International Journal of Computer Science and Applications,
©Technomathematics Research Foundation
Vol. 13, No. 2, pp. 16 – 28, 2016

A NEW METHOD OF EXPLORING A RANGE OF GENETIC PARAMETERS DURING THE EXECUTION OF A GENETIC ALGORITHM

ADAM BYERLY, ALEXANDER USKOV

Department of Computer Science and Information Systems, and InterLabs Research Institute
Bradley University, 1501 West Bradley Avenue
Peoria, Illinois 61625, U.S.A.
auskov@bradley.edu
http://cs.bradley.edu

This paper presents the application of an innovative parameter adaptation strategy for Genetic Algorithms. The strategy involves using the number of generations since an improved solution has been found as an input to a function that oscillates through a range of values for two sets of two genetic parameters. The function oscillates two parameters out of phase with each other and through a wider range of values as the number of generations since the last improvement in solution quality was discovered. As a result, a near continuous and near complete exploration of two dimensions of parameter values can be attempted. Further, in this paper, we present the experimental results obtained from applying this method to solving the Traveling Salesman Problem across fourteen fully connected, symmetric maps of sizes ranging from 52 to 442 cities, and show that a significant improvement is achieved even with relatively constrained amounts of computation time. Among the results that were achieved, we found that the average improvement in error across the fourteen standard maps was 214.15% when the proposed and developed method was used.

Keywords: Travelling Salesman Problem; combinatorial; heuristic; Genetic Algorithm; parameter adaptation

1. Introduction

1.1. *The Travelling Salesperson Problem and Genetic Algorithms*

The Travelling Salesman Problem (TSP) is one of the most studied problems in the class of NP-hard problems [Lawler et al. (1985)]. Largely because it is a good analogue for a broad group of real world problems including planning, logistics, microchip manufacturing, and DNA sequencing [Applegate (2006)]. The TSP, informally articulated, is finding a path (a modified Hamiltonian Circuit) through each city (and returning to the starting city) in a "sales route" such that the path travelled is the shortest possible route for the salesman.

Due to the NP-hard nature of the problem, deterministically solving instances that are not trivially small is intractable. As such, heuristic, stochastic processes, often inspired by nature, are a popular means used to solve instances to near-optimum, non-deterministically. The goal of using stochastic processes to solve NP-hard problems is essentially to find the nearest to optimal possible solution and in the fastest possible time, or at least in a time bounded well below that which would be needed to find an answer

deterministically. Achieving nearest to optimal possible solutions while achieving a reasonable execution time present competing goals.

One of the modern approaches to solving TSP problems is the application of various heuristics based on genetic algorithms (GAs). For example, Larranaga [Larranaga et al. (1999)] reviewed several different approaches to solving TSP with GAs. Particularly, the authors presented crossover and mutation operators to tackle the TPS problem with GAs using different representations such as binary representation, path representation, adjacency representation, ordinal representation and matrix representation. They compared the experimental results obtained from standard TSP problems using different combinations of crossover and mutation operators in relation with path representation.

Carter [Carter, Ragsdale (2005)] presents an applications of GAs to the multiple traveling salesperson problem (MTSP) that involves scheduling $m > 1$ salespersons to visit a set of $n > m$ locations so that each location is visited exactly once while minimizing the total (or maximum) distance traveled by the salespersons. The MTSP is similar to the notoriously difficult traveling salesperson problem (TSP) with the added complication that each location may be visited by any one of the salespersons. The authors proposed a new GA chromosome and related operators for the MTSP. Their research showed that the new approach results in a smaller search space and, in many cases, produces better solutions than previous techniques.

Albayrak [Albayrak, Allahverdi (2011)] presented a new mutation operator that has been developed to increase GA performance to find the shortest distance in the TSP; the authors called this method as Greedy Sub Tour Mutation (GSTM). The GSTM operator that they developed was compared with simple GA mutation operators in 14 different TSP examples selected from TSPLIB. The authors demonstrated that the application of the GSTM operator provided much more effective results for the best and average error values. The GSTM operator used with simple GAs decreased the best error values according to the other mutation operators with the ratio of between 74.24% and 88.32% and average error values between 59.42% and 79.51%.

