of "push" (e.g. mutation) to reach a new potentially more promising position on the surface. This feature helps GAs deal with fitness functions that are hard (and hence take long) to optimize, such as multi-modal functions (e.g. test functions *f3* and *f4* above).

Attached to each chromosome are two numbers; a mutation probability (P_m), and a new quantity, Life Time (or LT), which measures the number of generations passed since the chromosome was last modified (via crossover or mutation). Initially, P_m is equal to 1/l, where l is number of bits in the rest of the chromosome. In later generations, every chromosome that passes through (probabilistic) crossover and/or mutation is tested to see if it is identical to any of its parents. If it is, then its P_m is multiplied by its LT (and its LT is incremented by 1). If, on the other hand, this chromosome is altered (via crossover or/and mutation) then its P_m is reset to $1/l$ and its LT is reset to 0.

3.2 Reverse Traversal (RT), Phenotopic and Genotopic. Phenotopic Reverse Traversal deals with fitness surfaces that tend to drive the majority of the population towards local maxima and away from the global maximum (e.g. *f2*). RTP does this by getting a portion of the population to traverse the fitness surface against the gradient, i.e. towards minima rather than maxima. This also has the side effect of producing a more diverse population than simple fitness-proportional selection. In an RTP enhanced GA, 20% of the next generation is selected, via fitness proportional selection, but instead of selecting those individuals with the greatest fitness, RTP selects those with the lowest fitness.

Genotopic Reverse Traversal (RTG) deals with deceptive fitness surfaces (e.g. test function *f5* above). It does this by taking 20% of the individuals (after all selection and genetic operations are applied) and inverting their bits (turning 1's to 0's and 0's to 1's).This simple trick was the main factor behind the 100% reliability figure returned by the nGA for the fully deceptive function *f5*.

3.3 Non-Linear Fitness Amplification (NLA). This enhancement of the SGA is designed to deal with situations where the population converges to a rather flat neighborhood of a global optimum. In such cases, it is important that the selection mechanism becomes very sensitive to slight variations in the gradient of the fitness surface.

The way NLA works is straightforward: once the average fitness of the population exceeds 0.9, the fitness is scaled using equation (9); f' is the scaled fitness, f is the original un-scaled fitness and c is a constant (that we set to 100).

$$f' = 1 / (c\,(1 - f) + 1) \qquad (9)$$

The nGA introduces the three main new features explained above, but also uses a fixed probability of crossover equal = 0.7 (~ De Jong [8] empirically determined value), and implements elitism at 10%. To determine the minimum size of the population, a pre-run large population of 1000 individuals is created and the fitness of each individual is computed. Hence, the standard deviation of fitness of the population is computed (call that $SD_{fitness}$) and used in equation (10) (below). The size of the initial population is set to LowBound; but the population is allowed to grow to as much double that value (as a result of STM). Constant k is set to 3; the probability of failure (a) is set to 0.05; and sensitivity (d) to 0.005- see [13] for more detailed information about equation (10).

$$LowBound = -2^{k-1}\ .\ \ln(a)\ .\ SD_{fitness} / d \qquad (10)$$

In summary, the probability of mutation is variable and is determined by the mechanism outlined in STM. The probability of crossover is fixed at 0.7; and the size of the population is variable, but with lower and upper bounds.

As seen in Table 1, the new GA returned 100% reliability on all of the test functions. As to speed, reflected in the average number of evaluations to best fitness, the nGA is reasonably fast taking less than 146,000 fitness evaluations, on average, to achieve optimal convergence, which for an average population size of ~130 translates to 1023 generation. Finally, the amount of memory required to run the nGA is typical as the (average) maximum population size needed to reach optimal convergence never exceeded 178.

In figures 1a and 1b: blue stands for *f1*, red stands for *f2*, green stands for *f3*, black stands for *f4* and magenta for *f5*. The nGA was run 30 times per test function and the numbers used to plot the curves represent average values (over the 30 runs) of both fitness and diversity (entropy).

Figure 1a demonstrates the evolution of fitness. For four out of the five test functions, nGA's behavior is exemplary: it succeeds in converging by about 104 fitness evaluations; the only exception is function *f4*, which is the hardest multi-modal test function used. Indeed, the nGA performs better on the deceptive surface of function *f5* than on function *f4*, which is a testimony to the power of the anti-deceptive measures (Reverse Traversal of both colors) included in the nGA.

Figure 1b, on the other hand, demonstrates that the nGA maintains a high degree of diversity (entropy >= 10) throughout evolution - a positive feature of any GA.

