A Generic Adaptive Multi-Gene-Set Genetic Algorithm (AMGA)

Adi A. Maaita, Jamal Zraqou, Fadi Hamad and Hamza A. Al-Sewadi
Faculty of Information Technology
Isra University
Amman, Jordan

***Abstract*—Genetic algorithms have been used extensively in solving complex solution-space search problems. However, certain problems can include multiple sub-problems in which multiple searches through distinct solution-spaces are required before the final solution combining all the sub-solutions is found. This paper presents a generic design of genetic algorithms which can be used for solving complex solution-space search problems that involve multiple sub-solutions. Such problems are very difficult to solve using basic genetic algorithm designs that utilize a single gene-set per chromosome. The suggested algorithm presents a generic solution which utilizes both multi-gene-set chromosomes, and an adaptive gene mutation rate scheme. The results presented from experiments done using an automatic graphical user interface generation case study, show that the suggested algorithm is capable of producing successful solutions where the common single-gene-set design fails.**

Keywords*—Genetic algorithm; Multi-gene-set; Single-gene-set; Artificial Intelligence; Generic algorithm; Generic architecture

I. INTRODUCTION

Nature inspired algorithms have been presenting astonishing results in solving problems that are not structured in nature. Genetic algorithms (GAs) are a category of nature inspired algorithms that have been used extensively in solving problems requiring an advanced form of heuristic search throughout a solution space, besides numerical and combinatorial optimization problems [1].

GAs are inspired from the natural processes of sexual reproduction and natural selection [1, 2]. The complete characteristics of a living being is miraculously encoded within its chromosomes, which have those life codes stored as Deoxyribonucleic Acid (DNA) molecules [3]. These huge molecules contain the life codes as characteristic encoding genes comprised of combinations of the primary nucleobases which are cytosine, guanine, adenine, and thymine [4].

During the process of sexual reproduction, each of the parents contributes genetic characteristics through providing half of their child's chromosomes. These chromosomes then undergo a process called crossover which causes parent chromosomes to break and then recombine into chromosomes with a gene set contributed by both parents. Hence, this child combines characteristics from both parents [3, 4]. The chromosomes and the genes they hold are referred to as the Genotype [5]. As for the characteristics that result from those genes, they are referred to as the Phenotype [6]. Moreover, sometimes genetic mutations occur. Such genetic mutations are changes in the original sequence of genes and can lead to evolution, health problems, or may have no effect [7].

After the introduction to GAs in section 1, a brief account of previous work is listed in section 2. The general structure of the GA is described in section 3. The proposed Aadaptive Multi-gene-set Genetic Algorithm (AMGA) is outlined in section 4 then section 5 details the architecture of this algorithm. Implementation and results are included in section 6 and finally section 7 concludes the paper.

II. PREVIOUS WORK

GAs found a wide area of applications for which to generate useful solutions. These applications did not spare any direction such as physics, mathematics, chemistry, medical, economics, computer, bioinformatics, pharmacology, etc. In the remainder of this section, a brief literature review of applications utilizing GAs for obtaining solutions is presented.

Shimamoto et al. [8] utilized a GA for flexible real-time dynamic routing control for traffic changes in broadband networks. It generates the exact solution for finding a routing arrangement that keeps the traffic loss-rate below a target value.

Lienig [9] proposed a novel parallel GA approach for performance-driven VLSI routing running on a distributed network and optimizes physical constraints such as nets size, crosstalk and delay.

Chun et al. [10], examined heuristic algorithms as the search tools for diverse optimization problems. The examined algorithms included the Immune algorithm (IA), the GA and the evolution strategy (ES).

Chang and Ramakrishna [11] examined GAs for the shortest path routing solution of the traveling salesman problem TSP, proving that the GA is one of the best heuristic algorithms to solve this problem. Variable-length chromosomes and their genes were utilized for encoding the problem, and partial chromosomes crossover with curing of all the infeasible chromosomes by a simple repair function were used for creating the diversity within the population.

Juang [12] proposed a recurrent fuzzy network for dynamic systems processing by using a neural network and a GA for optimizing the neural network. The fuzzy network was called TSK. It implements a series of recurrent fuzzy if-then rules with TSK type consequent parts with supervised learning. It proved superiority when applied to dynamic system.

Ozpineci et al. [13] proposed Harmonic optimization of multilevel converters using a GA. Optimum switching angles for cascaded multilevel inverters was achieved for eliminating some higher order harmonics while maintaining the required fundamental voltage.