Snyder [Snyder et al. (2006)] presented an effective heuristic to solve the generalized TSP which is a variation of the well-known TSP in which the set of nodes is divided into clusters. The objective was to find a minimum-cost tour passing through one node from each cluster. The proposed method combines a GA with a local tour improvement heuristic. Solutions are encoded using random keys, which circumvent the feasibility problems encountered when using traditional GA encodings. On a set of 41 standard test problems with symmetric distances and up to 442 nodes, the heuristic found solutions that were optimal in most cases and were within 1% of optimality in all but the largest problems, with computation times generally within 10 seconds.

1.2. *Genetic Algorithms Applications to Solve Multimedia Problems*

In addition to studying GAs as applied to the TSP, there are publications available on applications of GAs to solve various multimedia-related problems. For example, Zhang [Zhang et al (1999)] discussed multimedia communication applications that require a source to send multimedia information to multiple destinations through a communication network. To support these applications, it is necessary to determine a multicast tree of minimal cost to connect the source node to the destination nodes subject to delay

constraints on multimedia communication. This problem is known as multimedia multicast routing and has been proven to be NP-complete. The authors proposed an orthogonal genetic algorithm for multimedia multicast routing; its salient feature is to incorporate an experimental design method called orthogonal design into the crossover operation. As a result, it can search the solution space in a statistically sound manner and it is well suited for parallel implementation and execution. The results of their research indicated that, for practical problem sizes, the proposed orthogonal GA can find near optimal solutions within moderate numbers of generations.

Chiu [Chiu et al (2000)] presented an application of GAs for video segmentation and summarization, indexing for browsing, and adapting to user access patterns. The evolutionary nature of GAs offers a further advantage by enabling incremental video segmentation. The proposed genetic segmentation algorithm operates on segments of a video string representation. It is similar to both classical GAs that operate on bits of a string and genetic grouping algorithms that operate on subsets of a set. The authors also defined similarity adjacency functions which are extremely expensive to optimize with traditional methods.

Kumsawat [Kumsawat et al (2005)] proposed the spread spectrum image watermarking algorithm using the discrete multiwavelet transform. The performance improvement they achieved with respect to existing algorithms was obtained by active use of GA-based optimization. In the proposed optimization process, the authors search for parameters that consist of threshold values and the embedding strength to improve the visual quality of watermarked images and the robustness of the watermark. These parameters are varied to find the most suitable for images with different characteristics. The experimental results show that the proposed algorithm yields a watermark that is invisible to human eyes and robust to various image manipulations.

These and multiple other publications on GAs and their application for engineering problems, including multimedia ones, clearly show that much of the current research in GAs for TSP and other scheduling problems in engineering focuses more on achieving more optimal solutions than on execution time. In addition, in the case of both online and offline adaptation of the genetic parameters, the comprehensive literature analysis clearly shows that currently multiple research projects are focused on adapting the parameters throughout the duration of the algorithm's execution, rather than using a more tactical approach.

1.3. *Project Goal*

This research project was focused on 1) adaptation of the GA parameters as a tactical solution to early GA stagnation, and 2) times of GA stagnation, rather than throughout the duration of the entire algorithm's execution. The premise being that when successive generations of a GA produce improved solutions, the genetic parameters are appropriate, but when stagnation sets in, other values for the genetic parameters should be attempted.

The research outcomes (data), findings, conclusions and recommendations are presented below.

The rest of this paper consists of the following parts. In Section 2, a brief description of adapting GA parameters upon stagnation is described. In Section 3 the specifications of the research environment are provided, and in Section 4 – research outcomes regarding

the efficiency of the proposed adaptation heuristics for GAs. Finally, in Section 5 discussion and next steps of this research project are presented.

2. Adapting GA Parameters upon Stagnation

The current research in both offline and online adaptation of genetic parameters has shown to be effective and is indeed promising; however, we wanted to research adaptation of the parameters as a tactical solution to early stagnation, localized to times of stagnation, rather than throughout the duration of the entire algorithm's execution.

