Evolving Hidden Genes in Genetic Algorithms For Systems Architecture Optimization

Ossama Abdelkhalik*
Shadi Darani
Department of Mechanical Engineering and Engineering Mechanics
Michigan Tech Univesity
Houghton, Michigan 49931

The concept of hidden genes was recently introduced in genetic algorithms to handle systems architecture optimization problems where the number of design variables is variable. Selecting the hidden genes in a chromosome determines the architecture of the solution. This paper presents two categories of mechanisms for selecting (assigning) the hidden genes in the chromosomes of genetic algorithms. These mechanisms dictate how the chromosome evolve in the presence of hidden genes. In the proposed mechanisms, a tag is assigned for each gene; this tag determines whether the gene is hidden or not. In the first category of mechanisms, the tags evolve using stochastic operations. Eight different variations in this category are proposed and compared through numerical testing. The second category introduces logical operations for tags evolution. Both categories are tested on the problem of interplanetary trajectory optimization for a space mission to Jupiter, as well as on mathematical optimization problems. Several numerical experiments were designed and conducted to optimize the selection of the hidden genes algorithm parameters. The numerical results presented in this paper demonstrate that the proposed concept of tags and the assignment mechanisms enable the hidden genes genetic algorithms to find better solutions.

1 Introduction

Systems architecture optimization problems arise in several applications such as in automated construction (in which hundreds or thousands of robots fabricate large, complex structures), autonomous emergency response, and smart buildings, transportation, medical technology, and electric grids [1]. In these complex systems, the automated system design optimization is crucial to achieve design objectives. The task of design optimization includes optimizing the system architecture (topology) in addition to the system variables. Optimizing the system architecture renders the problem a Variable-Size Design Space (VSDS) optimization problem (the number of design variables to be optimized is a variable.) Consider, for example, the optimization of a space interplanetary trajectory. The objective is to design a trajectory for a spacecraft to travel from the home planet to the target planet with, for instance, a minimum fuel consumption. As can be seen in Figure 1, the spacecraft can apply Deep Space Maneuvers (DSMs) which are propulsive impulses used to change the velocity of the spacecraft instantaneously; these DSMs consume fuel proportional to the amount of the DSMs impulse. The spacecraft can also benefit from free change in momentum, through as many as needed fly-bys of other planets. When the spacecraft performs a fly-by maneuver, we need to determine the height of closest approach to the fly-by planet as well as the plane of the fly-by maneuver. Hence, by changing the number of fly-bys the total number of variables change.

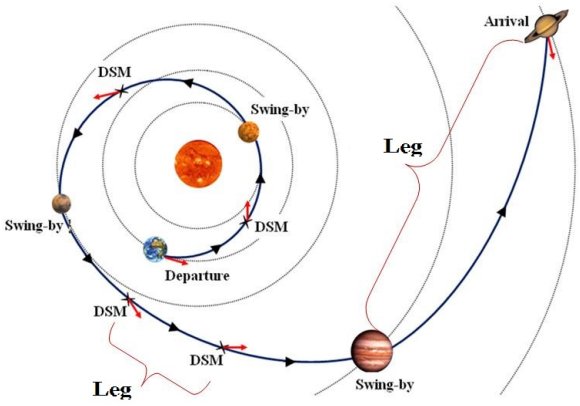

Fig. 1: Interplanetary Trajectory Optimization Problem Topology

The segment between any two planets is called a leg. A leg can have any number of DSMs. The architecture of a solution refers to the sequence of fly-bys and the number of DSMs in each leg. The determination of the mission architecture then means the determination of the number of fly-bys, the planets of fly-bys, and the number of DSMs in each leg. Other non architecture variables include launch and ar-

*Email: ooabdelk@mtu.edu

rival dates, dates and times of fly-bys, dates and times of DSMs, amounts and directions of DSMs impulses. This is a VSDS optimization problem.

Another example is the optimization of a microgrid system where there are several energy sources and co-located energy storage devices that can either sink or source power with their corresponding sources. The net power at each source/storage is metered to the grid main bus using a boost converter. For an efficient design of the microgrid, the number of storage elements (N) and their capacities need to be optimized. Storage is expensive and designing a microgrid, with storage sized properly, is an open problem. Associated with computing the optimal N is the optimal values for the duty ratios at the converters that controls the power metered to the main bus from each source. A more complex situation is when we have M microgrids that have the ability to interconnect. This provides a large number of permutations for exchanging power.

Systems design optimization problems are usually replete with local minima. Hence a global search algorithm is usually needed for optimizing the system variables, such as genetic algorithms (GAs) [2], particle swarm optimization [3], ant colony optimization [4], or differential evolution [5]. In VSDS optimization, the problem can be formulated as follows:

$$\begin{aligned} &\text{Minimize} \quad f(\vec{x},N) \\ \text{Subject to} \quad &\vec{g}(\vec{x}) \leq \mathbf{0}, \quad \vec{h}(\vec{x}) = \mathbf{0}, \quad \vec{x}^l \leq \vec{x} \leq \vec{x}^u \end{aligned} \tag{1}$$

where $\vec{x} = [x_1, x_2, ..., x_N]^T$, N is the number of design variables, $\vec{x}^u$ and $\vec{x}^l$ are the upper and lower bounds of the variables $\vec{x}$, respectively. The number of variables N in this formulation is variable, and its value dictates the architecture of the solution. The number of inequality constraints $\vec{g}$ and the number of equality constraints $\vec{h}$, each is also a variable.

The research on developing algorithms that can handle VSDS optimization problems (sometimes referred to as variable length optimization) has started since about two decades. GAs are not suitable for VSDS problems because they are designed to work only on problems of fixed number of variables. Reference [6] presents a variable length genetic programming, and compares it to the simulated annealing and the stochastic iterated hill climbing methods, on program discovery problems. A VSDS GA is presented in [7] in which a random operator is introduced to change the chromosome length, for the problem of Kauffman NK model. This random operator depends on the identity of genes which is given by their position relative to one end of the genotype. Reference [8] is a continuing work of [7] and analyzes the optimal location for the crossover point in VSDS problems. When two parents have different chromosome lengths, and given a selection for the crossover point in parent 1, reference [8] suggests that the crossover point in parent 2 be chosen such that the difference between the swapped segments is minimized. The method proposed in [8] is a search on all the possible crossover points in parent 2 to find the best cutoff point. The VSDS GA in reference [9] uses a two-point crossover, with different cutoff points in each parent, resulting in different lengths of the children chromosomes. This method is most useful in problems with variables of the same identity, like angles of a polyhedral where adding or removing one angle will result in a new polyhedral (e.g. triangle to rectangle or vice versa). Reference [10] presents a number of variable length representation evolutionary algorithms that improves the sampling of a VSDS, with application in evolutionary electronics. In reference [11], the number of different chromosome lengths is set a priori, and both parents have the same crossover point (same gene index of cutoff). Therefore the length of the chromosome is switched from parents to children in [11] (the length of child 2 is equal to length of parent 1 and length of child 1 is equal to length of parent 2). This method does not provide information regarding the optimal length of a solution. A different approach in VSDS GA is to have equal-length chromosomes in each generation, yet the chromosome length is allowed to change among different generations as presented in [12, 13]. In this method, the GA starts with short-length chromosomes and the best solution in a generation is transferred to the next generation with a longer chromosome length. In this way, the GA handles fixed-size chromosomes in each generation, and there is no need to define new evolutionary operations for GA. A dynamic-size multiple population genetic algorithm was developed in [14] where each generation consists of a number of sub-populations; all chromosomes in each sub-population are of the same length. Hence each sub-population evolves over subsequent generations as in a standard GA. The size of each sub-population, however, changes dynamically over subsequent generations such that more fit sub-populations are allowed to increase in size whereas lower fit sub-populations decreases in size. This approach has been applied to the trajectory optimization problem and demonstrated success in finding best know solution architectures. The computational cost of this method, however, is relatively high since it implements GA over several sub-populations in parallel. Also, only a finite number of architectures (assumed a priori) can be investigated using the method in [14]. A structured chromosome genetic algorithm was developed in [15, 16] where the standard one layer chromosome is replaced with a multi-layer chromosome for coding the variables; the number of genes in one layer is dictated by the values of some of the genes in the upper layers. Hence, it was possible to code solutions of different architectures. Yet, this structured-chromosome approach introduces new definitions for the crossover operation such that meaningful swapping between chromosomes of different layers is guaranteed. Some other algorithms are designed for specific problems. For instance, references [17] and [18] present tailored algorithms that search for the optimal structural topology in truss and frame structures, respectively. The dissertation in [19] presents a study on topology optimization of nanophotonic devices and makes a comparison between the homogenization method [20] and genetic algorithms [2]. As can be seen from the above discussion, many of the VSDS optimization algorithms are problem specific. The dynamic-size multiple population genetic algorithm has a high compu-

