openGA, a C++ Genetic Algorithm library

Conference Paper · October 2017

DOI: 10.1109/SMC.2017.8122921

5 authors, including:

Arash Mohammadi
Deakin University
11 PUBLICATIONS **30** CITATIONS
SEE PROFILE

Shady Mohamed
Deakin University
53 PUBLICATIONS **185** CITATIONS
SEE PROFILE

Kyle Nelson
Deakin University
15 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Saeid Nahavandi
Deakin University
725 PUBLICATIONS **5,380** CITATIONS
SEE PROFILE

2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

openGA, a C++ Genetic Algorithm library

Arash Mohammadi, Houshyar Asadi, Shady Mohamed, Kyle Nelson, Saeid Nahavandi

Deakin University, Geelong, Australia
Institute for Intelligent Systems Research and Innovation (IISRI)
Emails: arash.m@research.deakin.edu.au, houshyar.asadi@deakin.edu.au,
shady.mohamed@deakin.edu.au, kyle.nelson@deakin.edu.au, saeid.nahavandi@deakin.edu.au

***Abstract*—In this paper, an open source C++ Genetic Algorithm library is proposed called openGA. This library is capable of optimization in each of single objective, multi-objective and interactive modes. The main motivation for proposing this library is to provide freedom to users for designing their custom solution data model without limitations which many currently available software/libraries suffer from such as forcing a user to define the solutions as vectors or limiting the output of evaluation functions to a predefined format. In addition, the user has the entire control over genetic operations such as solution creation, mutation and crossover. The multi-object mode performs a Non-dominated Sorting Genetic Algorithm known as NSGA-III to obtain the pareto-optimal front while preserving the solution diversity. This library can handle multi-threading computations for single and multi-objective problems to increase the speed of the calculations significantly. The interactive mode is suitable for applications where human subjectivity is involved for evaluation of the cost function. Several simulation and tests are performed to verify the effectiveness of this library for calculations of optimization problems.**

I. Introduction

Genetic Algorithm (GA) is an adaptive heuristic search algorithm for optimization inspired by evolutionary natural selection within a population [1], [2], [3], [4]. GA is started with a set of solutions known as an initial population. Each solution is called chromosome or individual and it is made of different parameters known as genes. At each generation, the fitness of each solution is evaluated and the solution with a better fitness have a higher chance of survival and being transferred to the next generation. Offspring are produced from selected solutions via crossover and mutation operations. As solutions with better fitness values are more likely to be selected for producing offspring, the next generation is expected to have a better average fitness. The concepts such as mutation, crossover, inheritance and selection are inspired from evolutionary biology. The mentioned processes are repeated to produce new generations until one of termination conditions is met which typically include reaching a predefined maximum generation number, finding a satisfactory solution, user requests for a stop, average stall and best solution stall [3], [5]. The selection process determines the best solutions for the next generation. This process is critical because if its pressure is low, the convergence slows and if its pressure is high there is a chance of being stuck into local minima as a result of premature convergence [6], [1].

GA is a robust method for optimizations and it can tolerate reasonable noises. This method offers significantly better performance in optimization search through high dimensional space compared with conventional method [1]. This optimization method has a wide application in path planning and robotics [7], image processing [8], surface finishing and metalworking machinery [2], Motion Cueing Algorithm (MCA) [9], [10], [11], [12], Magnetic Resonance Imaging (MRI) [13], [14], [15], [16], [17], Electroencephalography (EEG) [18], [19], Inverse Kinematics problem [20], [21], [22], etc.

There are many problems with more than a single objective and often there is a trade-off between competitive objectives. In such problems, sorting the solutions will be difficult and there is no absolutely best solution, but the set of nondominated solutions called pareto-optimal front is preferred over the other solutions. Although scalarization through objective weighting converts a multi-objective problem into a single objective, the final solution depends on the selected weights hence such scalarization is avoided in many multi-objective applications [23]. There are several proposed multi-objective GA methods such as NSGA [24], SPEA2 [25] and PESA [26], to name but a few.