4.0 Summary and Conclusions

In this paper, we present a brief (but thorough) review and classification of parameterless GAs. We define and use a number of statistical measures applicable to any parameterless GA; they are also platform-independent. Having them facilitates the process of comparing any number of GAs without having to repeat other people's work. They are also meaningful, in that they allow GA users to choose those GAs that are most reliable, fastest, or require the least amount of memory.

In addition, we propose a new parameterless GA (nGA), one that was born out of the problems encountered with existing pGAs. Our main goals in proposing the nGA is to a) build a more reliable pGA (which is proven by the results of Table 1), and to do so by b) adding a small number of easily realizable amendments to the simple GA.

It is our hope that given our success here, people would be more willing to adopt parameterless GAs as a common tool of optimization, rather than normal GAs, which require quite a bit of manual tuning by a domain expert, prior to application.

5.0 References

[1]. Arabas, J., Michalewicz, Z., & Mulawka, J., GAVaPS- A genetic algorithm with varying population size, Proceeding of the 1st IEEE Conference on Evolutionary Computation, IEEE Press, 1994.

[2]. Annunziato, M., & Pizzuti, S., Adaptive parameterization of evolutionary algorithms driven by reproduction and competition, Proceeding of ESIT2000, pp 246-256, Achen Germany.

[3]. Back, T., The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm, In R. Manner and B. Manderick, editors, Parallel Problem solving from Nature, pp 85-94, Elsevier Amsterdam, 1992.

[4]. Back, T., & Schutz M., Intelligent mutation rate control in canonical genetic algorithm, Proceeding of the International Symposium on Methodologies for Intelligent Systems, pp 158-167, 1996.

[5]. Back, T., Evolutionary Algorithms in theory and practice, Oxford University Press, 1996.

[6]. Back, T., Eiben, A.E., & Van der Vaart, N.A., An empirical study on GAs "without parameters", In Schenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E.,Merelo, J. J., and Schwefel, H-P. (Ed): Parallel Problem Solving from Nature PPSN V, Lecture Notes in Computer Science Vol. 1917, pp 315-324, 2000.

[7]. Bryant, A., & Julstrom, What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm, Proceeding of the Sixth International Conference on Genetic Algorithms, pp 81-7, Morgan Kufmann, 1995.

[8]. De Jong, K. A., An analysis of the behavior of a class of genetic adaptive systems, Doctoral dissertation, University of Michigan, Ann Arbor, University Microfilms No 76-9381, 1975.

[9]. Eiben, A.E., Hinterding, R., & Michalewicz, Z., Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation, Vol. 3, No. 2, pp 124-41, 1999.

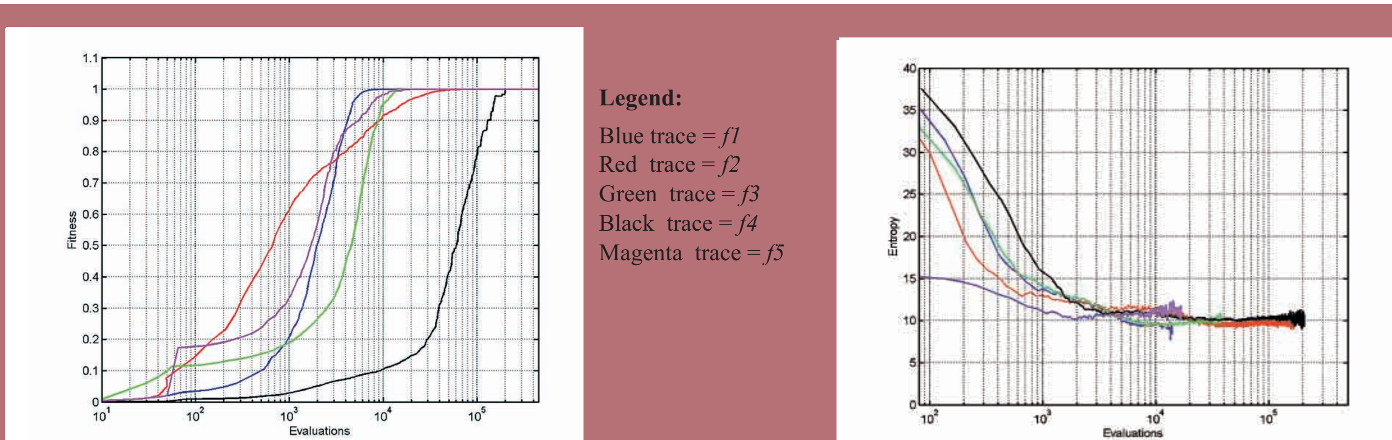

Figure 1: 1a) Fitness (left) and 1b) Entropy (right) for the New GA on Test Functions *f1* - *f5*

[10]. Fogarty, T., & Terence, C., Varying the probability of mutation in the genetic algorithm, Proceeding of the Third International Conference on Genetic algorithms, pp 104-109, Morgan Kufmann, 1989.