tational cost [14]. The structured chromosome genetic algorithm is relatively complex to develop since it requires new definitions for all GAs operations.

Inspired by the concept of hidden genes in biology, reference [21] presented the method of Hidden Genes Genetic Algorithms (HGGA) for solving VSDS optimization problems. In space trajectory optimization, the HGGA successfully found the best known solution architectures as reported in [22, 23]. Genetic algorithms is one of the methods being used for systems architecture optimization problems that are not VSDS. It is because the HGGA is based on genetic algorithms that it can handle systems architecture optimization problems. The added capability of the HGGA is that it can optimize among different solution architectures and can also develop new architectures that might not be known a priori, and hence it can handle VSDS problems. The method used in reference [21] to determine which genes are hidden in each chromosome in each generation, however, was very primitive. In [21], genes in a chromosome will only be hidden if a chromosome represents a non-feasible solution, Hence, the HGGA will not attempt to hide genes if the chromosome is a feasible solution. Subsequent developments on HGGA has introduced tags for the genes [24], where each gene is assigned a binary tag that determines whether it is hidden or not; hence a gene can be hidden even in feasible solutions if hiding that gene results in a more fit solution. In this paper, the problem of selecting the hidden genes in each generation is addressed. This paper develops mechanisms for the tags to evolve over generations. Two new concepts for hidden genes selection are presented in this paper. Section 2 presents a review for the most recent developments on HGGA. Section 3 presents the new methods of tags evolution. Section 4 presents VSDS mathematical functions and tests on the tags evolution methods. Section 5 presents the results of implementing these methods to solve an interplanetary space trajectory optimization problem. Finally, Section 6 presents a statistical analysis for the proposed methods.

2 Hidden Genes Genetic Algorithms

Section 2.1 presents necessary background material. Section 2.2 is a brief description from reference [21] that shows how HGGA works.

2.1 The Hidden Genes Concept in Biology

In genetics, the deoxyribonucleic acid (DNA) is organized into long structures called chromosomes. Contained in the DNA are segments called genes. Each gene is an instruction for making a protein. These genes are written in a specific language. This language has only three-letter words, and the alphabet is only four letters. Hence, the total number of words is 64. The difference between any two persons is essentially because of the difference in the instructions written with these 64 words. Genes make proteins according to these words. Since, not all proteins are made in every cell, not every gene is read in every cell. For example, an eye cell doesn't need any breathing genes on. And so they are shut off in the eye. Seeing genes are also shut off in the lungs. Another layer of coding tells what genes a cell should read and what genes should be hidden from the cell [25]. A gene that is being hidden, will not be transcribed in the cell. There are several ways to hide genes from the cell. One way is to cover up the start of a gene by chemical groups that get stuck to the DNA. In another way, a cell makes a protein that marks the genes to be read; Figure 2 is an illustration for this concept. Some of the DNA in a cell is usually wrapped around nucleosomes but lots of DNA are not. The locations of the nucleosomes can control which genes get used in a cell and which are hidden [25].

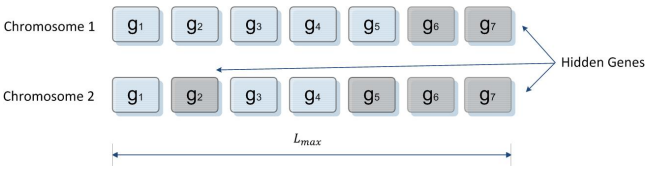

Fig. 3: Hidden genes and effective genes in two different chromosomes

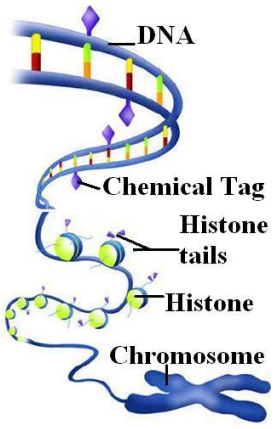

Fig. 2: Chemical tags (purple diamonds) and the "tails" of histone proteins (purple triangles) mark DNA to determine which genes will be transcribed. (picture is modified from [26])

2.2 Concept of Optimization Using Hidden Genes Genetic Algorithms

In the HGGA, the concept of hidden genes is used in GA optimization to hide some of the genes in a chromosome (solution); these hidden genes represent variables that do not appear in this candidate solution. In a topology optimization problem, the number of design variables depends on the specific values of some of the system's variables. Selecting different values for the system's variables changes the length of the chromosome. Different solutions have different number of design variables. So, in the design space, we have chromosomes of different lengths. Let L_{max} be the length of the longest possible chromosome. In the hidden genes concept, all chromosomes in the population are allocated a fixed length equal to L_{max}. In a general solution (a point in the design space), some of the variables (part of the chromosome) will be ineffective in objective function evaluations; the genes describing these variables are referred to as hidden genes. The hidden genes, however, will take part in the genetic operations in generating future generations. To illustrate this concept, consider Figure 3. Suppose we have two chromosomes, the first chromosome is represented by five genes (represented by five binary bits in this example) and the second chromosome is represented by three genes. Suppose also that the maximum possible length

for a chromosome in the population is fixed at 7. Hence, we can say that the first chromosome is augmented by two hidden genes and the second chromosome is augmented by four hidden genes. The hidden genes are not used to evaluate the the fitness of the chromosomes. Because all chromosomes have the same length, standard definitions of genetic algorithms operations can still be applied. Mutation may alter the value of a hidden gene. A crossover operation may swap parts of the chromosome that have hidden genes. A hidden gene in a parent may become an effective gene in the offspring. These hidden genes that become effective take part in the objective function evaluations in the new generations. Figure 4 shows a simple example for two parents with hidden genes and the resulting children after a crossover operation. In figure 4, genes are binary numbers and hidden genes are shown by grey color. The crossover point is between gene 2 and gene 3. As can be seen, the parent chromosomes swap the genes from crossover point and as a result the gene values change in both of the children chromosomes, and the number of hidden genes and/or the location of the hidden gene change. Assigning which gene in the children that need to be hidden is crucial for the efficiency of HGGA; assignment mechanisms are developed in this paper.

3 Hidden Genes Assignment Methods

This paper addresses the question of which genes are selected to be hidden in the HGGA, in each chromosome, in each generation, during the search for the optimal solution (optimal configuration). This mechanism of assigning hidden genes in a chromosome is vital for the efficient performance of HGGA. In previous work of HGGA [21], the mechanism that was used to assign the hidden genes (called "feasibility mechanism") was primitive. The feasibility mechanism rule assumes initially no hidden genes in a chromosome; if the obtained chromosome is feasible then there is no hidden genes. If the solution is not feasible, then starting from one end of the chromosome the algorithm hides genes - one by one - until the chromosome becomes a feasible chromosome.