The optimization problems with four or more objectives are typically categorized as many-objective problems. The demand for solving such problems is increasing. However, according to the high dimensionality of the objective space, many of multi-objective GA methods are inefficient for these problems. The main challenge in many-objective optimizations is a large fraction of nondominated solutions, expensive diversity evaluation and insufficient recombination [27]. A Nondominated Sorting GA known as NSGA-III [27], [28] is proposed to optimize many-objective problems while preserving the diversity of solutions.

Interactive Genetic Algorithm (IGA) is another attractive field of optimization. IGA is a GA method where solutions are subjectively evaluated by a human [29], [30]. In another word, in IGA the fitness function is replaced by a human user to map the computation results from feature parameters to the psychological space [31]. Therefore, the user

978-1-5386-1644-4/17/$31.00 ©2017 IEEE

preference guides the optimization. IGA is highly preferred in applications where the available mathematical methods are unable to evaluate solutions. This method is highly effective for users without prior knowledge of the target problem [29]. Fig. 1 shows the schematic of IGA method. The optimal solution of IGA is sufficient to be an area rather than a single optimal solution as a human being is incapable of recognizing solutions which are psychologically equivalent [30], [31].

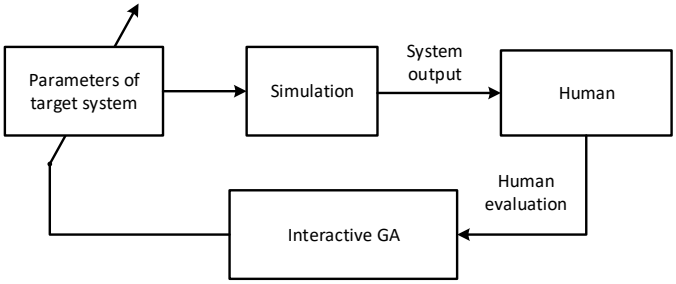

Fig. 1. Schematic of IGA

The number of generations in IGA is limited by human fatigue which is typically less than 20. Requesting a low resolution rating for cost value from a user rather than a rate between 0 to 100 is one of the effective ways to reduce psychological fatigue [31], [32]. The population of IGA is limited by restrictions on simultaneous visualization space and also potentials of a human being to remember presented solutions sequentially [31].

IGA is highly desired in applications involving human interests, aesthetic and creativity such as fashion design [33], dance-room spectroscopy [34], web design [35], graphic arts [36], computer graphic lighting [37], architecture [38], landscape design [39] and music [40]. IGA has also diverse engineering applications including robotics [41], [42], image processing [43], control [44] and particle filtering design [45].

In this paper, an open source GA library for C++ programs is introduced for working with three categories of problems: single objective, multi-objective and interactive GA optimization. This library provides a flexible framework for the programmer users to design their custom solution models. In addition, this library recognizes a middle computation stage where solutions are evaluated but the final costs are not finalized. Therefore, the middle costs are capable of carrying detailed information for future usage. The cost function is capable to reject a solution prior to, in the middle or at the end of the evaluation. This advantage is highly necessary as many constraints cannot be checked prior to the main evaluation. The structure and details of this library are further discussed in the next Section.

II. Proposed openGA library

A. Structure

In the openGA library, the model of genes for each solution is totally up to the designer and there is no restriction to mandate genes to be presented in form of a vector. Therefore, the programmer can define any structure or class prototype to represent a solution. Hence, the crossover and mutation operator functions must be defined by the programmer. The programmer also determines the method of result representation at each generation to the users such as console screen, file or graphic user interface. Fig. 3 depicts the flowchart for openGA for solving an optimization problem.

In the first step, the initial population is created. The gene mapping function is a function designed by the programmer. This function receives a random number generator function and produces a random structure or class from it. This function is responsible for filling all the genes with random numbers within the minimum and maximum of the allowed range.

