An Evolutionary Framework for Analyzing the Distance Preserving Property of Weighted Graphs

Emad Zahedi
Department of Mathematics,
Department of Computer
Science and Engineering,
Michigan State University
East Lansing, MI 48824, U.S.A.
Email: Zahediem@msu.edu

Masoud Mirmomeni
Department of Computer
Science and Engineering,
Michigan State University
East Lansing, MI 48824, U.S.A.
Email: Mirmomen@msu.edu

Abdol-Hossein Esfahanian
Department of Computer
Science and Engineering,
Michigan State University
East Lansing, MI 48824, U.S.A.
Email: Esfahanian@cse.msu.edu

***Abstract*—A subgraph H of a given graph G is *isometric* if the distances between every pair of vertices in H are the same as the distances of those vertices in G. We say a graph G is *distance preserving* if there exists an isometric subgraph of every possible order up to the order of G. Distance preserving property has been applied to many real world problems such as route recommendation systems and all kinds of shortest-path-related applications. Here, we propose a biologically-inspired search algorithm to address the problem of finding isometric subgraphs that consequently determines if a given graph is distance preserving. In this algorithm, using a well defined fitness function, selection operator selects almost isometric subgraphs to generate the offspring for the next generation. There is a trade-off between the population size and searching speed. On the one hand, the larger the population size is, the slower the search algorithm would be. On the other hand, by increasing the population size, we increase the likelihood of finding an existing isometric subgraph. Experimental results depict the performance of the proposed algorithm in finding isometric subgraphs even for challenging problems, and interestingly by these results one can see that "almost" all graphs are distance preserving. In closing, we show the smallest distance preserving graph whose product factors are not distance preserving. This graph has 80 vertices, and can be used as benchmark for algorithms in this concept.**

***Index Terms*—Distance preserving, evolutionary algorithm, isometric subgraphs, mutation rate, population size, selection operator.**

I. Introduction

Subgraph classification has been a challenging problem, and many researchers have addressed this problem by applying different methods in large graph database. Subgraphs whose properties are the same as the main graph are important since we can use them as smaller samples to analyze the properties of original large graphs when graph analysis is computationally expensive.

Subgraphs in which distances between every pair of vertices are the same as their distance in the main graph are called *isometric*. From theoretical point of view isometric subgraphs and isometric embeddings of graphs into another graphs widely have been considered, see for example [11], [18]. This concept was considered in an algorithmic point of view as well [23]. Distance property while isometric subgraphs come into play, has applications in network clustering [20], and also in chemical graphs [16].

One family of graphs related to isometric subgraphs is the set of distance-hereditary graphs in the literature. A connected graph G is called *distance-hereditary* if every connected induced subgraph of G is isometric. Distance-hereditary graphs were first proposed by Howorka [12] and have appeared in various papers, see for example [2], [7].

The definition of distance-hereditary graphs is restrictive; thus, researchers have generalized this definition and come up with the sequentially distance-preserving graphs. A graph is *sequentially distance preserving* if its vertices can be ordered such that deleting the first i vertices results in an isometric subgraph, for all $i \geq 1$ [15].

The distance-hereditary and sequentially distance preserving graphs are restrictive in real social networks. This happens because any graph of these families cannot contain an induced cycle of length 5 or more [21]. There is also a more complex notion in the literature that relaxes these properties which is called distance-preserving graphs. A graph is *distance preserving*, for which we use the abbreviation "dp", if it has an isometric subgraph of every possible order [19]. To make this definition clear, an example for a dp and a non-dp graph is given in Figures 1 and 2 respectively. Checking "distance preserving" property of a graph is indeed harder than checking if the graph is "distance hereditary" since distance preserving is an existential quantification and distance hereditary is a universal quantification for isometric subgraphs.

Distance preserving property has been applied to many real world problems such as route recommendation systems, logistics planning, and all kinds of shortest-path-related applications that run on resource-limited mobile devices [23].