In genetics, as discussed in Section 2.1, a cell makes a protein that marks the genes to be read. Inspired by genetics, it is proposed to use a tag for each of the genes that have the potential to be hidden (configuration gene) [24]. This tag determines whether that gene is hidden or not. The tag is implemented as a binary digit that can take a value of '1' or '0', as shown in Figure 5. For each gene x_i that can be hidden, a tag_i is assigned to decide whether it is hidden or not. If tag_i is 1, then x_i is hidden, and if it is 0, x_i is active.

The values of these tags evolve dynamically as chromosomes change during the optimization process. Preliminary work in [24] suggests mechanisms for tags evolution. This paper presents two different concepts for tags evolution. Both concepts are shown below with different variations on each concept.

3.1 Logical evolution of tags

During the chromosome crossover operation, the tags for the children are computed from the tags of the parents, using the logical OR operation. Using a logical operation is not new to GA. Reference [27] used them in the chromosomes crossover operation in GAs. Reference [28] presents a crossover operation where the similar genes in the parents are copied to the two children while the remaining genes in each child are randomly chosen from the two parents. Here, however, the logical operator is applied only on the tags. The crossover of two parents results in two children. Three logics are studied in this paper for the logical evolution of tags:

Logic A: For one child, a gene is *hidden* if the same gene is hidden in any of the parents (Hidden-OR). For the second child, a gene is *active* if the same gene is active in any of the parents (Active-OR). It is possible to think of the logic of the second child as the AND logical operation when used with the hidden state – that is in the second child a gene is *hidden* if the same gene is hidden in both of the parents (Hidden-AND). The resulting children from crossover operation is shown in Figure 6 along with the resulting new tags for the children.

Logic B: The Hidden-OR logic is used for both children. Even though the tags will be the same in both children, the two children represent two different solutions because they have different gene values.

Logic C: The Active-OR logic is used for both children.

3.2 Stochastic evolution of tags

In this concept, the tags are evolved using crossover and/or mutation operations, in a similar way to that of the design variables. Eight mechanisms are investigated using this concept. These mechanisms are:

Mechanism A: tags evolve through a mutation operation with a certain mutation probability. In this mechanism, the tags are separate from the design variables in the chromosome.

Mechanism B: tags evolve through mutation and crossover operations. In this mechanism, the tags are considered as discrete variables similar to the design variables in the chromosome. The tags are appended to the design variables; and hence their values are optimized along with the other variables through the selection, crossover, and mutation operations. In this mechanism the number of design variables is increased. The computational cost of evaluating the cost function is not changed though.

Mechanism C: tags evolve through a crossover operation. In this mechanism, the tags are considered as discrete variables similar to the design variables in the chromosome; yet only crossover operation can be applied to the tags.

Mechanism D: tags evolve through a mutation operation. In this mechanism, the tags are considered as discrete variables similar to the design variables in the chromosome; yet only mutation operation is applied to the tags.

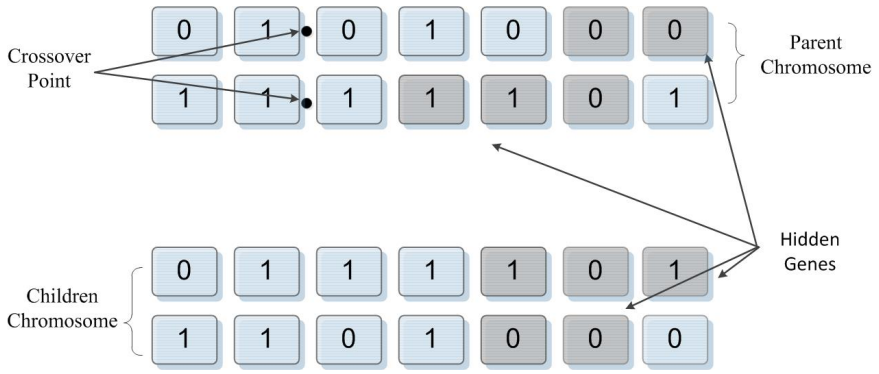

Fig. 4: Crossover operation in HGGA

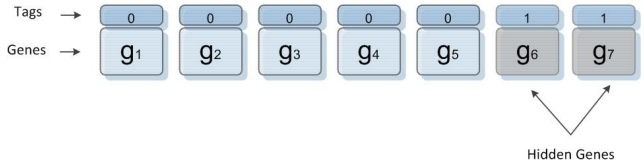

Fig. 5: HGGA and the tags concept

The mutation probability in this mechanism is the same as that used to mutate the main chromosomes.

Mechanism E: tags crossover independently from the genes. In other words, the tags may swap while the genes do not, or vise versa. This mechanism can be interpreted as a $2-D$ multiple crossover operator, one direction through tags and one direction through genes. Before applying the crossover operator, tags undergo a mutation operation.

Mechanism F: tags crossover using a logical fitness guided (arithmetic) crossover in which two intermediate chromosomes are produced. In these intermediate chromosomes, the genes are produced from a single crossover operator on parents and the tags are the outcome of the Active-OR logic on the parents' tags. In other words, parent X will have intermediate offspring Xx, and parent Y will have intermediate offspring Yy using the arithmetic crossover operation. The actual offspring is then created by a fitness guided crossover operation on the parents, and it is closer to the parent whose intermediate offspring has better cost.

Mechanism G: the arithmetic crossover is used with a modified cost function based on the number of genes that are hidden. The offspring is biased toward better parent (lower cost) with more hidden genes. The cost function is modified as follows: $f_{modified}(X) = f(X) - \sum_{i=1}^{M}(flag_i)$, where $flag_i$ is the value of the tag for gene i.

Mechanism H: the arithmetic crossover is used with a modified cost function based on the number of hidden genes. The offspring is biased toward better parent (lower cost) with less hidden genes. The cost function is $f_{modified}(X) = f(X) + \sum_{i=1}^{M}(flag_i)$.

The last four mechanisms (E through H) are presented in detail in reference [24]; hence no results will be presented for them in this paper. However, rankings are presented in Section 6 for all the mechanisms for comparison. Also, the numerical results presented in this paper will be compared to those presented in [24]. Numerical testing for the first four mechanisms (A through D) is presented in Section 4.

The HGGA method presented in this paper is relatively simple to implement. The genes undergo the standard GAs operations while the binary tags undergo different operations depending on the selected mechanism/logic as detailed above. Hence any existing code for GAs can be appended by a code that handles the tags, to create the proposed VSDS GAs.

4 Test Cases: VSDS Mathematical Functions

Multi-minima mathematical functions can be very useful in testing new optimization algorithms. For example, some mathematical functions can be used to tune the tags mutation probability for the different mechanisms presented in this paper, before they are used to solve more computationally intense problems. Three benchmark mathematical optimization problems were modified to make them VSDS functions, so that they can be used to test the HGGA. These functions are: the Egg Holder, the Schwefel 2.26, and the Styblinski-Tang functions. These functions have special structure; they are designed to test the effectiveness of global optimization algorithms. The general concept of modifying these functions to be VSDS is as follows. Consider the optimization cost function defined as:

$$F(X) = \sum_{i=1}^{N} f(x_i) \tag{2}$$

For each gene x_i, if its associated tag, tag_i, is 1 (hidden) then the gene x_i is hidden and hence $f(x_i)$ is set to zero (does not exist). This is consistent with the physical system test cases presented in Section 5. Unlike the hidden gene tags, the chromosomes evolve through the standard GA selection, mutation and crossover operations. For the chromosomes, a single point crossover and an adaptive feasible mutation operators are selected.