For calculation of next generations, the crossover and mutation functions provided by the programmer are called. As solution models are highly customized, there is no possible way for the library to have access to the solutions directly. Therefore, the crossover and mutation must be defined by the programmer.

The evaluation function is separated from the fitness function. The heavy process of evaluation of solutions are assumed to be performed in the evaluation function and all the necessary information are extracted by this function. The result of this function is called middle cost as it is not yet finalized. The middle cost can contain more detailed information than what is necessary to the calculate the final cost. These data can include output signals or any information which is computationally expensive to be re-evaluated while it is already obtained as a by-product of the evaluation function and the user prefers to store them rather than calculate them again for further display. The final fitness function is responsible for obtaining the final cost from the middle cost. The flowchart of the consequences of how an individual is processed is illustrated in Fig. 2. This process is assumed to be computationally light. The evaluation function is also responsible for checking the boundaries. In a case of infringing any boundary, this function is able to reject the solution and it will be replaced by another individual candidate. The nonlinear conditions on the solutions are not always trivially clear and many of them are known only after the evaluation. This is the main motivation for combining the nonlinear conditions with the evaluation function. This library is highly convenient for optimization problems with a computationally expensive evaluation of solutions. Therefore, unlike many other GA libraries, openGA allows the highest customization for both solutions and evaluation results to tailor each data structure for their specific problems.

The following stopping criteria is considered for termination of GA optimization:

- In single objective problems, the fitness of the best solution is stalled for an adjusted number of generations.
- In single objective problems, the average fitness of the entire population is stalled for an adjusted number of generations.
- A maximum number of generations reached.
- The user has requested for a stop.