Chowdhury et al. [14] designed an Encryption and Decryption algorithm for communication networks using a GA to robustly speed up and secure the total cryptography process. One point crossover and block cipher techniques were implemented for the simplification of the GA cryptosystem technique.

Mahdad et al. [15] presented a combined GA and fuzzy logic rules to enhance the optimal power flow with consideration of multi shunt flexible AC transmission systems. The presented method was effective in giving a near optimal solution and remarkably reduced the computation time.

Malhotra et al. [16] proposed a GA as an optimization tool for heuristic search applied to optimize process controllers for using natural operators. Their work explores the well-established methodologies of the literature to realize the workability and applicability of GAs for process control applications. GAs are applied to direct torque control of induction motor drive, speed control of gas turbines, and speed control of DC servo motors.

Kabeer et al. [17] used Boosted Feature Subset Selection (BFSS) as a preprocessing step to provide a gene subset that is fed to a GA, thus reducing the feature subset to smaller numbers and helping to generate a better optimal subset of genes. They claim that their hybrid approach shows better results compared to other well-known approaches when applied to leukemia, colon and lung cancer benchmarked datasets.

Ahmed [18] developed a simple GA using sequential constructive crossover to obtain heuristic solutions to the Bottleneck traveling salesman problem. He proposed a hybrid GA that incorporates 2-optimization search, another proposed local search and immigration to the simple GA for obtaining better solutions.

Umbarkar et al. [19] proposed Dual Population GA for solving Constrained Optimization Problems. It is based on maximum constraints satisfaction applied as a constraints handling technique and a Dual Population GA used as a meta-heuristic. It achieved close to optimum rather than exact optimum solution as compared with the Ant Colony algorithm, the Bee Colony algorithm, the Differential Evolution algorithm and the GA that have been used for solving the same problem set.

Moin et al. [20] proposed a hybrid GA with multi parents crossover for job shop scheduling problem (JSSP). The search space is reduced by generating a full-active schedule that satisfies precedence constraints, a neighborhood search is applied to exploit the search space for better solutions and to enhance the GA. Simulation suggests sustainability of this hybrid GA in solving JSSP.

Sankaran et al. [21] proposed a GA based parallel optimization technique aiming to improve the performance of batch schedule of two massively parallel application codes; a turbulent combustion flow solver (S3D) and a molecular dynamics code (LAMMPS). Experiments have shown a significant deviation from ideal weak scaling and variability in performance. This technique showed significant improvement in solving speed, besides improvement in variability and scalability.

The work in this paper suggests the use of a multi-gene-set chromosome GA with an adaptive gene mutation rate scheme, aiming to deal with multiple sub-problems in order to find a viable solution for complex solution-space problem. Such problems are usually very difficult to solve using the traditional single-gene-set chromosomes GA designs.

III. THE GENERAL STRUCTURE OF GENETIC ALGORITHMS

GAs were inspired from nature to solve problems to which we have no structured solutions that can be coded into algorithms. Examples of such problems are building and refining a set of production rules [22], then creating and adapting computer programs [23].

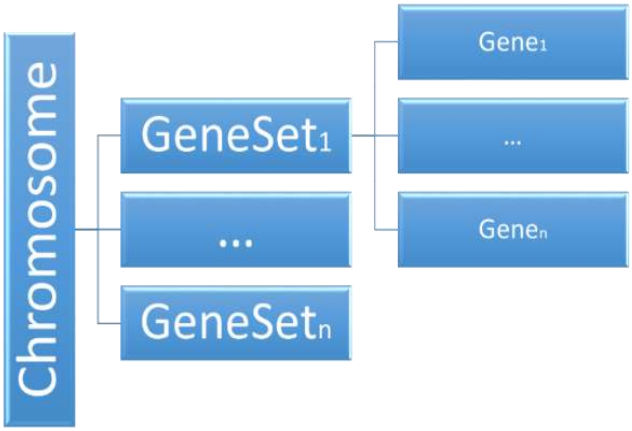

Fig. 1. A chromosome with multi-gene-sets

These kinds of problems require gradually evolved solutions rather than simply calculated ones. This is where GAs show their true power. They are evolutionary algorithms that use a number of initial solutions and attempt to evolve them in a way that eventually leads to the sought solution.

In order to perform an evolutionary search into the solution space, a GA requires a number of data structures, some specific rules and certain procedures. The main data structure of a GA is the cchromosome. The chromosome represents the genotype of a solution. This genotype is usually encoded as a string of 0's and 1's, however, it can be encoded using other types of data as well [1].