[11]. Grefenstette, J. J., Optimization of control parameters for genetic algorithms, In Sage, A. P. (Ed), IEEE Transactions on Systems, Man, and Cybernetics, Volume SMC-16-1, pp 122-128, New York: IEEE, 1986.

[12]. Harik, G., Cantu-Paz, E., Goldberg, D.E., Miller, B.L., The gambler's ruin problem, genetic algorithms, and the sizing of populations. Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 7 - 12, 1997.

[13]. Harik, G. R., & Lobo, F. G., A parameter-less genetic algorithm, Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.) GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, PP 258-267, San Francisco, CA: Morgan Kaufmann, 1999.

[14]. Hesser, J. & Manner, R., Towards an optimal mutation probability in genetic algorithms, Proceeding of the 1st Parallel Problem Solving from Nature, pp 23-32, Springer 1991.

[15]. Hinterding, R., Michalewicz, Z., & Peachy, T. C., Self-Adaptive genetic algorithm for numeric functions, Proceeding of the Fourth International Conference on Parallel Problem Solving from Nature, pp 420-429, in Lecture Notes from Computer Science, Springer Verlag, 1996.

[16]. Holland, J. H., Adaptation in natural and artificial systems, Ann Arbor, MI: University of Michigan Press, 1975.

[17]. Muhlenbein, H., How genetic algorithms really work: I. Mutation and Hill climbing, Parallel Problem Solving from Nature- pp SN II, 15-2, 1992.

[18]. Rechenberg, I., Evolutions strategie: Optimierung technischer systeme nach prinzipien der biologischen evolution, Frommann, 1973.

[19]. Schlierkamp-Voosen, D., & Muhlenbein, H. Adaptation of population sizes by competing subpopulations, Proceeding of International Conference on Evolutionary Computation (ICEC'96), Negoya, Japan, pp 330-335, 1996.

[20]. Schwefel, H-P., Numerische optimierung von computer-modellen mittels der evolutionsstrategie, Volume 26 of Interdisciplinary systems research. Birkhauser, Basel, 1997.

[21]. Srinivas, M., & Patniak, L. M., Adaptive Probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 4, pp 17-26,1994.

About the authors

Fady Draidi received the B.Sc.E.E. Degree from An-Najah National University, Nablus, Palestine, in 1999. Currently, he is completing an M.A.Sc. degree in Computer Engineering at Concordia University, Montreal, Canada. His research interests are Genetic Algorithms, image segmentation and enhancement, as well as shape detection.

Nawwaf Kharma is an Assistant Professor of Computer Engineering at Concordia University in Montreal. He had previously worked at the University of Paisley in Scotland and the University of British Columbia in Vancouver, B.C. He has research interests in Evolutionary Computation, Machine Learning and Pattern Recognition. He teaches both Genetic Algorithms and Machine Learning at the graduate level at Concordia. Dr. Kharma received his Ph.D. degree in Artificial Intelligence from Imperial College, University of London, England.

John F.N. Salik has professionally proven his aptitudes in various branches of advanced statistics, electronic hardware engineering, and software engineering. His special interest in statistics and related fields such as random processes, detection, estimation and information theory have strongly influenced his research activities over the years. John has received a technical degree from Vanier College with honours and has completed his bachelor's degree in Computer Engineering at Concordia University, where he is also completing his graduate research for his M.A.Sc. degree in Electrical Engineering.

DeCew Falls Hydroelectric Generating Station - Commemorative celebrations

IEEE recognizes the DeCew Falls Hydroelectric Generating Station as a Pioneering Project in distance transmission of electrical energy. The Power Generation Station, located in St. Catharines, Ontario, was the site of a Milestone Dedication Ceremony on May 2nd 2004. Members of the IEEE Hamilton Section, in co-operation with Ontario Power Generation, unveiled a commemorative plaque (see also page 13).

1. Ray Findlay of IEEE addresses the audience at the ceremonies.

3. Some of the dignatories present at the ceremonies. Seen are (left to right) Bill Kennedy, Janet Bradley, OPG Rep is Dave Heath (Plant Manager Niagara Systems - representative from Ontario Power Generation), Ray Findlay, Wally Read, Ron Potts and Honorable Jim Bradley (MPP Minister of Tourism and Recreation).

2. Members of the Hamilton Section with their banner and some of the pioneers who worked on the station commemorations. In the photo are (left to right) Ted Winch; Bob Barnett; Ray Findlay; Ed Shadeed; Janet Bradley; Scott Lowell; Blair MacCuish; Alan Jex and Ron Potts. Images by John Paytash (OPG).