2.1. *Hypothesis*

Our proposed method is based on the following hypothesis:

(1) At the onset of stagnation, we should begin oscillating the values of the (a) reproduction rate and the survival rate out of phase with each other, as well as the values of the (b) chromosome mutation rate and the gene mutation rate out of phase with each other.
(2) We should begin oscillating mildly and then slowly increase the amplitude of the oscillation only as the stagnation continues.
(3) We should cease the oscillation when a better solution is found and return to the original parameter values for the reproduction rate, survival rate, chromosome mutation rate, and gene mutation rate.
(4) If stagnation is again encountered, we repeat the process.

This process continues until the algorithm's stopping condition is met.

2.2. *Implementation*

Given that we wanted to oscillate through a range of values, we designed a function based on the sine function as follows:

(1) The input to the function (x) represents the number of iterations of the algorithm that have not yielded an improved solution.
(2) We added coefficients to both the input and output of the sine function so that we could control for the speed (B) and the amplitude (A) of the oscillation.
(3) We specified a vertical offset (O_V) from the y-axis and a horizontal offset (O_H) from the x-axis.
(4) Finally, as we wanted to oscillate mildly at first, and increase the oscillation rate as stagnation continued, we added a term to the equation and a parameter to control the strength of the amount the oscillation is initially dampened (ζ).

The resulting function is described by equation (1):

$$f(x) = \left(\frac{\zeta}{x+\zeta} - 1\right) \cdot [O_{\mathrm{v}} + A\sin(O_H + Bx)] \tag{1}$$

We will refer to instances of this function in the following text as *accelerators*.

3. Research Environment

3.1. *Research hardware setup*

One of the objectives of this research project was to effectively use the computing power needed to run hundreds of related tests. It was estimated that for accurate simulation and reliable research data we would need to run 1) 14 different maps, 2) a maximum of 10,000 seconds per map, and 3) 54 experiments per map. In other words, we would need up to 7,560,000 seconds (or 2,100 hours, or 87.5 full 24-hour days) to execute all of the required experiments on one computer system without any interruption (or, other parallel processes).

Rather than running on a single computer system for nearly 3 months, we chose to setup a research environment consisting of 20 workstations with identical hardware setup and with identical operating system and software images. Each workstation had a single Intel Core i7 920 4 core/8 thread 2.67 GHz processor with 12 GB of RAM and was imaged with the Microsoft Windows 7 operating system.

3.2. *Research software setup*

The software we developed to execute our experiments was written in standard C++ and compiled with Microsoft's Visual Studio 2013 C++ Compiler. The software was designed to query the operating system for available concurrency and then to allow each experiment to create that many threads for executing the sections of the code that were amenable to parallel execution. Each iteration in a GA must run after the prior in serial, however, within a generation, each chromosome is independent of the others. As such, the number of chromosomes (1000 in our experiments) were divided among the available hardware threads (8 on each workstation in our research environment). Additionally, as our research environment consisted of 64-bit operating systems, the software we developed was compiled for 64-bit execution.

The software was designed so that upon initialization, it would read a configuration file that included the experiments it should run. We then took the 756 experiments we wanted to run and divided them evenly (based on their allowed execution times) into 19 groups and created configuration files for them. We copied the executable code and one of the configuration files to each of the 19 workstations. We then, from an arbitrarily chosen workstation, used a freely downloadable tool from the *SysInternals Suite* [Microsoft Corporation (2016)], originally developed by Mark Russinovich, and now distributed by Microsoft. The tool, *PsExec*, allows for remote execution of code from an authorized central location. This allowed us to use Windows workstations as an ad-hoc "cluster" without having to physically interact with each "node" on the cluster, but rather from just a single workstation.

This design for our experiments, allowed them all to complete in under a total of 110 hours, rather than the originally expected 2,100 hours. After all experiments had finished, we gathered the separate outputs created by the executable code on each machine and collated them into a single spreadsheet. We then used Microsoft Excel to analyse the data and compute the minimums, maximums, averages, and standard deviations.

3.3. *Test data sets used*

One of the objectives of this research was to use the standard testing data sets used to evaluate TSP problem solutions. As a result, we used the standard TSP testing datasets from TSPLIB [Reinelt (2015)], which all have a deterministically computed optimum available. This allowed us to determine the differences between the solutions arrived at by our GA code, both with and without the proposed accelerators enabled, with the actual optimum. This approach also gives us the ability to compute an error measurement.