Another important part of a GA's structure is the algorithm for calculating a chromosome's fitness value. This algorithm is called the fitness function and it is responsible for providing a measure of a chromosome's quality as a solution to the problem at hand. It is important to note that a fitness function is problem specific. That means we need to create a new fitness calculating algorithm for each different problem. A fitness value is usually a measure of how close a certain solution is to the required solution [1, 2].

IV. THE PROPOSED ADAPTIVE MULTI-GENE-SET GENETIC ALGORITHM

The problem under consideration in this research considers a complex system with multiple objectives. Hence, an algorithm is proposed that is capable of looking into a complex solution space. It suggests an Adaptive Multi-gene-set Genetic Algorithm that has multiple heterogeneous solution fitness aspects. In other words, the problem requires searching for a complex solution that requires a genotype that contains multiple types of genes. The main challenge with such a genotype would lie in the complex crossover process. We simply cannot exchange gene data between genes containing different data semantics even if the data types match.

To visualize the multiple gene sets for a certain chromosome, a class diagram is shown in fig 4. Furthermore, neighboring multi-gene sets chromosomes would have some influence on each other through genetic operations such as crossover and mutations.

The crossover process is meant to create a new set of solutions that mix aspects of existing solutions. For that reason, we need a proper encoding of the genotype that preserves the purpose of crossover. The proposed algorithm suggests the use of multiple gene sets within a single chromosome. This would enable a complex problem to be divided into multiple smaller sub-problems which will be handled separately, and then those sub-solutions are gathered in order to create a complete final solution. Fig 2 illustrates a sketch for the expected crossover process amongst multi-gene set chromosomes.

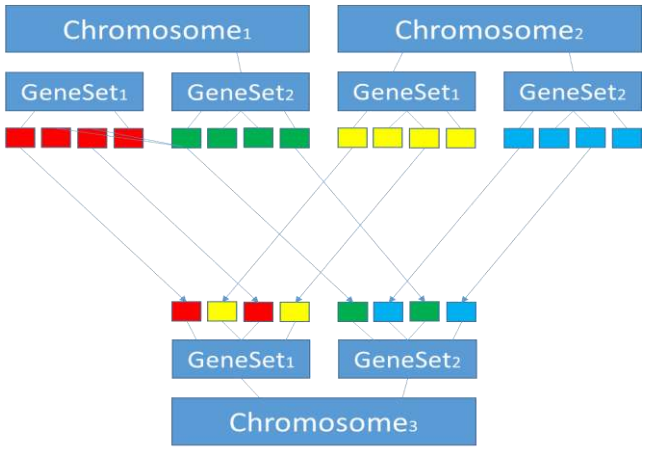

Fig. 2. Crossover amongst multi-gene-set chromosomes

Each sub-problem would require its own crossover process which is capable of generating offspring sub-solutions with data that are semantically correct. The fitness of each sub-solution needs to be calculated in order to determine whether that part of the complete solution needs further search for the current generation of the algorithm execution or not. This means that a sub-solution that may seem optimal for the sub-problem might not be suitable for the complete solution. Moreover, this would require searching for a new sub-solution during the coming generations until a suitable complete solution is found.

Such an architecture of the GA also allows for multi-threading to be efficiently used. This is made possible by separating the sub-solution search operations into distinct threads that run simultaneously and independently in parallel. This enables very efficient implementations of the algorithm to be created when compared to the traditional GA architecture.

The fitness of the complete solution needs to be calculated in each generation to determine how each of the sub-solutions search algorithms should operate .Fig 3 clarifies the contribution of Gene-set fitness to the calculation of Chromosomes fitness.