Our foremost concern is finding an algorithm to show whether a graph is dp or not, which is equivalent to finding an isometric subgraph of any order of a given graph, if it exists. One way to determine whether a given subgraph is isometric, would be computing all-pairs shortest paths for the subgraph, and for G as well. A Floyd type algorithm, which has time complexity at most $O(n^3)$ [10], can be used to address this problem; however, choosing appropriate subgraph

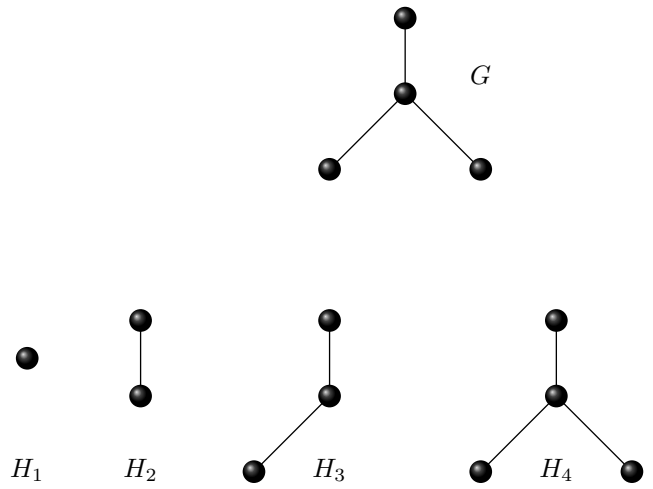

Fig. 1. An example for a distance preserving graph G with isometric subgraphs H_1, H_2, H_3 and H_4. Clearly all trees are distance preserving since removing any leaf cannot change the distances in the main graph, so by continuing removing leaves we obtain isometric subgraphs of order 1 up to the order of the main graph.

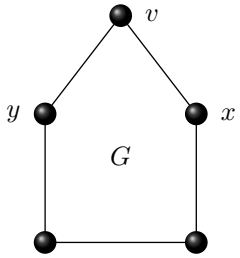

Fig. 2. An example for a non-distance-preserving graph G, which does not have any isometric subgraph of order 4. This happens because removing any vertex like v increases the distances between it's neighbors x and y from 2 to 3 in the main graph.

to see whether it is isometric is a complicated and time consuming process since dp is an existential property.

The question is which subgraph should be chosen and considered as an isometric subgraph to see whether the graph is dp or not? We conjecture this problem cannot be solved in polynomial time complexity, and it is computationally impracticable to do in a naive fashion. It is already shown that deciding if a graph is sequentially distance preserving is NP-complete [6].

On the other hand, biologically inspired search techniques such as genetic algorithm [8], evolutionary programming [1], swarm intelligence methods, and ant colony optimization algorithms [9] have shown great performance in challenging and nonlinear optimization problems without requiring pre-determined assumptions on the problem [14].

Among these approaches, genetic algorithm by providing a simple binary representation of the graph and its subgraphs propose simple ways which yield to a global optimum answer for a special graph searching problem [22]. By defining an accurate fitness function we can use these biologically inspired algorithms with their mutation and recombination operators to evolve population of subgraphs to find isometric subgraphs. In this research, we introduce a biologically inspired search algorithm that finds isometric subgraphs with different order up to the order of any given graph to see if the original graph is distance preserved. The fitness of candidate subraphs is their distance preserving characteristics. To measure this property, we check the difference between the distance of every pairs of nodes in the subgraph from their original distance. The more similar the distance is, the higher the fitness would be for the subgraph. The fittest subgraphs have higher chance to reproduce. This approach has the potential to find isometric subgraphs for sparse challenging graphs.

The remaining sections of this paper are structured as follows. Section II gives the graph theory background and discusses the biologically inspired search algorithm. In section III the outlines of the search algorithm for isometric subgraphs is given. Section IV is devoted to results and discussions. In section V concluding remarks are presented and finally, in section VI it is shown that the smallest order of a dp Cartesian product of graphs whose factors are not dp is 80. Moreover, the graph for such order is displayed. This graph is sparse and it is challenging to find isometric subgraphs of it for algorithms in this concept. So it can be used as a benchmark.