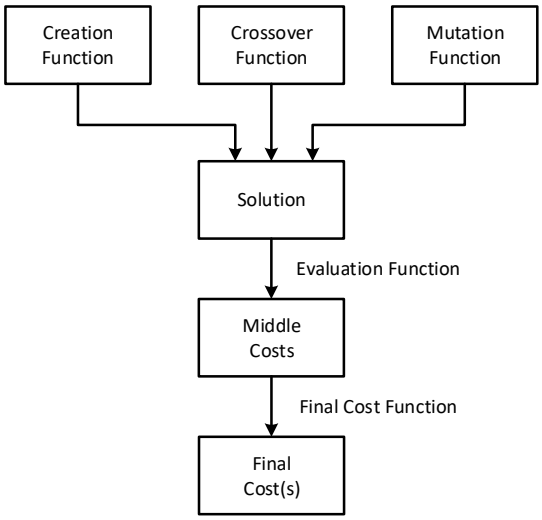

Fig. 2. The flowchart of cost process of solutions

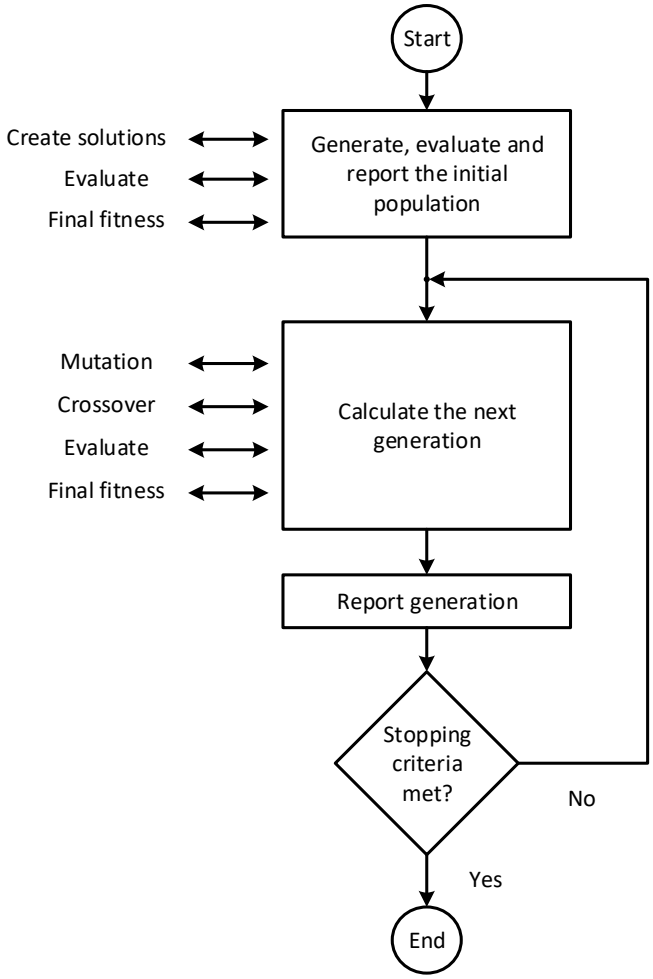

Fig. 3. The flowchart of openGA for solving an optimization problem

The modes of optimization are as follows

- Single objective mode
- Multi-objective mode (many-objective)
- Interactive mode

A single objective mode is used for where the user pursues only an individual objective. Each generation has a best and an average fitnesses. The population can be easily sorted and ranked for extracting elites and the selection operation.

The multi-objective mode is suitable for when more than a single objective is concerned and the importance of no objective is comparable with another via any weight. When solution A is worse than solution B in terms of one or some objectives without being better in terms of any objective then solution A is called dominated by solution B. In a multi-objective mode, the final output is the set of nondominated solutions. The final solution is determined via a decision-making process by the user which is separate from GA optimization. The multi-objective GA optimization method of the proposed library follows NSGA-III which is designed for many-objective problems.

The interactive mode is very similar to single objective (non-interactive) mode except for minor differences. For example, in single objective mode, the fitness functions evaluate each solution separately, while in IGA mode, the fitnesses of all solutions are calculated together hence they can influence each other as well.

After calculation of each generation, the generation is reported to a function provided by the programmer. A chronometer records the execution time for each generation separately. The execution time, best and average cost (single objective), the collection of solution data, pareto-optimal front (multi-objective) are reported to a user defined function after calculation of each generation.

B. Parallel Computation

OpenGA is designed for flexible modeling and optimization of problems with computationally expensive evaluation. As in single objective and multi-objective modes solution can be evaluated independently, evaluating the entire population is parallel is highly desired for us. Therefore, by default, the settings are adjusted to evaluate the population in parallel to increase the computational speed by engaging all CPU (Central Processing Unit) cores in computation. To implement this multi-threading design, a thread pool is applied with the adjustable number of workers which by default is equal to the number of CPU cores. Considering the fact that this library is written in C++ with low overhead in multi-threading computations, if not the fastest, this is one of the fastest open source libraries for solving single and multi- objective genetic algorithm problems.

C. License

OpenGA is open source and distributed under Mozilla Public License Version 2.0. The C++ codes are available at github website for public access.

D. Configuration

OpenGA uses C++ templates to benefit from compile time optimizations. Therefore, the data structure of chromosome

and middle cost are announced in the type definition of genetic class. The user can adjust problem mode, population number, crossover and mutation fraction, verbosity, elite count (single objective), average cost stall tolerance and its maximum count (single objective), the best cost stall tolerance and its maximum count (single objective), reference vector divisions and enabling them (multi-objective), enabling multi-threading and the number of workers in thread pool.

The user also must provide customized functions for random solution generation, evaluation, fitness calculation, mutation, crossover and report function.

III. Simulations and Discussion

In this Section, a single objective and a multi-objective problem are considered for the verification of the openGA library.

The single objective mode of openGA is verified via optimization of Rastrigin test function [46], [47] as follows

$$\begin{aligned} &\min\{f(\boldsymbol{x})\} \\ &f(\boldsymbol{x}) = 10n + \Sigma_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)) \\ &\forall i \in \{1 \dots n\}, -5.12 \leq x_i \leq +5.12 \end{aligned} \tag{1}$$

where n is the number of control variables. The analytical local minima are at $\boldsymbol{x} = \mathbf{0}$ where $f(\boldsymbol{x}) = 0$.