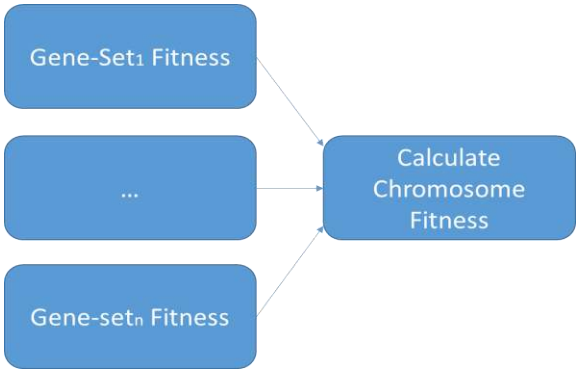

Fig. 3. Gene-set fitness values contribute towards the total chromosome fitness

A. *Adaptive mutation rates*

When the search results seem to converge towards unsuccessful solutions then it is time to add new genetic material to the mating pool. This is where mutation comes into play. Mutation is a process that changes the data of the genotype by creating new values that are usually randomly generated within predefined constraints. Mutation rate is usually predefined prior to the execution of the GA. This limits the behavior of the GA to a semi-static form when it comes to steering the search process towards a solution. Hence, the GA would not be able to increase the rate of introduction of new genetic material when the search converges towards unsuccessful solutions in order to escape that convergence. The GA will also be unable to reduce that rate of new genetic material introduction when the search seems to converge towards successful solutions. This reduction is very helpful when there is a need to concentrate the search within the existing genetic material that represents successfully evolving solutions.

In order to succeed, a GA needs to be fine-tuned by setting a proper mutation rate through a process of trial and error. If an unsuitable mutation rate is used, the search may never be able to converge successfully towards a solution. A higher than needed mutation rate would cause the algorithm to search the solution space blindly. A lower than needed mutation rate may lead the algorithm to converge towards unsuccessful solutions. This would be the result of existing genes representing the current mating pool of solutions being overly dominant and if those solutions have low fitness values then the new generated solutions are most likely to have a low fitness values as well.

On the other hand, an adaptive mutation rate would steer the search conducted by the GA towards successful solutions more efficiently than using a static rate. In this adaptive scheme, the mutation rate is adapted for each sub-solution according to statistics showing convergence towards successful solutions, or convergence towards unsuccessful solutions. Each mutation rate is re-evaluated for each generation. According to that evaluation, the mutation rate is increased or decreased according to equation (1), where mr_{new}, mr_{old}, and r_c are the new mutation rate, old mutation rate and convergence rate, respectively.

$$mr_{new} = mr_{old} + (mr_{old} \times r_c) \quad (1)$$

It is important to note that the rate of convergence is represented as negative values when the convergence is towards less successful solutions and it is positive when the convergence is towards more successful solutions.

V. THE ARCHITECTURE OF THE ADAPTIVE MULTI-GENE-SET GENETIC ALGORITHM

The class diagram representing the architecture of a traditional single-gene-set chromosome GA is shown in fig 4. The GA consists of a population of chromosomes that accommodate genes and data structures representing gene-data.

The proposed adaptive multi-gene-set chromosome GA is completely different from the traditional generic model as clarified in the class diagram shown in fig 5. The algorithm is designed to be generic in nature as it requires minimal modifications in order to be used for solving any evolutionary search problem. For example, an interested developer using this architecture would only need to create new classes implementing the interfaces IGene and IGeneData in order to solve any type of problem requiring chromosomes with multiple gene sets.