II. Background

A. Graph theory

We refer the reader to [4] for a general overview of graph theory, which includes any definition and notation not given in this paper. In this paper, every graph is finite, simple and connected. Let G be a graph with the vertex set $V(G)$ and edge set $E(G)$, for ease, we use $|G|$ as the number of the vertices of G and also $H \subseteq G$ as H is a subgraph of G. The vertex set adjacent to $v \in V(G)$ is called the *neighborhood* of the vertex v and is denoted by $\mathcal{N}_G(v)$. Let $H \subseteq G$ then

$$\mathcal{N}_G(H) = \{u \mid u \in \mathcal{N}_G(v) - V(H) \text{ for all } v \in V(H)\}.$$

If it is clear from context, for brevity, we use $\mathcal{N}(v)$ and $\mathcal{N}(H)$ instead of $\mathcal{N}_G(v)$ and $\mathcal{N}_G(H)$ respectively.

In a graph G, a *path* is a sequence of distinct vertices $v_0, \ldots, v_k$ such that $v_i v_{i+1} \in E(G)$ for $i = 0, \ldots, k-1$. The length of the path is k, which is the number of edges. A *cycle* of a graph is a sequence of vertices $v_0, \ldots, v_k$ in which $v_0, \ldots, v_{k-1}$ are distinct and $v_0 = v_k$, and $v_i v_j \in E(G)$ if $|i - j| = 1 \pmod k$. The length of a cycle C is its number of edges. The *girth* of a graph G is the smallest length of a cycle in G and denoted by $\mathrm{girth}(G)$.

The *distance* between vertices u, v in G, $d_G(u, v)$, is the minimal length of a path connecting these vertices. If it is clear from context, For brevity, we will use $d(u, v)$, instead of $d_G(u, v)$. A path P from u to v with length $d(u, v)$ is called u-v *geodesic* path. The maximal possible length of all geodesics in G is called the *diameter* of G and denoted by $\mathrm{diam}(G)$.

An induced subgraph H of a graph G is called an *isometric* subgraph if $d_H(a, b) = d_G(a, b)$, for every pair of vertices $a, b \in V(H)$, denoted by $H \leq G$. An n vertex graph G

is called *sequentially distance preserving (sdp)*, if there is an ordering, $v_1, v_2 \ldots, v_n$, of the vertices of G such that $G - \{v_i\}_{i=1}^{s} \leq G$ for $s = 1, \ldots, n$. A connected graph G is called *distance preserving (dp)* if it has an i-vertex isometric subgraph for every $i = 1, \ldots, |G|$.

The *Cartesian product* of G and H is the graph, denoted $G \square H$, on the vertex set $V(G) \times V(H)$ whose edge set is

$$E(G \square H) = \{(a,x)(b,y) \mid ab \in E(G) \text{ and } x = y, \text{ or } xy \in E(H) \text{ and } a = b\}.$$

The reader can consult the book of Imrich and Klavzar [13], for more details about graph products.

B. Biologically inspired algorithms

Biologically inspired algorithms are part of branch of computer science that generally uses and applies subfields related to the topics of connectionism, social behaviour and emergence to solve computationally challenging problems that usually do no have closed form optimal solution. On the one hand, such algorithms massively depend on the fields of biology, computer science and mathematics since they apply ideas and models that have been inspired by examining living organisms to solve unsolved sophisticated problems (usually NP hard) in computer science. On the other hand, such algorithms can help scientist to have a better understanding of complicated behavior of living organisms by proposing simple computational models (in comparison to complex biological dynamics) and test the validity of proposed hypotheses on fast computational models.

From their earliest days, bio-inspired algorithms were not only applied to calculating missile trajectories and deciphering military codes but also to modeling the brain functionality, imitating complex behavior of social animals, trying to model learning skill in human being, and simulating biological evolution. Bio-inspired algorithms have their ups and down over the years, but since the early 1980s they have all undergone a resurgence in the computation research community because of rapid increase in computation speed. After this re-birth, bio-inspired algorithms were divided into three branches. The first field is artificial neural networks (cutting-edge deep learning methodology is the off-spring of this sub-family), the second field was machine learning, and the third branch nowadays is called "evolutionary computation," of which genetic algorithms are the most prominent member of this category.