The test code is written in C++ and compiled by GNU Compiler Collection (GCC) [48], [49] in Linux environment. The computation is performed using multi-threading with an 8-cored CPU. This number of cores increases the speed of optimization multiple times.

The 5 variable Rastrigin problem is optimized within $0.208\ sec$ and in 90 generations according to Fig. 4. The population is set to 1000; the crossover rate is 0.7 and the mutation rate is 0.1. The final result is

$$\boldsymbol{x} = \begin{bmatrix} 1.63444841 \times 10^{-6} \\ 1.950667389 \times 10^{-6} \\ 4.602581993 \times 10^{-7} \\ 4.741552103 \times 10^{-6} \\ 6.771424889 \times 10^{-6} \end{bmatrix} \tag{2}$$

For testing openGA to solve a multi-objective GA, a DTLZ2 [50] problem is optimized as follows

$$\begin{aligned} &\min\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\} \\ &f_1(\boldsymbol{x}) = (1 + g(\boldsymbol{x}))\cos(x_1\frac{\pi}{2}) \\ &f_2(\boldsymbol{x}) = (1 + g(\boldsymbol{x}))\sin(x_1\frac{\pi}{2}) \\ &g(\boldsymbol{x}) = \Sigma_{x_i \in \boldsymbol{x}}(x_i - 0.5)^2 \\ &\forall i \in \{1 \dots n\}, 0 \leq x_i \leq 1 \end{aligned} \tag{3}$$

The population is set to 100 with 100 reference vector. The optimization test is performed within $0.27sec$ and the pareto-optimal front is depicted in Fig. 5. Reducing the number of reference vector down to 10 can produce gaps in the output results. Hence, the number of reference vector is set equal to the number of population. The objectives are competing with each other and there is no dominated objective on the pareto-front. Therefore, each two arbitrary selected solutions have advantages and disadvantages when compared with each other.