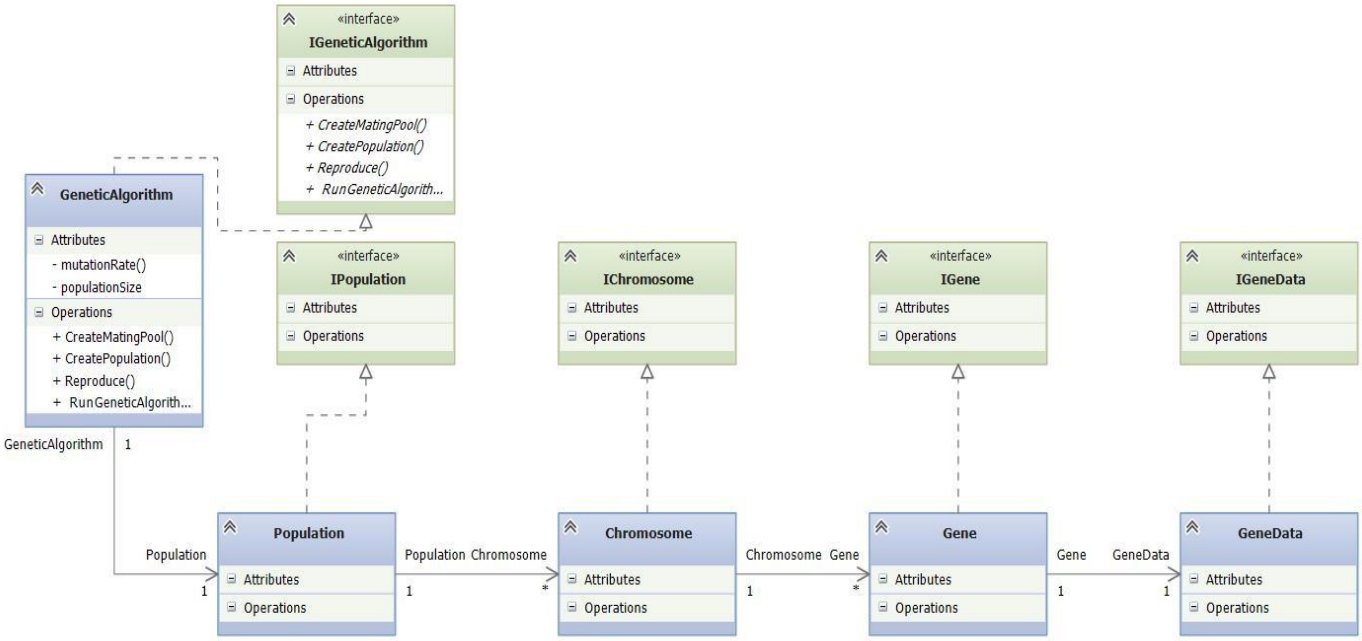

Fig. 4. Class diagram for the single-gene-set GA architecture

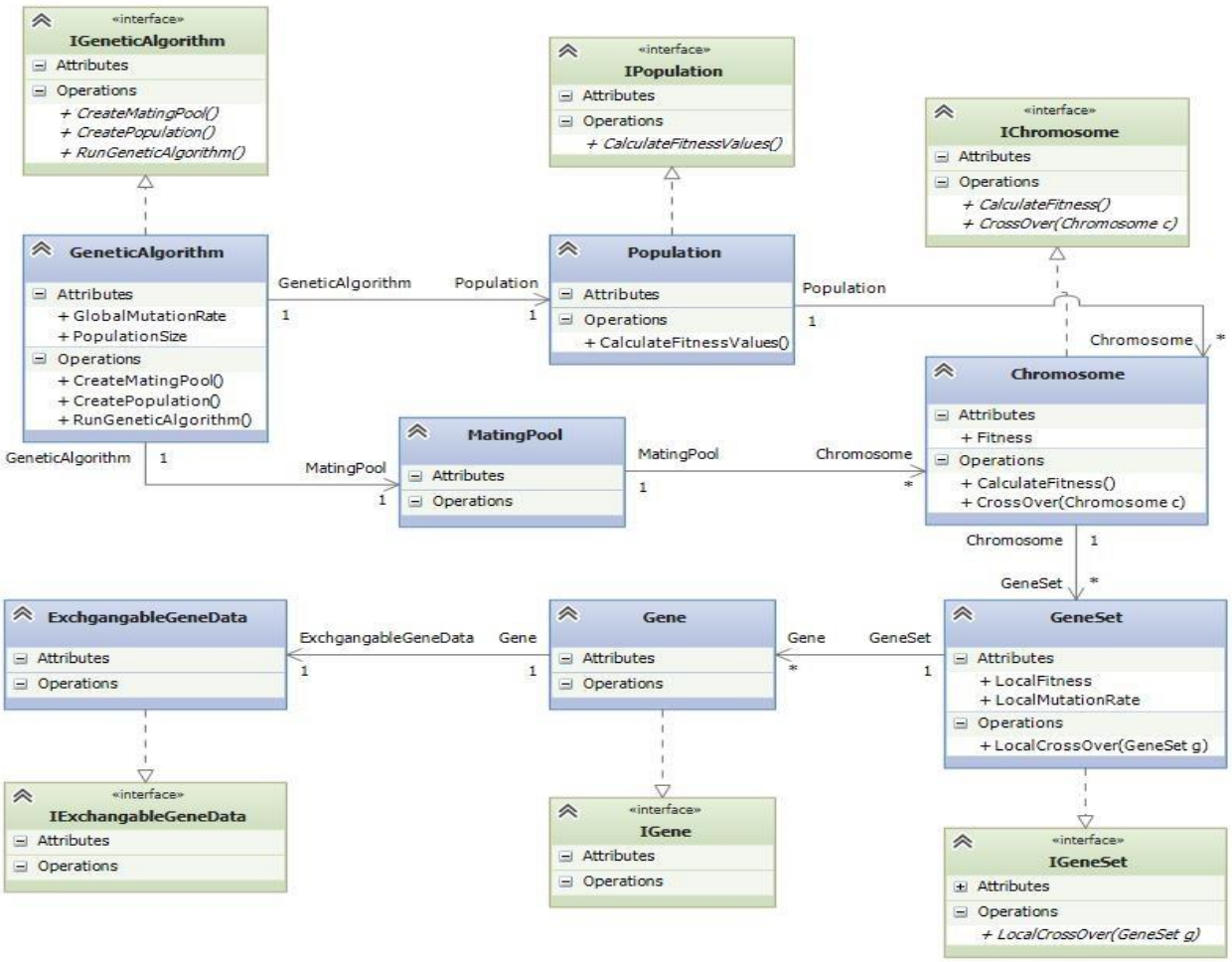

Fig. 5. Class diagram of the multi-gene-set GA architecture

VI. Implementation and Results

To demonstrate the effectiveness of the proposed AMGA, an experiment is conducted to implement it in the process of graphical user interface generation. The results are then compared to those obtained when conducting the same experiment using a static mutation rate single-gene-set GA.

The problem of automated graphical user interface generation using GAs was discussed in [24-26].

The genetic material of the graphical user interface generation problem consists of two types of gene-sets per chromosome. The first gene-set contains information on the containers used to host controls on a form. The second gene-set contains information on the controls that are used to create the graphical user interface.

The single-gene-set GA relies on a single gene-set to represent both the controls and the containers. In order to do that, the containers were treated as controls, and each regular control contained an attribute that stores a serial number representing its parent control. Each control was represented as a gene containing the following data:

1) *X-axis location in pixels.*
2) *Y-axis location in pixels.*
3) *Width in pixels*
4) *Height in pixels.*
5) *Margin in pixels.*
6) *Padding in pixels.*
7) *Parent container serial number.*
8) *Dock location on the form (top, bottom, left, right, full, none)*

Using that gene data, the algorithm needs to search for candidate formations of the supplied controls in order to accommodate the following criteria:

1) *Controls of the same type should be located within the same container.*
2) *Controls should be horizontally stacked if the parent container is vertically oriented and should be vertically stacked if the parent container is horizontally oriented.*
3) *Controls should be left aligned if they exist in a vertically oriented container and should be center aligned if they exist in a horizontally aligned container.*