Genetic Algorithms (GAs) were first introduced by Prof. John Holland in the late 60s and later developed by Holland and his students and colleagues at the University of Michigan in early 70s. In comparison to other evolutionary computation frameworks and programming, Holland's primary goal was to propose a general purpose algorithms to solve not just a specific problems, but rather by formally studying the process of evolution, natural selection, and adaptation as it occurs in biological populations to develop ways in which the mechanisms of natural adaptation might be imported into computer systems. His 1975 book "Adaptation in Natural and Artificial Systems" demonstrated this algorithm as an abstraction of biological evolution and gave a theoretical framework for natural selection, evolutionary operators, and adaptation under the GA. Holland's GA is a framework for generating better solutions from ancestral populations, which are represented by strings of "bits" as "chromosomes" by applying an evolutionary force similar to "natural selection" together with the geneticsinspired operators of crossover, mutation, and inversion. The selection operator gives higher chance of reproduction to those chromosomes in the ancestral population. Crossover operator combines subparts of two selected chromosomes, roughly mimicking biological recombination between two singlechromosome or "haploid" organisms; mutation operator randomly changes the allele binary values of some locations in the chromosome; and finally inversion reverses the order of a contiguous section of the chromosome, thus rearranging the order in which genes are arrayed. Since then, Holland's GA and other evolutionary computation frameworks have been successfully applied to different NP hard problems and have had superb performance.

In this research, we introduce an evolutionary algorithm that generates population of graphs in which the distance property is preserved. Here, the distance property of each graph G with vertex set $V(G)$ and edge set $E(G)$ is analyzed to decide which graph to be used as a member of anscestral population population to generate distance preserved descendants over the course of evolution. In this algorithm, a mating event is performed as follows: Pick a vertex v of the graph uniformly at random. A neighbor of v is chosen for mating with a fitness bias. Crossover and probabilistic mutation are used to produce a single new individual which may or may not be used to replace the individual on vertex v. The details of how the neighbor is picked for mating and how to decide if the new individual replaces the individual on vertex v are called the local mating rule of the graph based genetic algorithm [5].

III. Methodology

A. Problem Definition

Problem III.1. *Let G be a connected graph. Does G contain an isometric subgraph of order k, for some $k \leq |G|$?*

Answering this question helps us to determine whether G is a dp graph or not by considering all k for $k = 1, 2, \ldots, n$. Note that when G is a tree, every connected induced subgraph is obtained by subsequently removing some leaves; thus, the problem is challenging when the graph contains cycles. In other words when there are two different paths between a pair of vertices in a graph then the graph contains a cycle and removing a node from the shorter path may result in a non-isometric subgraph.

B. Algorithm Outline

In this section, an evolutionary search algorithm for solving distance-preserving graph problem is proposed. The flowchart of the proposed algorithm is demonstrated in Figure 3 in which each block will be illustrated in the following subsections. The population size N is a user defined parameter.