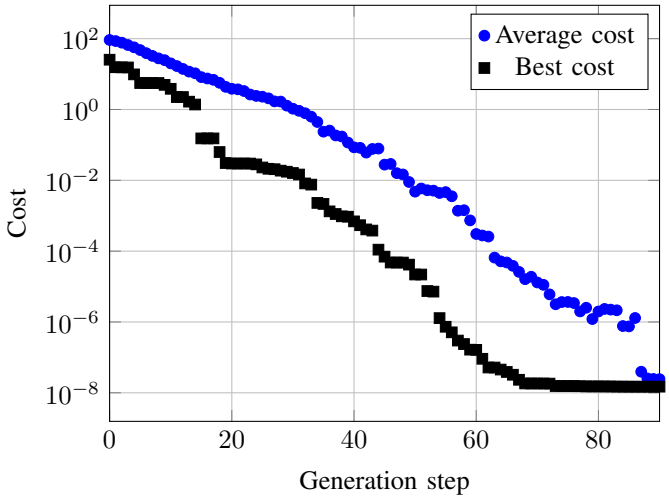

Fig. 4. Single objective GA convergence for Rastrigin problem

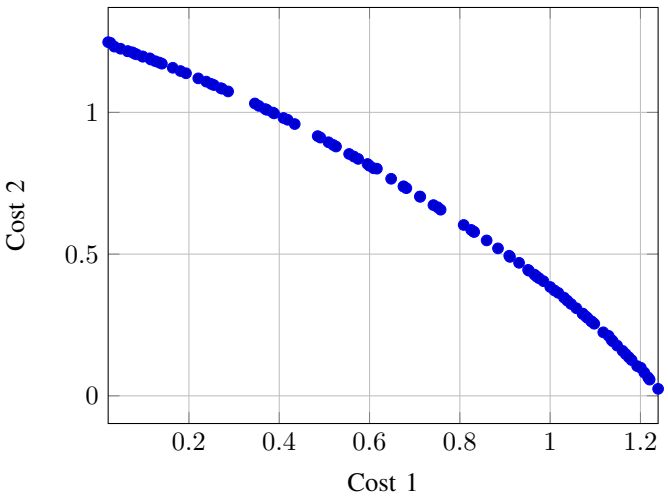

Fig. 5. The optimization pareto-front of DTLZ2 problem

Despite the unique goals of this GA library are totally different from the other available GA implementations, a performance benchmarking has been conducted. To compare the performance of openGA with the other algorithms, a 5-variable Rastrigin optimization scenario under similar conditions and settings have been applied and executed via MATLAB GA, GAlib and the proposed openGA library. The average result of multiple running has been provided in Table. I.

TABLE I
BENCHMARK COMPARISON BETWEEN MATLAB GA, GALIB AND THE PROPOSED OPENGA

	MATLAB GA	**GAlib**	**openGA**
Implementation Language	MATLAB	C++	C++
Norm of best result	0.0138	0.0420	0.0106
Execution time	1.251 sec	0.247 sec	0.207 sec
Supporting custom genes	✗	✗	✓
Chromosomes with additional information	✗	✗	✓

IV. CONCLUSION

In this paper, an open source C++ library for is introduced for the fast calculation of heavy optimization problems. This library is very efficient and employs a multi-threading evaluation approach to solve optimization problems even faster. The single, multi- and many- objective as well as interactive optimization problems can be handled by this library. The model of solutions is highly flexible and there is no necessity for the programmer to provide them in form of a vector. This level of generalization is handled by user defined creation, crossover and mutation functions for solutions. The architecture of this library allows the user to shift the constraint checking to after or in the middle of evaluation. This library separates the evaluation process from final fitness function and allows the result of the evaluation function to carry additional information such as fine details about components which constitute the final costs. Such additional details are common by-products of an evaluation process while in conventional GA solver designs, they are simply removed thus an additional effort is required to obtain the same data. Therefore, even in problems with computationally expensive evaluations, the proposed software architecture provides a significantly convenient environment to have an immediate access to the comprehensive results of evaluations after a GA optimization or during each generation report. Simulation results have shown the effectiveness of this library through single objective and multi-objective validation tests.

REFERENCES

[1] R. Sivaraj and T. Ravichandran, "A review of selection methods in genetic algorithm," *International journal of engineering science and technology*, vol. 1, no. 3, pp. 3792–3797, 2011.

[2] R. Saravanan, P. Asokan, and M. Sachidanandam, "A multi-objective genetic algorithm (ga) approach for optimization of surface grinding operations," *International journal of machine tools and manufacture*, vol. 42, no. 12, pp. 1327–1334, 2002.

[3] M. Kumar, M. Husian, N. Upreti, and D. Gupta, "Genetic algorithm: Review and application," *International Journal of Information Technology and Knowledge Management*, vol. 2, no. 2, pp. 451–454, 2010.

[4] M. Gorji Sefidmazgi, M. Moradi Kordmahalleh, and A. Homaifar, "Identification of switched models in non-stationary time series based on coordinate-descent and genetic algorithm," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 1399–1400.

[5] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A field guide to genetic programming*. Lulu. com, 2008.

[6] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, vol. 1, pp. 69–93, 1991.

[7] G. Yasuda and H. Takai, "Sensor-based path planning and intelligent steering control of nonholonomic mobile robots," in *Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE*, vol. 1. IEEE, 2001, pp. 317–322.

[8] M. Gong and Y.-H. Yang, "Multi-resolution stereo matching using genetic algorithm," in *Stereo and Multi-Baseline Vision, 2001.(SMBV 2001). Proceedings. IEEE Workshop on*. IEEE, 2001, pp. 21–29.