4) *Controls of the same type within the same container should have the same height value in pixels.*

The multi-gene-set GA relies on two separate gene sets for representing the containers and the controls respectively. This requires creating two gene types. The first type represents the containers and it contains the following data:

1) *Container serial number.*
2) *Container type.*

While the exchangeable data include:

1) *Container dock location on the form (top, bottom, left, right, full, none).*
2) *Container width.*
3) *Container height.*

As for the second type, it represents the controls and it contains the following data:

1) *Control name.*
2) *Control type.*

While the exchangeable data include:

1) *X-axis location in pixels.*
2) *Y-axis location in pixels.*
3) *Width in pixels.*
4) *Height in pixels.*
5) *Margin in pixels.*
6) *Padding in pixels.*
7) *Parent container serial number.*

The obtained best fitness values for the case study when implementing the single-gene-set algorithm running for a maximum number of 1000 generations and having 5 containers with different numbers of controls (e.g. 10, 20 & 30), are summarized in Tables I. While the obtained best fitness values for implementing the AMGA running for the same parameters, are summarized in table II.

TABLE I. RUNNING THE SINGLE-GENE-SET GA

Single-gene-set Algorithm			
Max No. of generations = 1000			
No. of Containers	***No. of Controls***	***Generation of termination***	***Best fitness value***
5	10	1000	71.70%
5	20	1000	72.60%
5	30	1000	61.90%

TABLE II. RUNNING THE MULTI-GENE-SET GA

Multi-gene-set Algorithm			
Max No. of generations = 1000			
No. of Containers	***No. of Controls***	***Generation of termination***	***Best fitness value***
5	10	81	100%
5	20	556	100%
5	30	1000	93.70%

Comparison of the two tables indicates that the AMGA was able to achieve successful solutions where the single-gene-set GA failed for the same number of generations.

The fitness values were calculated for both, AMGA algorithm and single-gene-set GA as a function of the number of generations for different numbers of controls, (namely 10, 20 and 30). The results are plotted in fig 6 - 8.