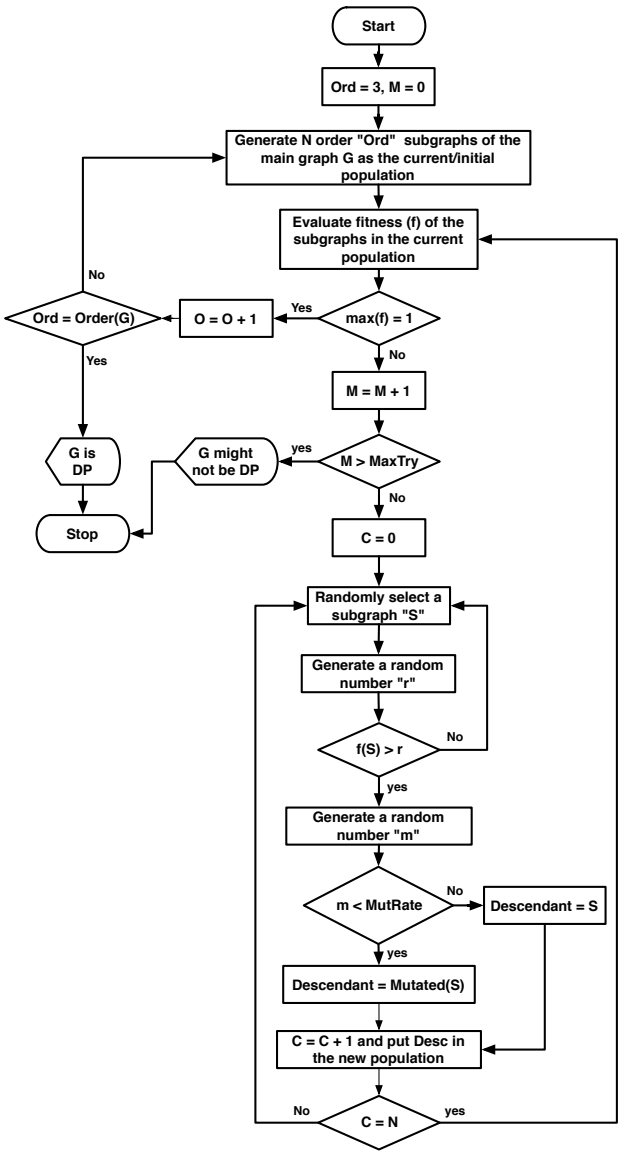

Fig. 3. The proposed genetic-based search algorithm for distance preserving graphs.

1) Generating the initial population: Every connected induced subgraph with diameter 2 is an isometric subgraph; hence the evolutionary algorithm begins it's search for isometric subgraphs of order greater than or equal to three. For the initial population, we can randomly choose N connected subgraphs of all $\binom{|G|}{3}$ induced subgraphs with order three of G. But choosing these N subgraphs for the initial population needs to know all subgraphs of G with order three that is computationally expensive when the order of the graph is large. One way to remedy this issue would be randomly generating N connected subgraphs of G with order three in which, we randomly pick a vertex, a, and then randomly select a neighbor of a, namely b, and then choose a random vertex c adjacent to $\{a, b\}$ in G. We consider the induced subgraph on vertex set $\{a, b, c\}$ as an element of the initial population. In fact, the randomness of this way is the same, but the time complexity is much less.

2) Fitness function: Distance similarity between induced subgraphs and main graph is used to define the fitness function of the evolutionary algorithm. Among all sugraphs of a graph G, the maximally distance preserved subgraphs have the highest fitness, and consequently these subgraphs have higher chances to get selected and reproduce for the next generation. By above aforementioned criterion, we define the fitness function as follows:

$$f(H) = 1 - \left| \frac{\sum_{x,y \in V(H)} d_H(x,y) - \sum_{x,y \in V(H)} d_G(x,y)}{\binom{H}{2} \times \mathrm{diam}(H)} \right|,$$

where the first summand explains the summation of all distances in H and the second one is the summation of all distances between all pair of vertices of H in the original graph G. Clearly if H is a subgraph of G then

$$d_H(x,y) \geq d_G(x,y) \geq 0$$

for every pair of vertices $x, y \in V(H)$. Therefore, if H is isometric in G then $d_H(x,y) = d_G(x,y)$ that gives:

$$f(H) = 1.$$

In general, a subgraph H maximally preserves the distances in G if the phrase inside the absolute value of $f(H)$ is minimal. Having said that, two summations are close to each other and $f(H)$ is closed to 1. Moreover, the denominator of the fraction normalized inside of the absolute value makes the fraction to be a number between zero and one. Note that almost all graphs have diameter 2 [17]. And also using results in [3], one can find a diameter in random graphs according to the chosen model of randomness. Having a fix number for diameter reduces the time complexity of the algorithm. Generally speaking, by proposing this fitness function, we cover all real world networks.

3) Selection operator : Pick an arbitrary graph H from the population and an arbitrary real number, r, within the interval $[0, 1]$. If $f(H) \geq r$ then H is selected for the next generation. By using this operator, those subgraphs whose fitness values are high have more chance to get selected unlike those subgraphs whose fitness values are low.

4) Mutation: Suppose an element of the population, more precisely a subgraph H of G, is selected for the next generation. We mutate the subgraph in the following order of steps. Pick a number k uniformly at random in the interval [0,1]. Let the mutation rate be μ, if $k \leq \mu$ then remove a vertex v from H and randomly choose a neighbor u of $H - \{v\}$ in G, $u \in \mathcal{N}(H - \{v\})$, the vertex u is then added to $H - \{v\}$ to create a subgraph $\big(H - \{v\}\big) \cup \{u\}$ for the next generation and if $k > \mu$, the subgraph will not be mutated. The mutation operator is operational when all subgraphs in the population have fitness smaller than one. That means in the population there is no isometric subgraph and the algorithm could not find any isometric subgraph of order "O". Therefore, by using mutation operator, we give

this chance to the subgraphs with high fitness value to evolve in the direction of isometric subgraphs. However, in distance-preserving graphs, existence of at least one isometric subgraph for each possible order is a necessary and sufficient condition.