In the test cases presented in this section, the population size is 400, the number of generations is 50, the elite count is

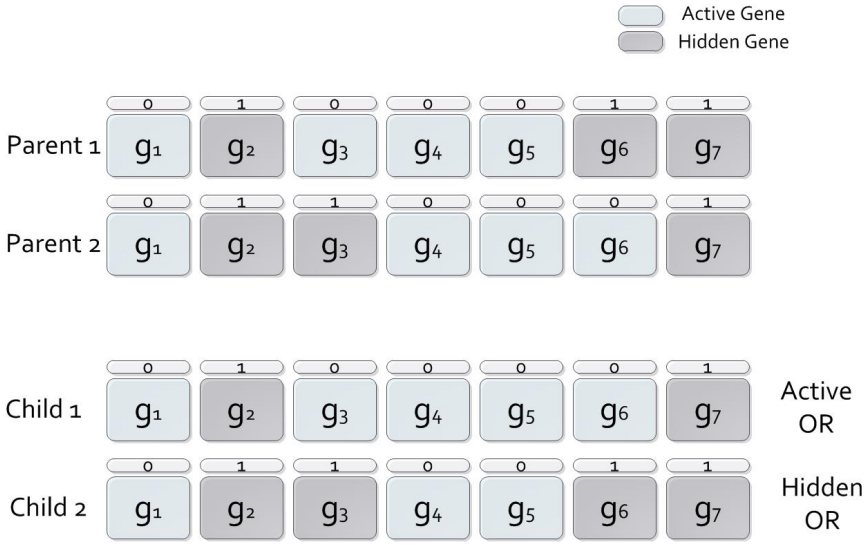

Fig. 6: Tags of children as computed using the logic A

20, the genes mutation probability is 0.01, and the crossover probability is 0.95. For the purpose of statistical analysis, each numerical experiment is repeated n times (n identical runs) and the success rate in finding the optimal solution is assessed. The success rate S_r is computed as $S_r = j_s/n$, where n is the number of runs and j_s is a counter that counts how many times the optimal solution is obtained as the solution at the end of an experiment [29].

4.1 Results Using Stochastically Evolving Tags

4.1.1 Schwefel 2.26 Function

The Schwefel 2.26 function is defined as follows:

$$f(X) = \sum_{i=1}^{N} f_i(x_i) = \sum_{i=1}^{N} -\frac{1}{N} x_i sin(\sqrt{x_i}). \quad (3)$$

subject to $-500 \leq x_i \leq 500$.

Here it is assumed that $N = 5$ (maximum possible number of variables is 5). Minimizing $f(X)$, the optimal solution is known, and it is $f_{min} = -418.9829$. First, the standard GA is used to solve this problem. In using the standard GA, all variables are active. The results show that a minimum of -418.8912161 is obtained using the standard GA. For the HGGA, there are N tags in this case. The results of HGGA Mechanism A for different mutation probability values are presented in Table 1.

As shown in Table 1, the minimum obtained function value is -418.8967549 which is lower than the solution obtained using the standard GA. The occurrence probability (success rate) for Mechanism A is 100%. For Mechanism B, the minimum obtained function value is -418.4088183 with occurrence probability of 100%. For Mechanism D, the results show that the minimum obtained function value is -418.1721324 with occurrence probability of 90%. For Mechanism C, the minimum obtained function value is -418.2119 with occurrence probability of 100%. The results of these four HGGA mechanisms on the Schwefel 2.26 function are summarized in Figure 7, where the percentages on the figure indicate the success rate of the mechanism.

Table 1: Mechanism A: Tags Separated and Mutated

Mutation Probability of Tags	Function Value	Occurrence Probability
0.00001	-418.8803137	100
0.0001	-418.889191	100
0.001	**-418.8967549**	100
0.005	-418.1629931	80
0.01	-418.2677072	80
0.02	-416.8354092	20
0.03	-418.8471854	90
0.04	-417.941699	90
0.1	-418.7225599	50

4.1.2 Egg Holder Function

The Egg Holder function is defined as follows:

$$\begin{aligned} f(X) &= \sum_{i=1}^{N-1} f_i(x_i, x_{i+1}) \\ &= \sum_{i=1}^{N-1} -(x_{i+1}+47) sin(\sqrt{x_{i+1}+0.5x_i+47}) \\ &\quad -x_i sin(\sqrt{x_i - x_{i+1} - 47}) \end{aligned} \quad (4)$$

where $-512 \leq x_i \leq 512$.

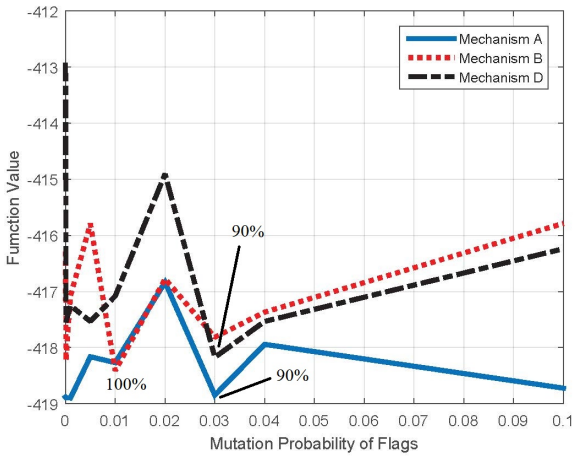

Fig. 7: Comparison of Three HGGA tags mechanisms - using Schwefel 2.26 Function

Table 2: Mechanism A: Tags Separated and Only Mutated

Mutation Probability of Tags	Cost Value	Occurrence Probability
0.00001	−3615.942419	40
0.0001	−3216.79243	100
0.001	−3390.917369	10
0.005	−3655.26687	20
0.01	−3499.743532	10
0.02	−3424.28003	10
0.03	**-3692.314081**	60
0.04	−3640.861368	20
0.1	−3587.802092	10

This is an interesting case study for the HGGA. Each function $f_i(x_i, x_{i+1})$ is a function of two variables. Hence if a variable x_i is hidden, this does not necessarily mean the function f_i goes to zero. There are few ways of handling this type of functions depending on what these functions represent in a physical system. For example, in the interplanetary trajectory optimization problem (will be discussed in detail later in this paper), an event of planetary fly-by is associated with few variables, of them are the fly-by height and the fly-by plane orientation angles. If one of these variables is hidden then that implies the whole fly-by event is hidden and hence the other variables in this event are also hidden. This example suggests that a function $f_i(x_i, x_{i+1})$ (could be representing the cost of a fly-by) would assume a value of zero if any of the variables x_i or x_{i+1} is hidden. In other situations, however, this is not the case. A function $f_i(x_i, x_{i+1})$ may have a non-zero value despite one of the variables x_i or x_{i+1} being hidden. Here in this mathematical function, a tag tag_i is assigned to each function f_i and hence the tag_i value determines whether the value of the function f_i is zero or not.

First, the standard GA is used to optimize this function assuming $N = 5$. The best solution found by the standard GA has a function value of -3657.862773. For the HGGA, there are $N-1$ tags for the $N-1$ functions f_i. Different mutation probability values are tested for Mechanism A and the results are presented in Table 2 for $N = 5$.

As shown in Table 2, the mutation probability of 0.03 has the lowest cost function of -3692.314081 with occurrence probability of 60%. This solution is better than the solution obtained using the standard GA.

A similar analysis is conducted for Mechanism B. The results show that the lowest obtained function value is -3599.845154 with mutation probability of 0.0001 and occurrence probability of 10%. For Mechanism D, the crossover and mutation operations are applied to the N design variables (x_i) and only mutation is applied for the rest $N-1$ variables (the tags). The results show that the lowest obtained function value is -3484.255119 with mutation probability of 0.01 and occurrence probability of 10%. For Mechanism C, the tags are variables (total number of GA variables $2N-1$) and crossover and mutation operations are carried out on the N design variables. Only the crossover operation is applied on the next $N-1$ variables (the tags). The minimum obtained function value is -3559.5751 with occurrence probability of 10%. Mechanism A produced the best solution in this test for the Egg Holder function. The results of these tests are summarized in Figure 8, where the percentages on the figure indicate the success rate of the mechanism.