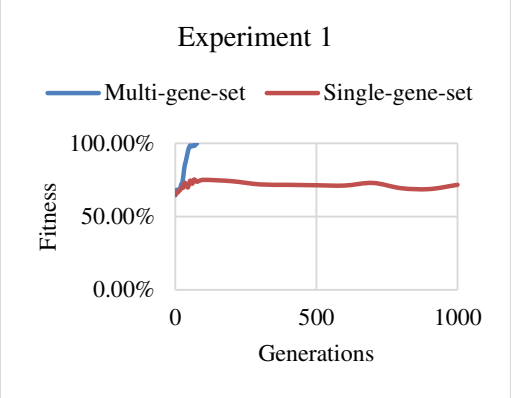

Fig. 6. Performance of the multi-gene-set algorithm when compared to its single-gene-set counterpart for the automated GUI generation problem with 5 containers and 10 controls

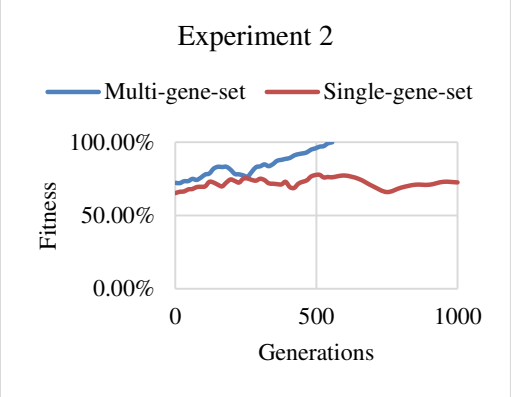

Fig. 7. Performance of the multi-gene-set algorithm when compared to its single-gene-set counterpart for the automated GUI generation problem with 5 containers and 20 controls

It is noticed that in the case of the multi-gene-set GA, convergence towards the best fitness (100% fitness) is achieved after 81 generations for 10 controls. But as the number of controls increases, the convergence gets slower as the available space on the canvas or form is too little to host all the required controls according to the required conditions, or it would even be impossible to place all the required controls on the canvas as the totality of the controls' areas

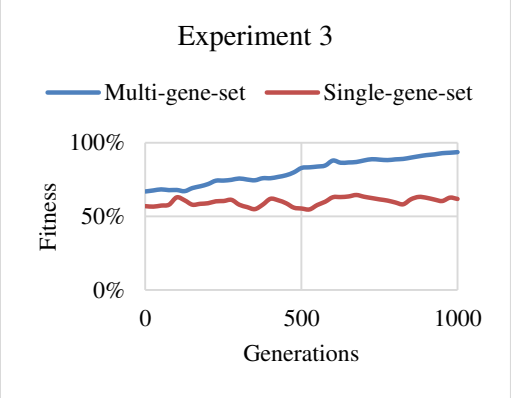

Fig. 8. Performance of the multi-gene-set algorithm when compared to its single-gene-set counterpart for the automated GUI generation problem with 5 containers and 30 controls

would be larger than the actual available space on the canvas. On the other hand, it can noticed that no convergence was achieved at all in the case of the single-gene-set GA.

VII. CONCLUSIONS

The results presented in this paper show that the AMGA is capable of converging towards solutions more efficiently than its single-gene-set counterpart. The results presented also show that the algorithm is also capable of recovering from divergence from solutions adaptively by increasing the rate of mutation when that is required. It is also capable of reducing the rate of mutation when the algorithm begins to converge towards successful solutions using the existing genetic material.

Future work relating to that presented in this paper would involve using meta-heuristic optimization techniques as part of the generic architecture of the AMGA. This would allow for more complex types of problems to be solved by avoiding the problem of the algorithm getting stuck on a local maxima. Such optimizations would also allow for better performance for the AMGA if implemented correctly.

REFERENCES

[1] Mitchell M., "An Introduction to Genetic Algorithms", 1998: MIT Press. 209.

[2] Luger G.F., "Artificial Intelligence: Structures and Strategies for Complex Problem Solving", 2008: Addison-Wesley Publishing Company. 784.

[3] Brown T., "Introduction to Genetics: A Molecular Approach", 1st Ed. 2011: Garland Science.

[4] Brooker, R.J., "Genetics: Analysis and Principles", 3rd Ed. 2008: McGraw-Hill Higher Education.

[5] Sahni N. , Song Yi, Q. Zhong, N. Jailkhani, B. Charloteaux, M. E. Cusick and M. Vidal, "Edgotype: a fundamental link between genotype and phenotype", Elsevier, Current Opinion in Genetics & Development 2013, Vol. 23, pp. 649–657

[6] Cooper D. N., Michael Krawczak, Constantin Polychronakos, Chris Tyler-Smith, and Hildegard Kehrer-Sawatzki, "Where genotype is not predictive of phenotype: towards an understanding of the molecular basis of reduced penetrance in human inherited disease", Hum Genet 2013, Vol. 132, PP1077–1130

[7] "The New Genetics, National Institute of General Medical Sciences National Institutes of Health U.S. Department of Health and Human Services", NIH Publication No.10¬662Revised April2010, http://www.nigms.nih.gov.