5) Increasing order criterion: If there is a subgraph with fitness 1 in the population, the search algorithm has successfully found the isometric subgraph with order "O". Now, we can increase the order to "$O+1$" and repeat the searching procedure for this order. To do so, we use the same selection operator to generate the initial population for order "$O+1$". Moreover, we use a similar mutation operator to increase the order of all selected subgraphs. Let H be one of the selected subgraphs. To create a new graph using H, we randomly choose a neighbor w of $\mathcal{N}(H)$ in G and add w to the graph H. Now $H \cup w$ is the desired subgraph. We use $H \cup w$ as an element of the new population in which all subgraphs have order $|H|+1$.

As we discussed earlier, maximally distance preserved subgraphs are selected to reproduce next generation. The reason of this selection is that if an induced subgraph H of a graph G is not isometric, then there is a pair of vertices u, v in H such that $d_H(u,v) > d_G(u,v)$. If we randomly select a vertex w in $\mathcal{N}(H)$ then w less likely have this chance to be on a u-v geodesic in G.

6) Termination criterion: If an element of the population has order equal to $|G|$, or the population evolves for more than a user defined threshold of maximum generation, then the algorithm terminates and G is dp. If number of generation for one order exceeds the generation threshold (which is a user defined parameter), G might not be dp.

The pseudo-code of this method is given in Algorithm 1, and the details of this algorithm are presented in the Figure 3.

Algorithm 1 pseudocode of the evolutionary based search algorithm

Procedure:
Step 0. Set order "O" of subgraphs to 3.
Step 1. Generate N subgraphs of order "O".
Step 2. At generation "k" do the following:

a. Evaluate the fitness of subgraphs.
b. **If** the maximum fitness is 1.
- **If** "O" equals to the order of the main graph G.
  = G is dp (Stop)
- **Else:**
  = Increase "O" by 1 and go to step 1.

Else:
- Increase k by 1.
- **If** k exceeds the "$maximum\ generation$".
  = G might not be dp (stop).
- **Else:**
  = Generate k generation by using the recombination and mutation operators. The fittest subgraph should have higher chance to reproduce.
  = Go to step a.

IV. Experimental results

Here we present the results of our algorithm on graphs randomly generated of order 10 to 100. And also for each given order, we generated 10 graphs with different edge probabilities. The population size varies from 10 to 400. Experiments run under the Linux operating system on an Intel 10 1536 cluster of 192 dual core Intel 2200+t processors (High Performance Computing Laboratory, Michigan State University http:// hpcc.msu.edu). The mutation rates for our experiments would be .001, .002, .005, .01 and .05. We have 100 replicates for each experimental set up. Since there is no search algorithm other than brute force in the literature; therefore, we compare the performance of our algorithm with brute force algorithm.

A. Overall performance of the evolutionary algorithm.

In this section, we demonstrate how many times out of 100 runs, this algorithm terminates correctly as graphs, with different orders, are dp. Here, we fix the population size to 10, mutation rate to 0.01, and finally the probability of existence of an edge in graphs is 0.03. As we can see in Figure 4, the evolutionary algorithm terminates more than 95 percent correctly for graphs of order up to 100. Since this search problem is very time consuming the brute force algorithm failed to give a result in a limited time.

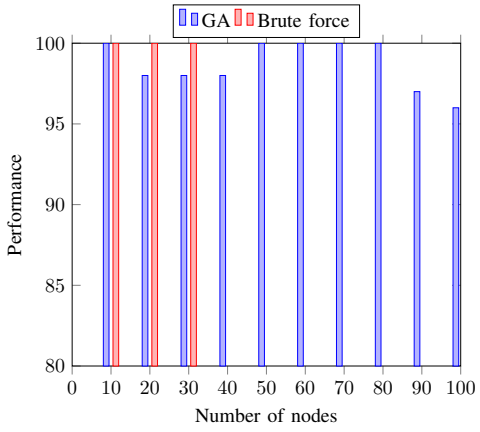

Fig. 4. Performance of the genetic and brute-force algorithms in finding dp property of randomly generated graphs. Note that the brute-force algorithm does not terminate for graphs of order greater than 30 in our time limit which is 450 seconds.