For our experiments, we chose 14 datasets of increasing city count, specifically: berlin52, kroA100, pr144, ch150, kroB150, pr152, rat195, d198, kroA200, ts225, pr226, pr299, lin318, and pcb442. (Note that these data set names are reproduced exactly as they were given on the TSPLIB website and will be referred to by those names in the results below.)

We chose a union of three possible conditions for the algorithm to stop on:

(1) for all experiments, the total number of generations was limited to 20,000;
(2) for all experiments, the number of generations of stagnation was limited to 2,000;
(3) for all experiments, we limited the duration to 10,000 seconds of clock time.

As 2,000 generations of stagnation was a stopping condition, we chose our constant for parameter ζ in equation (1) so that the oscillation would grow steadily starting with the first generation of stagnation to the peak of the range we wanted to oscillate within at around 2,000 generations. This value for parameter ζ was 5,000.

The following parameter values were used for all experiments:

(1) an initial reproduction rate of 0.5;
(2) an initial survival rate of 0.5;
(3) an initial chromosome mutation rate of 0.001;
(4) an initial chromosome mutation rate of 0.001;
(5) a population size of 1000.

In order to achieve a continually changing phase relationship between each of the two sets of our two accelerator functions, we chose a ratio of 15:16 for the two values of the B parameters to our functions, which represent the speeds of the oscillation. Equations (2) and (3) show the functions derived from these desired characteristics for our survival and reproduction rate accelerators, respectively.

$$f(x) = \left(\frac{5000}{x + 5000} - 1\right) \cdot \left[0.85 + 0.85 \sin\left(-\frac{\pi}{2} + 0.03333333x\right)\right] \quad (2)$$

$$f(x) = \left(\frac{5000}{x + 5000} - 1\right) \cdot \left[0.85 + 0.85 \sin\left(-\frac{\pi}{2} + 0.03792593x\right)\right] \quad (3)$$

In addition to the survival and reproduction rate accelerators, we implemented similar functions for our chromosome mutation rate and gene mutation rate accelerators as shown in equations (4) and (5) respectively:

$$f(x) = \left(\frac{5000}{x+5000} - 1\right) \cdot \left[-0.37 - 0.37 \sin\left(-\frac{\pi}{2} + 0.03125x\right)\right] \quad (4)$$

$$f(x) = \left(\frac{5000}{x+5000} - 1\right) \cdot \left[-0.37 - 0.37 \sin\left(-\frac{\pi}{2} + 0.03555556x\right)\right] \quad (5)$$

3.4. *Legend used*

The following legend has been used in all experiments and research data outcomes given below in Tables 1-3.

Map. This is the name of a particular map used in the experiment from TSPLIB.

Map Size. This refers to the number of cities in the map used. Consequently, since all of the graphs are fully connected and symmetrical, this means that the number of possible solutions for a map of size *n* is equal to

$$\frac{(n-1)!}{2} \quad (6)$$

Acceleration Enabled. The value in this column is 'Yes', when the GA was running with the accelerators enabled, and 'No' when not. Each set of experiments was performed once with accelerators enabled and once without to provide a control to compare with.

Avg, Max, Min, and StdDev. These are the average, maximum, minimum, and standard deviation of the computed error aggregated across the 25 experiments.

Optimum. This is the theoretical optimum for the graph as computed deterministically and reported by TSPLIB.

Avg, Max, and Min. These are the average, maximum, and minimum of the values the algorithm discovered aggregated across the 25 experiments that each row in the table represents.

Avg Iterations at Stop. This is the average number of generations the GA ran for before arriving at the value the algorithm would ultimately report. If an algorithm arrived at its most optimum value on the y^{th} generations, but ran for $y+n$ generations, y is the value that is reported.

Avg Duration. This is the average number of seconds the experiments ran for. Note that despite having time limits set, the averages on the larger maps exceed the time limits. This is because on the larger maps, a single iteration ran for many seconds, and we enforced the time limits after a finished iteration, rather than interrupting an iteration in progress.

Stagnation Stop. This is the number of the 25 experiments for which the GA stopped because the best found solution had stagnated for 2,000 generations.

Duration Stop. This is the number of the 25 experiments for which the GA stopped because the algorithm had run for 10,000 seconds.