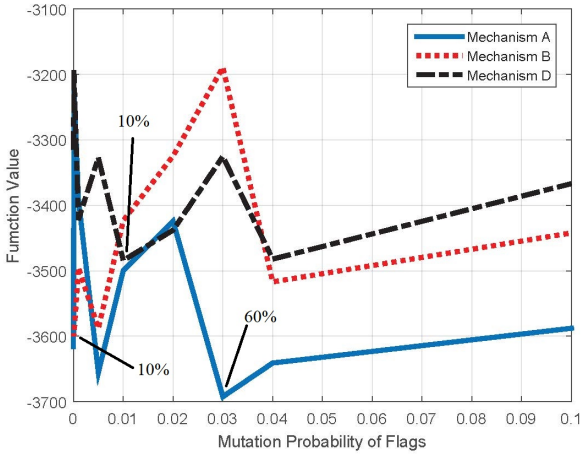

Fig. 8: Comparison of Three HGGA tags mechanisms - using Egg Holder Function

4.1.3 Styblinski-Tang Function

The Styblinski-Tang function is defined as:

$$f(X) = \sum_{i=1}^{N-1} f_i(x_i, x_{i+1}) = \sum_{i=1}^{N} \frac{1}{2}\left(x_i - 16x_i^2 + 5x_i\right). \quad (5)$$

subject to $-10 \leq x_i \leq 10$. It is assumed that $N = 5$. Standard GA results in a minimum obtained function value of -195.8304861. For the HGGA Mechanism A, the obtained function value is -195.8306992, with a mutation probability of 0.03, with occurrence probability of 100%.

For Mechanism B, the results show that the minimum obtained function value is -195.828109 with mutation probability of 0.01 and occurrence probability of 100%. For Mechanism D, the results show that the minimum obtained function value is -195.8261996 with mutation probability of 0.005 and occurrence probability of 100%. For Mechanism C, the minimum obtained function value is -195.82982 with occurrence probability of 100%.

4.2 Results Using Logically Evolving Tags

The logical mechanisms for tags assignment presented in Section 3.1 are tested on the three mathematical functions described above. The results are summarized in Table 3 that lists the obtained solution using each Logic, for each of the functions.

Table 3: Solutions obtained using logical evolution methods of HGGA

Function	Logic A	Logic B	Logic C
Egg Holder	-3386.2499	-3192.6139	-3634.1438
Schwefel 2.26	-418.9635	-415.8967	-418.9632
Styblinski-Tang	-195.8280	-195.8293	-195.8242

Comparing the results obtained in this section to the results obtained using the stochastic assignment mechanisms, it can be concluded that for the Egg Holder function, Logic C works best among all the methods while Mechanism A works best for the Schwefel 2.26 and the Styblinski-Tang functions.

5 Test Cases: Interplanetary Trajectory Optimization

In general, interplanetary trajectory optimization is usually a challenging problem, and in some cases it is nearly impossible to find the optimal solution. Several examples of such complex problems can be found in The Global Trajectory Optimization Competition online portal of the European Space Agency [30]. A typical problem statement can be written as follows: For a given range of the departure date from the home planet Earth, a given range of the arrival date to a target planet, and a given dry mass of the spacecraft, find the mission architecture (that is: how many fly-bys in the mission, and how many DSMs in each leg), as well as the dates and times of fly-bys, the fly-bys planets, the dates and times of DSMs, and the amounts and directions of these DSMs, and the exact launch and arrival dates, such that a given objective is optimized (e.g. the fuel mass needed for the whole mission is minimized). This VSDS optimization problem is used in this paper to test the ability of the HGGA, with the new definitions of hidden genes tags and evolution mechanisms, in autonomously searching for the optimal architecture as well as the non-architecture variables. This type of problems is characterized by the high computational cost of evaluating the cost function; and hence the high computational cost of the optimization process in general. Both Logic A and Mechanism A for evolution of hidden genes tags are tested. For each interplanetary trajectory optimization problem, the numerical experiment is repeated identically 100 times and the success rate is calculated.

5.1 Earth - Jupiter Mission Using HGGA

The Earth-Jupiter mission is optimized using the HGGA and the results are compared to the best known solution in the literature. The design variables and their given upper and lower bounds are listed in Table 4. In this test, it is assumed that the maximum number of fly-bys is 2. Hence the chromosome has 2 genes for the fly-by planets; each gene carries the planet identification number (the planet identification number ranges from 1 to 8 for all the planets in the solar system as shown in Table 4). Each of these two genes has a tag. If both tags have a value of 1, then the two genes are hidden and that solution has no fly-bys. If one gens is hidden and one is active, then the solution has one fly-by and the fly-by planet identification number is the value of that gene. The same concept is applied regarding the DSMs. In this test case it is assumed that the maximum number of DSMs in each leg is 2. Since the maximum number of fly-bys is 2 then the maximum number of legs is 3, and hence the maximum number of DSMs is 6. For each DSM, we need to compute the optimal time (TD) at which this DSM occurs. A gene and a tag are added for each DSM time TD, and hence there are 6 genes and 6 tags for TD_i ($i = 1 \cdots 6$) in this mission. Note that if a fly-by is hidden, then its leg disappears and the DSMs in that leg automatically become hidden. Note also that even if a fly-by exists, a DSM in its leg can be hidden depending on the value of its own tag. Each DSM is an impulse represented by a vector of three components (it has the units of velocity). So, the chromosome will have genes for $6 \times 3 = 18$ scalar components of the DSMs. Note that these 18 genes are grouped in groups of three genes since each three are the components of one DSM vector; hence if one DSM is hidden then its 3 genes get hidden together. Table 4 shows the ranges for 6 DSMs, each has 3 components. The Time Of Flight (TOF) in each leg is also a variable; there is a gene for each TOF in the mission. Hence in this Jupiter mission, we have 3 genes for the TOFs. Note that there are no tags associated with the TOF genes since the state of each gene (hidden or active) is determined based on the fly-by tags. If

a fly-by exists then there is an active gene for a TOF associated with it. There is at least one TOF in a mission; this case of only one TOF corresponds to the case of no fly-bys. To complete a fly-by maneuver, we need to calculate the optimal values for the normalized altitude h of the spacecraft above the planet as well as the plane angle η of the maneuver. Hence, two genes for the altitudes and two genes for the angles are added. Similar to the TOF variables, no tags are needed for the h and η genes. There are also 6 genes for the departure impulse, flight direction, the arrival date and the departure date.

Table 4: Lower and upper bounds of Earth-Jupiter problem

Design Variable	Lower Bound	Upper Bound
planet of Fly-by #1	$1\,(Mercury)$	$8\,(Neptune)$
planet of Fly-by #2	1	8
Epoch of DSM (TD_i), $i = 1\cdots6$	0.1	0.9
DSM_i vector (km/s), $i = 1\cdots6$	$[-5,-5,-5]$	$[5,5,5]$
TOF_i (days), $i = 1\cdots3$	80	800
Flyby altitudes h_i, $i = 1\cdots3$	0.1	10
Flyby angle (rad) η_i, $i = 1\cdots3$	0	2π
Departure Impulse (km/s)	$[-5,-5,-5]$	$[5,5,5]$
Flight Direction	Posigrade	Retrograde
Departure Date	01 Sep.2016	30 Sep.2016
Arrival Date	01 Sep.2021	31 Dec.2021

Due to the high computational cost of evaluating the fitness of a solution and hence the high computational cost of searching for the optimal solution, the problem is usually solved in two steps [23]. The first step is to assume no DSMs in the trajectory, and this step is called zero-DSM. In the second step, the fly-by planets sequence is obtained from the first step, and held fixed, to search for the optimal values of the remaining variables; this step is called the Multi Gravity Assist with Deep Space Maneuvers (MGADSM). Each of the two steps has an element of the architecture to be optimized: the first step optimizes the sequence of fly-bys and the second step optimizes the number of DSMs in each leg. This two-step approach is detailed in reference [23], and has shown to be computationally efficient. The number of generations and population size are selected to be 100 and 300, respectively.