Moreover when the order of graphs are increased, the running time would be increased too, see Figure 5.

B. Effect of population size.

In this experiment, we check the effect of population size on the performance of the algorithm on a given graph of order 30. Here the mutation rate is fixed to 0.05, and the probability of existence of an edge in the generated graph is 0.4. Our choice of picking 0.4 as the edge probability makes the problem more challenging. The results are shown in Figures 6 and 7.

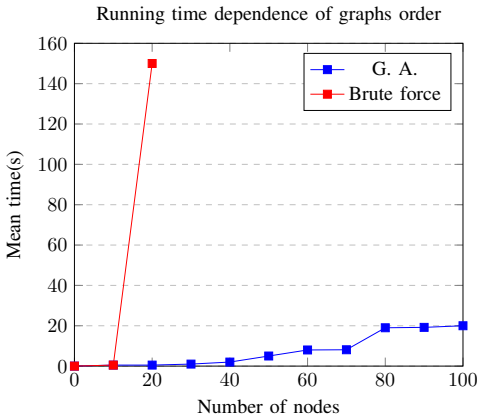

Fig. 5. Running time of the genetic and brute force algorithms. Brute-force algorithm does not terminate in our time limit which is 450 seconds for graphs of order greater than 30.

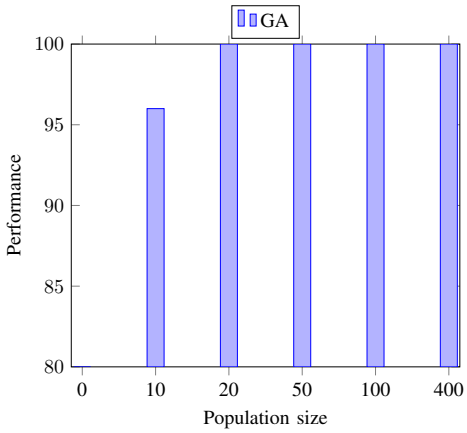

Fig. 6. Effect of population size on performance of genetic algorithm in finding dp property. Genetic algorithm successfully finds dp graphs when population size is greater than 20.

C. Effect of mutation rate

In this experiment, we consider the effect of mutation rate on the performance of the algorithm. Here, we fix the graph's order to 90, population size to 10, and the probability of existence of an edge in the graph is 0.4. As we see in Figures 8 and 9, the optimal value for mutation rate is 0.005.

D. Effect of edge probability

In this experiment, we check how the probability of existence of edges effects on the performance and running time of the algorithm. We set the order of graph to 100, population size to 10, and mutation rate to 0.001. As we see in Figure 10 when the edge probability goes up the performance is getting better. The selected graphs are dp, hence the brute force algorithm always gives result correctly.

V. Concluding Remarks

The problem of finding an isometric subgraph of order k for a given graph G is challenging, even if the graph has an isometric subgraph of order k. We conjecture this problem cannot be solved in polynomial time; thus, in this paper we apply the heuristic algorithms to address this problem. Among these algorithms, we utilized biologically inspired search algorithm to search for the dp property of a given graph. Currently, brute force algorithm is the only available algorithm for solving this problem. Therefore, brute force algorithm is used to check the performance of the proposed genetic algorithm. While the accuracy of proposed algorithm is more than $\%96$, its running time is reasonably faster than brute force approach. Subgraphs that maximally preserve the distances in the original graphs have higher chances to get selected and reproduce for the next generation. Although there is always a slight chance of failure when the input graph is sparse, by increasing the population size and increasing the mutation rate we can avoid such failures by losing the speed of the proposed algorithm. The presence of certain cycles can cause a graph not to be dp. In the existence of such cycles in the graph, we can attach these cycles to the offspring of selected ancestral parents to have isometric subgraphs containing cycles whose existence makes problem in finding isometric property. This

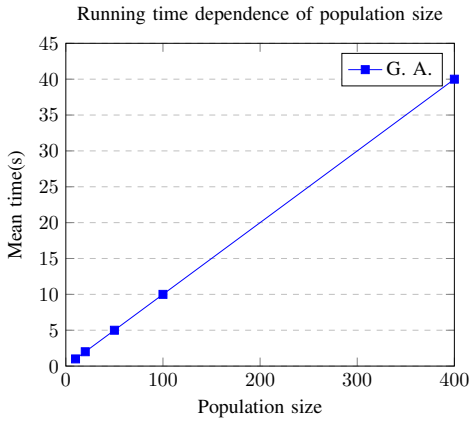

Fig. 7. Running times of the genetic algorithm for different population sizes.