4. Research Outcomes

All 576 planned tests were executed in the month of February of 2016 at the Networking Computer Laboratory, the Department of Computer Science and Information

Systems, Bradley University (Peoria, IL, U.S.A.) with 20 computer systems – each as described in section 3.1 above.

The following procedure was repeated twice, once with the accelerators enabled, and once without the accelerators enabled:

(1) 27 experiments were conducted for each of the designated 14 data sets;
(2) we ordered the 27 results of those experiments by the reported solution's error from optimum;
(3) the highest and lowest errors were removed, leaving 25 results for further analysis.

A summary of obtained research data for different maps, map sizes, and modes of accelerators is presented in Tables 1-3 and Figures 1-2.

Table 1 below summarizes the error of the values the proposed GA discovered vs. the theoretical optimum. Error is computed as shown in equation (7)

$$\left[\left(\frac{x_{\mathrm{A}}}{x_{\mathrm{O}}}\right) - 1\right] * 100 \tag{7}$$

where x_{A} is the value computed by the GA and x_{O} is the optimum value as computed by deterministic methods as published in TSPLIB.

Table 2 summarizes the actual values the algorithm discovered.

Table 3 summarizes the number of iterations the algorithm ran for, the duration of the experiments' execution, and the stopping condition that terminated the algorithms' execution.

Table 1. Error vs. Optimum values of *Avg, Max, Min,* and *StdDev*

Map	**Map size**	**Acceleration Enabled**	***Avg***	***Max***	***Min***	***StdDev***
berlin52	52	No	0.688	4.875	0.031	1.227
berlin52	52	Yes	0.412	3.425	0.031	0.961
kroA100	100	No	6.008	9.395	3.374	1.568
kroA100	100	Yes	3.000	6.740	0.770	1.482
pr144	144	No	49.727	75.786	25.920	13.102
pr144	144	Yes	15.295	31.084	2.869	7.501
ch150	150	No	18.936	34.940	11.356	7.451
ch150	150	Yes	8.707	17.208	4.017	3.460
kroB150	150	No	42.418	55.247	13.223	11.320
kroB150	150	Yes	16.154	23.460	8.196	4.587
pr152	152	No	49.113	68.200	16.476	12.839
pr152	152	Yes	14.199	23.895	3.947	4.878
rat195	195	No	72.423	89.680	61.299	7.015
rat195	195	Yes	29.995	36.114	25.110	3.011
d198	198	No	38.830	54.529	20.856	11.972
d198	198	Yes	18.160	24.236	12.976	3.348
kroA200	200	No	81.616	93.274	69.045	6.827
kroA200	200	Yes	34.284	41.231	27.599	3.846
ts225	225	No	109.077	122.080	88.461	8.330
ts225	225	Yes	54.618	69.709	40.858	7.159
pr226	226	No	113.714	138.955	92.750	12.880
pr226	226	Yes	65.850	84.636	51.737	7.437
pr299	299	No	147.879	168.497	128.555	10.080

pr299	299	Yes	94.818	104.256	79.884	6.014
lin318	318	No	135.020	151.189	118.382	9.696
lin318	318	Yes	96.720	104.179	87.515	4.830
pcb442	442	No	216.847	243.953	196.667	11.049
pcb442	442	Yes	183.114	200.469	166.320	7.995