[8] Shimamoto, N., Hiramatsu, A. and Yamasaki, K. (2000). A dynamic routing control based on a genetic algorithm, IEEE International Conference on Neural Networks, 1993, Vol.2, pp. 1123-1128.

[9] Lienig, J. (1997). A parallel genetic algorithm for performance-driven VLSI routing, IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 29-39.

[10] Chun J. S., Hyun-Kyo Jung and Song-Yop Hahn, "A study on comparison of optimization performances between immune algorithm and other heuristic algorithms", IEEE Transactions on Magnetics, Vol. 34, No 5, 1998, pp. 2972-2975.

[11] Chang W. A. and Ramakrishna, R.S., "A genetic algorithm for shortest path routing problem and thesizing of populations", IEEE Transactions on Evolutionary Computation, Vol. 6, No. 6, 2002, pp. 566-579.

[12] Juang C. F., "A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms",IEEE Transactions on Fuzzy Systems, Vol. 10, No. 2, 2002, pp. 155-170.

[13] Ozpineci B., L. M. Tolbert, and J. N. Chiasson, "Harmonic optimization of multilevel converters usinggenetic algorithms", IEEE 35th Annual Power Electronics Specialists Conference, vol.5, 2004, pp. 3911-3916.

[14] Chowdhury S., S. K. Das, and A. Das, "Application of Genetic Algorithm in Communication Network Security", International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 3, No. 1, January 2015.

[15] Mahdad B., T. Bouktir and K. Srairi, "Optimal power Flow of the Algerian Network using Genetic Algorithms/Fuzzy Rules", Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008, pp. 1-8.

[16] Malhotra R., N. Singh & Y. Singh, " Genetic Algorithms: Concepts, Design for Optimization of Process Controllers", Computer and Information Science Vol. 4, No. 2; March 2011, www.ccsenet.org/cis

[17] Kabeer S. J., M. T.Moin, M. A. Rahman, A. Mottalib, M. H. Kabir, "BFSSGA: Enhancing the Performance of Genetic Algorithm using Boosted Filtering Approach", IJCA Journal, Volume 51 - Number 19. 2012.

[18] Ahmad, Z. H., "A Hybrid Genetic Algorithm for the Bottleneck Traveling Salesman Problem", ACM Transactions on Embedded Computing Systems, Vol. 12, No. 1, Article 9, Publication date: January 2013.

[19] A. J. Umbarkar A. J., M. S. Joshi, and P. D. Sheth, "Dual Population Genetic Algorithm for Solving Constrained Optimization Problems", International Journal of Intelligent Systems and Applications(IJISA, IJISA Vol. 7, No. 2, January 2015, PP.34-40, DOI: 10.5815/ijisa.2015.02.05

[20] Moin N. H., O. C. Sin, and M. Omar, "Hybrid Genetic Algorithm with Multiparents Crossover for Job Shop Scheduling Problems, Mathematical Problems in Engineering Volume 2015 (2015), Article ID 210680, 12 pages.http://dx.doi.org/10.1155/2015/210680

[21] Sankaran R., J. Angel and W. M. Brown, "Genetic algorithm based task reordering to improve the performance of batch scheduled massively parallel scientific applications, published on line on 8 APR 2015, DOI: 10.1002/cpe.345, to appear in Concurrency and Computation, Practice and Experience.

[22] Holland, J. H., "Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems, in Computation & intelligence", F.L. George, Editor. 1995, American Association for Artificial Intelligence. pp. 275-304.

[23] Koza, J. R., "Genetic programming: on the programming of computers by means of natural selection", 1992: MIT Press. 680.

[24] G., O.A.M.N.V., "Interactive Design Of Web Sites With A Genetic Algorithm", 2002: Lisbon, Portugal. pp. 355 - 362.

[25] Quiroz, J.C.R.L., S.J. ; Shankar, A. ; Dascalu, S.M., "Interactive Genetic Algorithms for User Interface Design", 2007, IEEE Singapore. pp. 1366 - 1373.

[26] Plessis M. Cd. and L. Barnard, "Incorporating layout managers into an evolutionary programming algorithm to design graphical user interfaces", Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology. 2008, ACM: Wilderness, South Africa. pp. 41-47