5.2 Earth - Jupiter Mission: Numerical Results and Comparisons

All mechanisms evolving the tags over subsequent generations were tested on the Earth-Jupiter problem. Mechanism A (with a mutation probability of 5%) generated the best solution and so its solution is presented here in detail. The solution of the first phase shows that the trajectory consists of 2 fly-bys around planets Venus and then Earth; the mission sequence is then Earth-Venus-Earth-Jupiter (EVEJ). The second phase results in adding one DSM, and the total mission cost is 10.1266 km/sec (the fuel consumption can be measured in velocity units). The resulting mission is detailed in Table 5.

Also both Logic C and Logic A were tested on this problem; logic A demonstrated superiority compared to logic C in this problem and hence the only the results of logic A are here presented. The total cost for the mission is 10.1181 km/sec. The detailed results of both steps are presented in Table 6.

Comparing the results of Logic A with Mechanism A, we can see that the cost of the mission using Logic A is slightly better than that obtained using Mechanism A. The mission architecture is the same from both methods, while the values of the other variables are slightly different. The success rate of an algorithm is a measure for how many times the algorithm finds the best found solution in a repeated experiment. This experiment was repeated 200 times using Logic A and the success rate is 75.5%, as shown in Figure 14.

Previous solutions in the literature for this problem can be divided into two categories. The first category of methods do not search for the optimal architecture; rather the trajectory is optimized for a given architecture. Reference [31], for instance, presents a minimum cost solution trajectory for this Earth-Jupiter mission, assuming a fixed planet sequence of EVEJ. The departure, arrival, and fly-bys dates were also assumed fixed, with a launch in 2016 and a mission duration of 1862 days. The primer vector theorem solution has four DSMs. Two DSMs are applied in the first two legs. The total cost for this solution is 10.267 km/s, which is about slightly higher than the obtained cost in this paper. The method presented in this paper, however, has the advantage of the autonomous search for the optimal architecture of the solution. The obtained solution in this paper has the same planet sequence of EVEJ but a different DSM architecture compared to [31]. Reference [23] presents the solution to this problem using the HGGA but without the tags concept. Reference [23] implements a simple feasibility check in assigning the hidden genes in each chromosome. The solution in [23] also finds the planet sequence of EVEJ, and has a total mission cost of 10.182 km/sec, which is slightly higher than the cost obtained in this paper. This problem was also solved using Mechanisms E and F (presented in Section 3) and the results were presented in [24]. The total cost obtained using Mechanism E is 10.1438 km/s and using Mechanism F is 10.9822 km/s, which are higher than the cost obtained in this paper. The mission trajectory obtained using Mechanism A is shown in Figure 9.

As a demonstration for how the tags evolve over subsequent generations, consider this Earth-Jupiter problem solved using Logic C. The population size is 300 and the number of generations is 100. Six tags are examined. Figure 10 shows the number of times each tag has a value of '1' in each generation. For example, tag 6 takes a value of '1'

Table 5: HGGA solution of Earth-Jupiter problem using Mechanism A

Mission parameter	Zero-DSM model (first step)	MGADSM model (second step)
Departure Date	$11-Sep-2016, 04:50:27$	$31-Aug-2016, 02:15:15$
Departure Impulse (km/s)	3.6283	3.4414
DSM_1 date	–	$26-Oct-2016, 21:29:29$
DSM_1 impulse (km/s)	–	0.036516
Venus flyby date	$07-Sep-2017, 01:58:51$	$05-Sep-2017, 09:52:46$
Post-flyby impulse (km/s)	0.031108	0.001704
Pericenter altitude (km)	1333.7876	1255.2411
Earth flyby date	$03-Apr-2019, 09:55:30$	$29-Mar-2019, 09:56:34$
Post-flyby impulse (km/s)	0.4685	0.4522
Pericenter altitude (km)	637.7999	637.8000
Arrival date	$25-Dec-2021, 23:05:00$	$25-Dec-2021, 18:23:09$
Arrival impulse (km/s)	6.2813	6.1948
TOF (days)	$360.88083, 934.21184, 636.66743$	$370.3177, 570.0026, 902.3518$
Mission cost (km/s)	10.4092	10.1266

Table 6: HGGA solution of Earth-Jupiter problem using Logic A

Mission parameter	Zero-DSM model (first step)	MGADSM model (second step)
Departure Date	$01-Sep-2016, 22:58:19$	$29-Aug-2016, 16:04:38$
Departure Impulse (km/s)	3.4811	3.1398
DSM date	–	$07-Oct-2016, 05:57:10$
DSM impulse (km/s)	–	0.34746
Venus flyby date	$05-Sep-2017, 10:42:20$	$06-Sep-2017, 19:14:50$
Post-flyby impulse (km/s)	0.006200	$1.9788e-05$
Pericenter altitude (km)	1330.9042	972.1739
Earth flyby date	$29-Mar-2019, 23:48:29$	$29-Mar-2019, 14:31:32$
Post-flyby impulse (km/s)	0.4398	0.4396
Pericenter altitude (km)	637.8000	637.8000
Arrival date	$19-Sep-2021, 03:07:38$	$23-Sep-2021, 15:01:36$
Arrival impulse (km/s)	6.1999	6.1891
TOF (days)	$368.4889, 570.5459, 904.1383$	$373.1321, 568.8033, 909.0209$
Mission cost (km/s)	10.1299	10.1181

in all the population members in generations 55 and above. In the 30^{th} generation, for instance, tag 6 takes a value of '1' in only 40 chromosomes and takes a value of '0' in the other 260 chromosomes. The other 5 tags converge to a value of '0' in the last population in all the chromosomes.

6 Statistical Analysis

A statistical analysis is conducted on the methods presented in this paper. Two different analysis tools are implemented. The first is evaluating the success rate for each method in solving different problems. The second tool ranks the proposed algorithms using the Sign test [32].

For each mathematical function presented in Section 4, the success rate of each logical and stochastic mechanism

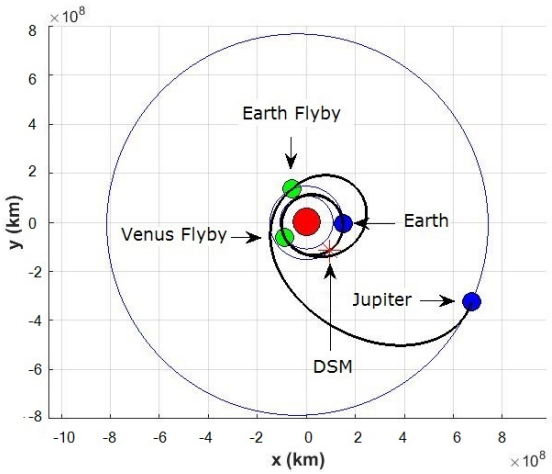

Fig. 9: Mechanism A: EVEJ Trajectory for MGADSM Model

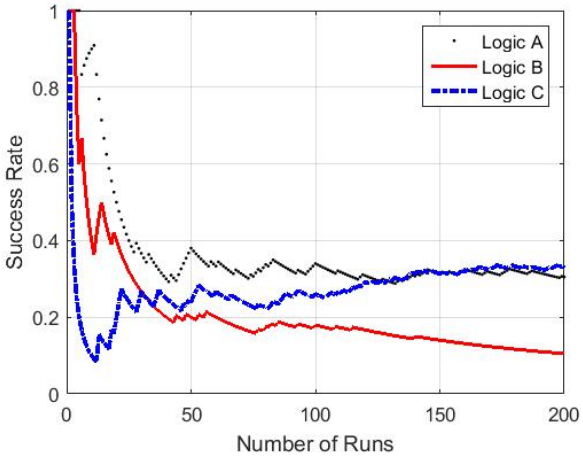

Fig. 11: Success rate versus number of runs for Egg Holder function using three different logics for tags evolution