Table 2. Values Discovered with Theoretical Optimum

Map	**Map size**	**Acceleration Enabled**	***Optimum***	***Avg***	***Max***	***Min***
berlin52	52	No	7542	7593.89	7909.65	7544.37
berlin52	52	Yes	7542	7573.07	7800.28	7544.37
kroA100	100	No	21282	22560.54	23281.50	22000.10
kroA100	100	Yes	21282	21920.37	22716.40	21445.80
pr144	144	No	58537	87645.55	102900.00	73709.80
pr144	144	Yes	58537	67490.02	76732.90	60216.30
ch150	150	No	6528	7764.12	8808.87	7269.34
ch150	150	Yes	6528	7096.40	7651.35	6790.25
kroB150	150	No	26130	37213.71	40565.90	29585.00
kroB150	150	Yes	26130	30351.16	32260.20	28271.60
pr152	152	No	73682	109869.61	123933.00	85821.50
pr152	152	Yes	73682	84143.80	91288.30	76590.00
rat195	195	No	2323	4005.38	4406.27	3746.98
rat195	195	Yes	2323	3019.79	3161.92	2906.31
d198	198	No	15780	21907.38	24384.70	19071.10
d198	198	Yes	15780	18645.70	19604.50	17827.60
kroA200	200	No	29368	53336.87	56760.60	49645.00
kroA200	200	Yes	29368	39436.42	41476.80	37473.30
ts225	225	No	126643	264781.88	281249.00	238673.00
ts225	225	Yes	126643	195812.68	214925.00	178386.00
pr226	226	No	80369	171759.60	192046.00	154911.00
pr226	226	Yes	80369	133292.13	148390.00	121950.00
pr299	299	No	48191	119455.25	129391.00	110143.00
pr299	299	Yes	48191	93884.93	98432.80	86688.00
lin318	318	No	42029	98776.48	105572.00	91783.90
lin318	318	Yes	42029	82679.59	85814.50	78810.50
pcb442	442	No	50778	160888.88	174653.00	150642.00
pcb442	442	Yes	50778	143759.50	152572.00	135232.00

Table 3. Iterations at Stop, Duration, and Stopping Condition

Map	**Map size**	**Accele-ration Enabled**	***Avg. Iterations at Stop***	***Avg. Duration***	***Total Stop***	***Stag-nation Stop***	***Duration Stop***
berlin52	52	No	128	225	0	25	0
berlin52	52	Yes	254	245	0	25	0
kroA100	100	No	782	371	0	25	0
kroA100	100	Yes	4040	846	0	25	0
pr144	144	No	6438	1354	1	24	0
pr144	144	Yes	14663	2823	3	22	0
ch150	150	No	1760	682	0	25	0
ch150	150	Yes	10180	2308	0	25	0
kroB150	150	No	3082	908	0	25	0
kroB150	150	Yes	14441	3005	6	19	0
pr152	152	No	8621	1852	3	22	0
pr152	152	Yes	16338	3334	8	17	0
rat195	195	No	6529	1873	1	24	0

rat195	195	Yes	19269	4608	23	2	0
d198	198	No	8657	2377	2	23	0
d198	198	Yes	19200	4621	22	3	0
kroA200	200	No	7405	2242	1	24	0
kroA200	200	Yes	19732	4893	25	0	0
ts225	225	No	6274	2152	1	24	0
ts225	225	Yes	19656	5392	25	0	0
pr226	226	No	17936	4906	18	7	0
pr226	226	Yes	19792	5309	24	0	0
pr299	299	No	16367	6971	14	10	0
pr299	299	Yes	19875	8135	24	0	0
lin318	318	No	18511	8260	20	4	0
lin318	318	Yes	19823	8770	24	0	0
pcb442	442	No	14083	10000	0	0	24
pcb442	442	Yes	13922	10000	0	0	24

Fig. 1 presents a graph of the aforementioned functions (2) and (3) over 2,000 generations of stagnation; in this case the solid line represents the outcomes for function (2) and the dashed line – for function (3). The generations of stagnation are shown along the *X* axis, and the amount added to the survival rate (solid line) and to the reproduction rate (dashed line) are shown on the *Y* axis.