[9] A. Mohammadi, H. Asadi, S. Mohamed, K. Nelson, and S. Nahavandi, "MPC-based motion cueing algorithm with short prediction horizon using exponential weighting," in *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 000 521–000 526.

[10] A. Mohammadi, S. M. Houshyar Asadi, K. Nelson, and S. Nahavandi, "Future reference prediction in model predictive control based driving simulators," in *Australasian conference on robotics and automation (ACRA2016)*, 2016.

[11] H. Asadi, A. Mohammadi, S. Mohamed, D. R. Zadeh, and S. Nahavandi, "Adaptive washout algorithm based fuzzy tuning for improving human perception," in *International Conference on Neural Information Processing*. Springer, 2014, pp. 483–492.

[12] H. Asadi, A. Mohammadi, S. Mohamed, and S. Nahavandi, "Adaptive translational cueing motion algorithm using fuzzy based tilt coordination," in *International Conference on Neural Information Processing*. Springer, 2014, pp. 474–482.

[13] F. A. Razzaq, S. Mohamed, A. Bhatti, and S. Nahavandi, "Non-uniform sparsity in rapid compressive sensing mri," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2253–2258.

[14] ——, "Defining sub-regions in locally sparsified compressive sensing mri," *signal*, vol. 22, p. 361, 2013.

[15] S. Mohamed, S. Haggag, S. Nahavandi, and O. Haggag, "Towards automated quality assessment measure for eeg signals," *Neurocomputing*, 2017.

[16] S. Nahavandi, F. A. Razzaq, S. Mohamed, A. Bhatti, and P. Brotchie, "Locally sparsified compressive sensing in magnetic resonance imaging," in *Integrated Systems: Innovations and Applications*. Springer, 2015, pp. 195–209.

[17] M.-H. Kao, M. Temkit, and W. K. Wong, "Recent developments in optimal experimental designs for functional magnetic resonance imaging," *World journal of radiology*, vol. 6, no. 7, p. 437, 2014.

[18] I. Hettiarachchi, S. Mohamed, and S. Nahavandi, "A marginalised markov chain monte carlo approach for model based analysis of eeg data," in *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on*. IEEE, 2012, pp. 1539–1542.

[19] R. Roy, M. Mahadevappa, and C. Kumar, "Trajectory path planning of eeg controlled robotic arm using ga," *Procedia Computer Science*, vol. 84, pp. 147–151, 2016.

[20] P. Costa, J. Lima, A. I. Pereira, P. Costa, and A. Pinto, "An optimization approach for the inverse kinematics of a highly redundant robot," in *Proceedings of the Second International Afro-European Conference for Industrial Advancement AECIA 2015*. Springer, 2016, pp. 433–442.

[21] S. Momani, Z. S. Abo-Hammour, and O. M. Alsmadi, "Solution of inverse kinematics problem using genetic algorithms," *Applied Mathematics & Information Sciences*, vol. 10, no. 1, p. 225, 2016.

[22] M. Ayyıldız and K. Çetinkaya, "Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-dof serial robot manipulator," *Neural Computing and Applications*, vol. 27, no. 4, pp. 825–836, 2016.

[23] A. H. Dias and J. A. De Vasconcelos, "Multiobjective genetic algorithms applied to solve optimization problems," *IEEE Transactions on Magnetics*, vol. 38, no. 2, pp. 1133–1136, 2002.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[25] E. Zitzler, M. Laumanns, L. Thiele *et al.*, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[26] D. Corne, J. Knowles, and M. Oates, "The pareto envelope-based selection algorithm for multiobjective optimization," in *Parallel problem solving from nature PPSN VI*. Springer, 2000, pp. 839–848.