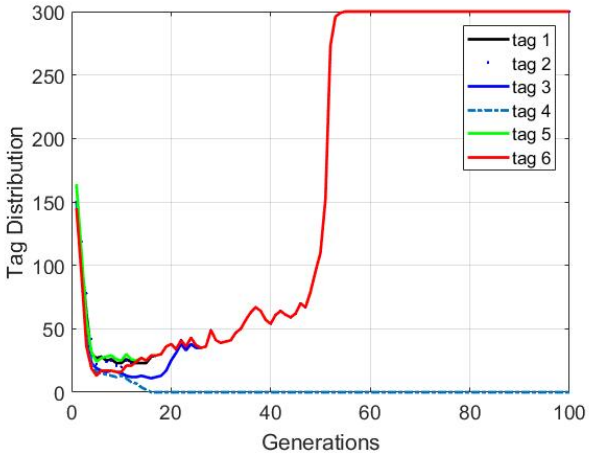

Fig. 10: Evolution of tags using Logic C in the Earth-Jupiter problem

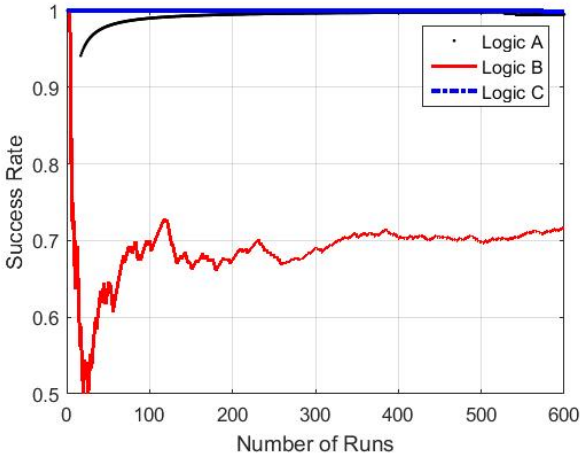

Fig. 12: Success rate versus number of runs for Schwefel 2.26 function using three different logics for tags evolution

is calculated numerically. By repeating the same numerical experiment, the obtained solution in each experiment is compared to the best obtained solution and a success rate can be updated as the experiment being repeated. For each of the three logical mechanisms presented in this paper, the success rates in finding the best solution for the Schwefel 2.26 function is shown in Figure 12. Logic B has a success rate of about 70% which is less than that of logics A and C, which is about 100%. For each of the three logical mechanisms presented in this paper, the success rates in finding the best solution for the Egg Holder function is shown in Figure 11. Logic B has a success rate less than that of logics A and C. Both logics A and C settle at a success rate of about 30%. In all three mathematical functions, Logic A and Logic C have very close success rates and Logic B has a lower value for the success rate.

For the stochastic mechanisms presented in this paper, the success rate of the Schwefel 2.26 function is shown in Figure 13. In Figure 13, the mutation rate of Mechanism A is 0.001, for Mechanism B is 0.01, and for Mechanism D is 0.03. As shown in Figure 13, Mechanisms A, B, and C have higher success rates compared to Mechanism D. Note that Mechanism D has also resulted in higher function values.

The success rate was also computed for the Earth-Jupiter trajectory optimization problem, using both Mechanism A and Logic A. The results are plotted in Figure 14. The success rate for both approaches is about 75%.

The Sign test is a pairwise comparison between different algorithms. It has been applied on the algorithms presented in this paper. Each algorithm is ranked based on the total number of cases in which the algorithm produces the best function value. Table 7 summarizes the results obtained in Section 4, where the best function value obtained using each algorithm is listed. Table 7 also lists the results obtained using Mechanisms E, F, G, and H which are detailed in reference [24]; they are used here for the purpose of ranking. Listed also in Table 7 is the solutions obtained using the No-tag HGGA which is originally developed in reference [21].

Table 7: Best objective function values of all the test cases obtained from all algorithms

	Egg Holder	Schwefel 2.26	Styblinski-Tang
Logic A	−3386.2499	−418.9635	−195.8280
Logic B	−3192.6139	−415.8967	−195.8293
Logic C	−3634.1438	−418.9632	−195.8242
Mechanism A	−3692.314081	−418.8967	−195.8306992
Mechanism B	−3599.845154	−418.4088	−195.828109
Mechanism C	−3559.5751	−418.2119	−195.82982
Mechanism D	−3484.255119	−418.1721	−195.8261996
Mechanism E	−3552.8947	−418.2850	−195.8298
Mechanism F	−3644.2279	−418.0799	−195.8278
Mechanism G	−2571.8028	−375.3514	−191.8393
Mechanism H	−2134.7217	−303.5320	−194.5325
No-tag HGGA	−2749.2646	−335.1844	−156.6646

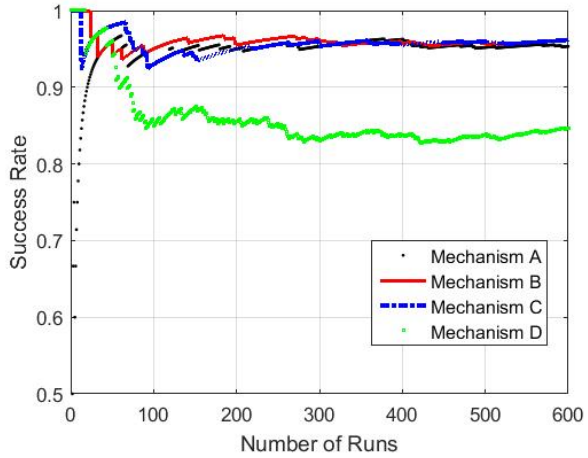

Fig. 13: Success rate of the Schwefel 2.26 function for different stochastic mechanisms

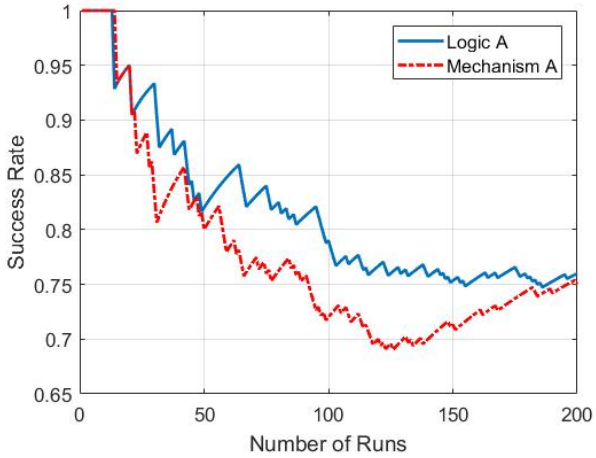

Fig. 14: Success rate versus number of runs for the Earth-Jupiter trajectory optimization problem using Mechanism A and Logic A

Using the data presented in Table 7, the Sign rank is computed by counting how many times each algorithm resulted in the best function value among all functions. Table 8 lists the Sign rank for each algorithm. Mechanism A then Logic A have the highest ranks. It is to be noted here, though, that this metric is best when the number of tested functions is high.