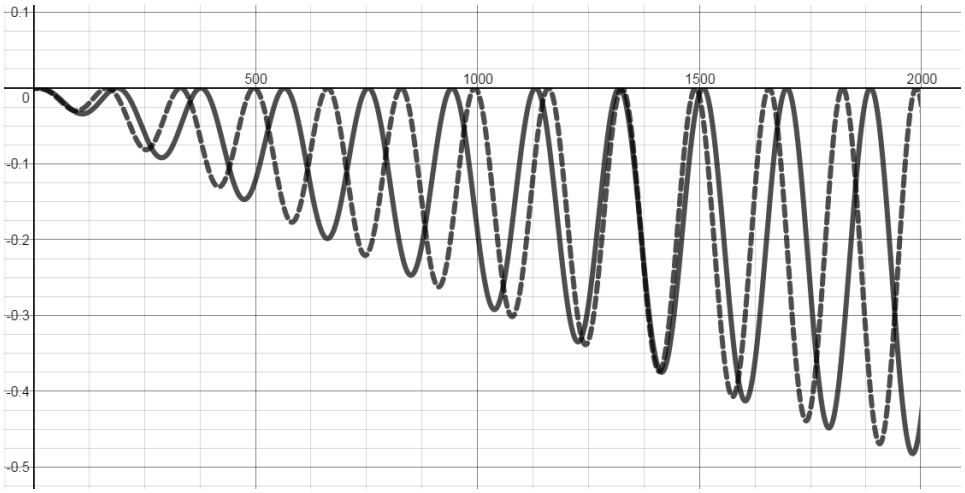

Fig. 1. Plot of Survival and Reproduction Rate Accelerators

Fig. 2 presents a graph of the aforementioned functions (4) and (5) over 2,000 generations of stagnation; in this case the solid line represents the outcomes for function (4) and the dashed line – for function (5). The generations of stagnation are shown along the *X* axis. The amount added to the chromosome mutation rate (solid line) and to the gene mutation rate (dashed line) are shown on the *Y* axis.

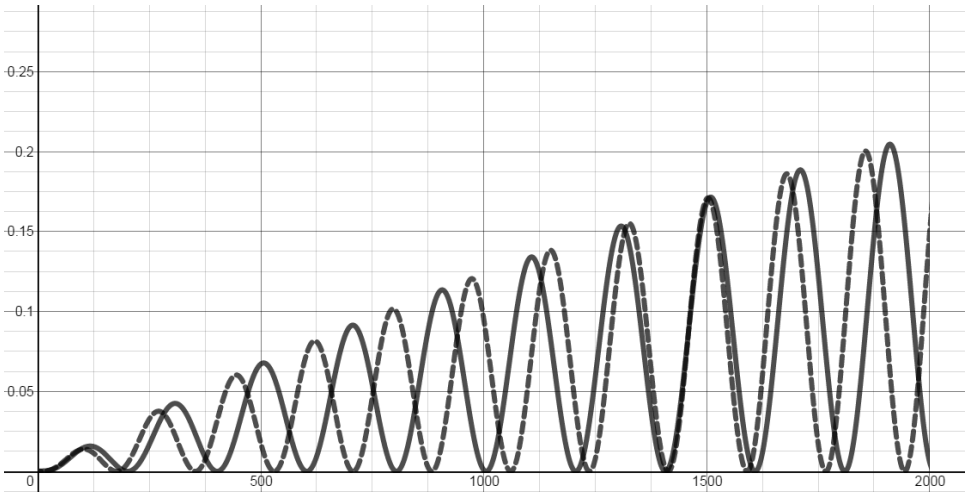

Fig. 2. Plot of Chromosome Mutation and Gene Mutation Rate Accelerators

5. Conclusions and Next Steps

5.1. *Conclusions*

The ongoing research project is focused on 1) adaptation of the GA parameters as a tactical solution to early GA stagnation, and 2) times of GA stagnation, rather than throughout the duration of the entire GA's execution. The premise being that when successive generations of a GA produce improved solutions, the genetic parameters are appropriate, but when stagnation sets in, other values for the genetic parameters should be attempted.

The research data obtained from numerous executed experiments clearly proves the effectiveness of the proposed and developed innovative parameter adaptation strategy for GAs. The strategy involves using the number of generations since an improved solution has been found as an input to a function that oscillates through a range of values for two sets of two genetic parameters. The function oscillates two parameters out of phase with each other and through a wider range of values as the number of generations since the last improvement in solution quality was discovered. As a result, a near continuous and near complete exploration of two dimensions of parameter values can be attempted.

576 comprehensive tests were executed to verify the correctness and effectiveness of the proposed innovative parameter adaptation strategy for GAs. The experimental results obtained from applying this strategy to solving the TSP across fourteen fully connected, symmetric maps of sizes ranging from 52 to 442 cities, clearly show that a significant improvement is achieved even with relatively constrained amounts of computation time. Particularly, all 14 maps experienced:

(1) a better average error when accelerators were enabled as compared to when they were not; the average improvement in error across the 14 maps was 214.15%.

(2) a better maximum error when accelerators were enabled as compared to when they were not; the average improvement in maximum error across the 14 maps was 194.06%.
(3) a better minimum error when accelerators were enabled as compared to when they were not; the average improvement in minimum error across the 14 maps was 269.17%.
(4) a smaller standard deviation when accelerators were enabled as compared to when they were not. The average standard deviation when accelerators were not enabled was 8.954; the average standard deviation when accelerators were enabled was 4.751.