[27] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints." *IEEE Trans. Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[28] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach." *IEEE Trans. Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, 2014.
[29] F. Ito, T. Hiroyasu, M. Miki, and H. Yokouchi, "Discussion of offspring generation method for interactive genetic algorithms with consideration of multimodal preference," in *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2008, pp. 349–359.
[30] S.-B. Cho, "Emotional image and musical information retrieval with interactive genetic algorithm," *Proceedings of the IEEE*, vol. 92, no. 4, pp. 702–711, 2004.
[31] H. Takagi, "Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001.
[32] B. G. Woolley and K. O. Stanley, "A novel human-computer collaboration: combining novelty search with interactive evolution," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 233–240.
[33] M. Khajeh, P. Payvandy, and S. J. Derakhshan, "Fashion set design with an emphasis on fabric composition using the interactive genetic algorithm," *Fashion and Textiles*, vol. 3, no. 1, pp. 1–16, 2016.
[34] E. Davies, P. Tew, D. Glowacki, J. Smith, and T. Mitchell, "Evolving atomic aesthetics and dynamics," in *International Conference on Evolutionary and Biologically Inspired Music and Art*. Springer, 2016, pp. 17–30.
[35] A. Asllani and A. Lari, "Using genetic algorithm for dynamic and multiple criteria web-site optimizations," *European journal of operational research*, vol. 176, no. 3, pp. 1767–1777, 2007.
[36] M. Lewis, "Evolutionary visual art and design," in *The art of artificial evolution*. Springer, 2008, pp. 3–37.
[37] K. Aoki, H. Takagi, and N. Fujimura, "Interactive ga-based design support system for lighting design in computer graphics," in *Proceedings of the 4th International Conference on Soft Computing*, vol. 2, 1996, pp. 533–536.
[38] A. Serag, S. Ono, and S. Nakayama, "Using interactive evolutionary computation to generate creative building designs," *Artificial life and Robotics*, vol. 13, no. 1, pp. 246–250, 2008.
[39] S. Koma, Y. Yamabe, and A. Tani, "Research on urban landscape design using the interactive genetic algorithm and 3d images," *Visualization in Engineering*, vol. 5, no. 1, p. 1, 2017.
[40] M. Fukumoto and T. Hatanaka, "Parallel distributed interactive genetic algorithm for composing music melody suited to multiple users' feelings," in *Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference on*. IEEE, 2016, pp. 1–6.
[41] M. Virčíková and P. Sinčák, "Discovering art in robotic motion: From imitation to innovation via interactive evolution," in *International Conference on Ubiquitous Computing and Multimedia Applications*. Springer, 2011, pp. 183–190.
[42] H. Peng, C. Zhou, H. Hu, F. Chao, and J. Li, "Robotic dance in social roboticsa taxonomy," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 3, pp. 281–293, 2015.
[43] A. Miyamoto, M.-A. Konno, and E. Bruhwiler, "Automatic crack recognition system for concrete structures using image processing approach," *Asian Journal of Information Technology*, vol. 6, no. 5, pp. 553–561, 2007.
[44] N. Kubota, Y. Nojima, F. Kojima, and T. Fukuda, "Multiple fuzzy state-value functions for human evaluation through interactive trajectory planning of a partner robot," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 10, no. 10, pp. 891–901, 2006.
[45] Y. Zhang, S. Wang, and J. Li, "Improved particle filtering techniques based on generalized interactive genetic algorithm," *Journal of Systems Engineering and Electronics*, vol. 27, no. 1, pp. 242–250, 2016.
[46] N. Hansen and S. Kern, "Evaluating the cma evolution strategy on multimodal test functions," in *Parallel problem solving from nature-PPSN VIII*. Springer, 2004, pp. 282–291.
[47] M. Molga and C. Smutnicki, "Test functions for optimization needs," *Test functions for optimization needs*, 2005.
[48] A. Griffith, *GCC: the complete reference*. McGraw-Hill, Inc., 2002.
[49] GCC Team & others, "GCC, the gnu compiler collection," *https://gcc.gnu.org*, 2013.
[50] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Scalable test problems for evolutionary multiobjective optimization*. Springer, 2005.