Table 8: Sign test ranks

	Sign rank		Sign rank
No-tag HGGA	0	Mechanism A	2
Mechanism B	0	Mechanism C	0
Mechanism D	0	Mechanism E	0
Mechanism F	0	Mechanism G	0
Mechanism H	0	Logic A	1
Logic B	0	Logic C	0

7 Conclusion

The concept of binary tags is introduced in genetic algorithms to enable hiding some of the genes in a chromosome, so that they can be used to search for optimal architectures in VSDS problems. The proposed binary tags concept mimics biological cells in hiding the genes that are not supposed to be effective in the cell, while they could be effective in other cells. Mechanisms for assigning the chromosome hidden genes are proposed and investigated in this paper. Two categories of mechanisms for evolving the values of these tags in subsequent generations are proposed in this paper. The first one uses logical operations to evolve the tags while the other one uses stochastic operations for tags evolution. Numerical tests were conducted on mathematical optimization problems as well as the interplanetary trajectory optimization for a spacecraft mission from Earth to planet Jupiter. The implementation of the new hidden genes assignment mechanisms to the space trajectory optimization problem and the mathematical optimization problems demonstrated its capability in searching for the optimal architecture, in addition to improving the solution compared to the original hidden genes genetic algorithm approach that does not implement the tags concept. It is demonstrated in this paper that, for the trajectory optimization problem, it is possible to autonomously compute the optimal number of fly-bys, the planets to fly-by, and the optimal number of deep space maneuvers, in addition to the rest of the design variables using the proposed algorithms. Statistical analysis conducted in this paper on the mathematical optimization problems showed that, in terms of optimality of the solution, Mechanism A and Logic A performed better than the other algorithms.

Acknowledgments

This work was funded by National Science Foundation, award no. 1446622.

Superior, a high performance computing cluster at Michigan Technological University, was used in obtaining the results presented in this publication.

References

[1] National Science Foundation. Cyber Physical Systems Program. http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503286.

[2] Goldberg, D. E., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[3] Kennedy, J., and Eberhart, R. C., 1995. "Particle swarm optimization". In Proceeding of the 1995 IEEE International Conference on Neural Networks, IEEE Service Center.

[4] Dorigo, M., Maniezzo, V., and Colorni, A., 1996. "The ant system: Optimization by a colony of cooperating agents". *IEEE Transactions on Systems, Man, and Cybernetics-Part B,* **26**, pp. 29–41.

[5] Price, K., Storn, R. M., and Lampinen, J. A., 2005. *Differential evolution: a practical approach to global optimization*. Natural Computing Series. Springer.

[6] O'Reilly, U.-M., and Oppacher, F., 1994. *Program search with a hierarchical variable length representation: Genetic Programming, simulated annealing and hill climbing*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 397–406.

[7] Harvey, I., 1992. "Toward a practice of autonomous systems". *Proceedings of First European Conference on Artifcial Life*.

[8] Harvey, I., 1992. "The saga cross: The mechanics of recombination for species with variable length genotypes". *Parallel Problem Solving from Nature 2*.

[9] Lee, C.-Y., and Antonsson, E., 2000. "Variable length genomes for evolutionary algorithms". *In Proceedings of the Genetic and Evolutionary Computation Conference*.

[10] Zebulum, R. S., Vellasco, M., and Pacheco, M. A., 2000. "Variable length representation in evolutionary electronics". *Evolutionary Computation,* **8**(1), pp. 93–120.

[11] Ryoo, J., and Hajela, P., 2004. "Handling variable string lengths in GA-based structural topology optimization". *Struct Multidisc Optim,* **26**.

[12] Kim, I. Y., and de Weck, O., 2004. "Variable chromosome length genetic algorithm for structural topology design optimization". *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*.

[13] Kim, I. Y., and de Weck, O., 2005. "Variable chromosome length genetic algorithm for progressive refinement in topology optimization". *Struct Multidisc Optim,* **29**.

[14] Abdelkhalik, O., and Gad, A., 2012. "Dynamic-size multiple populations genetic algorithm for multigravity-assist trajectories optimization". *AIAA Journal of Guidance, Control, and Dynamics,* **35**(2), March–April, pp. 520–529.

[15] Nyew, H. M., Abdelkhalik, O., and Onder, N., 2012. "Structured chromosome evolutionary algorithms for multi-gravity-assist trajectories optimization". In AAS / AIAA Astrodynamics Specialist Conference, no. AIAA 2012–4519.

[16] Nyew, H. M., Abdelkhalik, O., and Onder, N.,

2015. "Structured-chromosome evolutionary algorithms for variable-size autonomous interplanetary trajectory planning optimization". *Journal of Aerospace Information Systems*, pp. 1–15.

[17] Allison, J. T., Khetan, A., and Lohan, D., 2013. "Managing variable-dimension structural optimization problems using generative algorithms". In 10^{th} World Congress on Structural and Multidisciplinary Optimization.

[18] Guo, X., Zhang, W., and Zhong, W., 2014. "Topology optimization based on moving deformable components: A new computational framework". *Computing Research Repository,* **abs/1404.4820**.

[19] Yang, L., 2011. "Topology optimization of nanophotonic devices". Phd, Technical University of Denmark, Department of Photonics Engineering, Building 343, DK-2800 Kongens Lyngby, Denmark.

[20] Suzuki, K., and Kikuchi, N., 1991. "A homogenization method for shape and topology optimization". *Computer Methods in Applied Mechanics and Engineering,* **93**, pp. 291–318.

[21] Abdelkhalik, O., 2013. "Hidden genes genetic optimization for variable-size design space problems". *Journal of Optimization Theory and Applications,* **156**(2), pp. 450–468.

[22] Abdelkhalik, O., 2011. "Multi-gravity-assist trajectories optimization: Comparison between the hidden genes and the dynamic-size multiple populations genetic algorithms". In AAS / AIAA Astrodynamics Specialist Conference, no. AAS11-620.

[23] Gad, A., and Abdelkhalik, O., 2011. "Hidden genes genetic algorithm for multi-gravity-assist trajectories optimization". *AIAA Journal of Spacecraft and Rockets,* **48**(4), July–August, pp. 629–641.

[24] Abdelkhalik, O., and Darani, S., 2016. "Hidden genes genetic algorithms for systems architecture optimization". In Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, ACM, pp. 629–636.

[25] Starr, B. "Spooled dna and hidden genes: The latest finding in how our dna is organized and read". *The Tech Museum of Innovation, Department of Genetics, Stanford School of Medicine, 201 South Market Street San Jose, CA 95113*. `http://www.thetech.org/genetics/news.php?id=31`.

[26] The National Institute of General Medical Sciences, 2010. The new genetics. On line website, April. National Institutes of Health.

[27] Park, J.-M., Park, J.-G., Lee, C.-H., and Han, M.-S., 1993. "Robust and efficient genetic crossover operator: homologous recombination". In Neural Networks, 1993. IJCNN '93-Nagoya. Proceedings of 1993 International Joint Conference on, Vol. 3, pp. 2975–2978 vol.3.

[28] Ku, S., and Lee, B., 2001. "A set-oriented genetic algorithm and the knapsack problem". In Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, Vol. 1, pp. 650–654.

[29] Vasile, M., Minisci, E., and Locatelli, M., 2010. "Analysis of some global optimization algorithms for space trajectory design". *Journal of Spacecraft and Rockets,* **47**(2), March-April, pp. 334 –344.

[30] European Space Agency, Mission Analysis and Design, Advanced Concepts Team. Global Trajectory Optimisation Competition. `https://sophia.estec.esa.int/gtoc_portal/`.

[31] Olympio, J. T., and Marmorat, J.-P., 2007. Global trajectory optimisation: Can we prune the solution space when considering deep space maneuvers? Final Report Ariadna ID: 06/4101, Contract Number: 2007/06/NL/HI, "European Space Agency", Sep.

[32] Derrac, J., Garcia, S., Molina, D., and Herrera, F., 2011. "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms". *Journal of Swarm and Evolutionary Computation,* **1**.