5.2. *Next Steps*

The next steps in this research project include but are not limited to:

(1) Applying the developed strategy to problems
    a. with larger solution spaces, and
    b. in various applied areas such as optimization of network traffic scheduling, optimization of a structure and data packets transfer in virtual private networks (VPNs), for example, multi-VPN optimization for scalable routing [Bateni (2010)], a greedy randomized adaptive search procedure with a biased random-key genetic algorithm to solve multi-layer network optimization problem [Pedrola (2013)], optimization of VPN site selection [Scharf (2013)], and other areas;
(2) Follow a design of experiments methodology in order to discover:
    a. proper genetic parameter settings when the method is in use, which may differ from genetic parameter settings when the method is not in use;
    b. proper values for the parameterization of the accelerator functions.
(3) Analyze the relationship between problem size and parameterization of the accelerator functions.
(4) Analyze the runtime, reported solutions, and behavior of the accelerator functions in the absence of run time limits and total generation limits.

References

Albayrak, M., Allahverdi, N., (2011) Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms, Expert Systems with Applications, vol. 38, Issue 3, Mar. 2011, pages 1313–1320, Elsevier.

Applegate, D.L. (2006) The Traveling Salesman Problem: A Computational Study. Princeton: Princeton UP, U.S.A.

M. Bateni, A. Gerber, M. Hajiaghayi, S. Sen (2010) Multi-VPN Optimization for Scalable Routing via Relaying, IEEE/ACM Transactions on Networking, vol. 18, no. 5, ACM (2010), pp. 1544 – 1556

Carter, A., Ragsdale, C. (2006) A new approach to solving the multiple traveling salesperson problem using genetic algorithms, European Journal of Operational Research, vol. 175, issue 1, 16, Nov. 2006, , Elsevier, pp. 246–257

Chiu, P., Girgensohn, A., Polak, W., Rieffel, E., Wilcox, L. (2000) A genetic algorithm for video segmentation and summarization, 2000 IEEE International Conference on Multimedia and Expo - ICME 2000, vol. 3, pp. 1329-1332, New-York, NY, U.S.A., IEEE.

Kumsawat, P., Attakitmongcol, K., Srikaew, A. (2005) A new approach for optimization in image watermarking by using genetic algorithms, IEEE Transactions on Signal Processing, Nov, 2005, vol. 53, issue 12, pp. 47-7-4719, IEEE.

Larrañaga, P. , Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S. (1999) Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators, Artificial Intelligence Review, Apr. 1999, vol. 13, issue 2, pp 129-170, Springer.

Lawler, E.L., Lenstra,, J.K., Rinooy Kan, A.H.G., Shmoys, D.B. (1985) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, Chichester, U.S.A.

Pedrola, O., Ruiz, M., Velasco, L., Careglio, D., González de Dios, O., Comellas, J. (2013) A GRASP with path-relinking heuristic for the survivable IP/MPLS-over-WSON multi-layer network optimization problem, Computers & Operations Research, vol. 40, no. 12, Elsevier, pp. 3174–3187

Reinelt, G. TSPLIB, available at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ Universität Heidelberg (Accessed 26 June 2015).

Scharf, M., Gurbani, V., Voith, T., Stein, M., Roome, W., Soprovich, G. (2013) Hilt, V., Dynamic VPN Optimization by ALTO Guidance, The Second European Workshop on Software Defined Networks (EWSDN), Alcatel-Lucent.

Snyder, L., Daskin, M. (2006) A random-key genetic algorithm for the generalized traveling salesman problem, European Journal of Operational Research, vol. 174, Issue 1, 1 Oct. 2006, pages 38–53, Elsevier.

Microsoft Corporation (2016). Windows SysInternals, available at https://technet.microsoft.com/en-us/sysinternals/bb842062.aspx (Accessed 12 February 2016).

Zhang, Q., Leung, Y.-W. (1999) An orthogonal genetic algorithm for multimedia multicast routing, IEEE Transactions on Evolutionary Computation, Apr. 1999, vol. 3, issue 1, pp. 53-62, IEEE