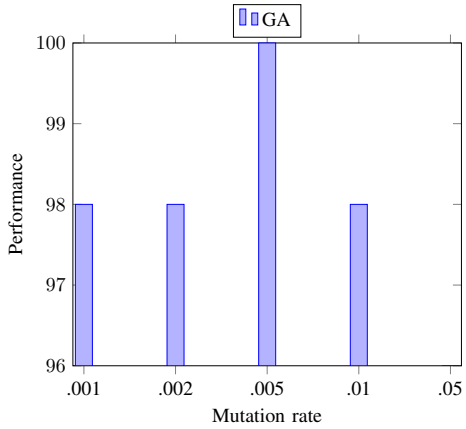

Fig. 8. Effect of mutation rate on performance of genetic algorithm in finding dp property. This figure demonstrates that the optimal mutation rate is 0.005.

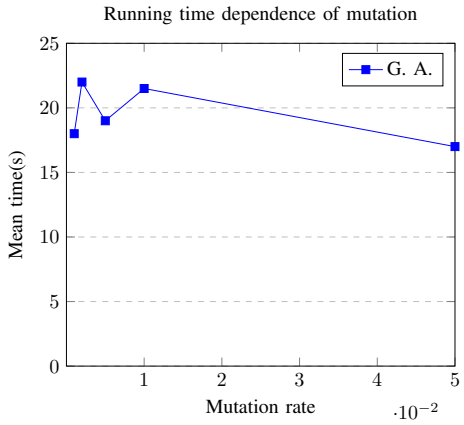

Fig. 9. Running time of the genetic algorithm for different mutation rates.

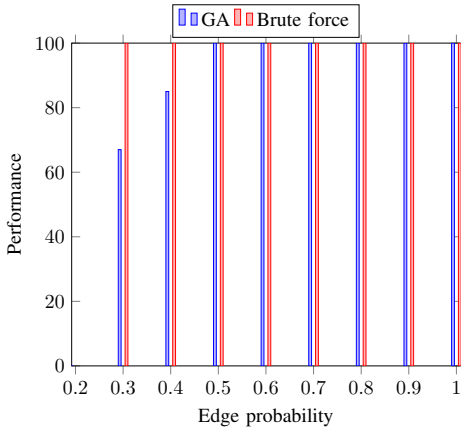

Fig. 10. Performances of genetic algorithm in comparison with the performance of brute-force algorithm in checking dp property. As the edge probability goes up we have more accurate results for the genetic algorithm.

algorithm can be easily extended to determine if a graph is sequentially distance preserving by having a harsh selection operator (r equals to 1).

Appendix

Here, we show that the smallest order of a dp Cartesian product of graphs whose factors are non-dp graphs is 80. This graph is sparse and is a really good example to see how algorithms are efficient for determining a graph is distance preserving or not. In order to find this graph we need some notations and results from [15], [24], which we present below. Let

$$\mathrm{DP}'(G) = \big\{A \subseteq V(G) \,\big|\, G - A \leq G\big\},$$

and

$$\mathrm{dp}'(G) = \big\{|A| \,\big|\, A \in \mathrm{DP}'(G)\big\}.$$

We used brute force algorithm and found all non-dp graphs of order 1 up to the order 9. The results are listed in the following table.

Table 1 presents the properties of non-dp graphs of oreder 5 up to 9. Column one shows the order of graphs. Column two gives the number of non-dp graphs, column three shows the number of non-dp graphs like G, such that $1 \notin \mathrm{dp}'(G)$, and finally column four gives the number of non-dp graphs like G, such that $1 \in \mathrm{dp}'(G)$ and $2 \notin \mathrm{dp}'(G)$, for the corresponding graphs in the column one.

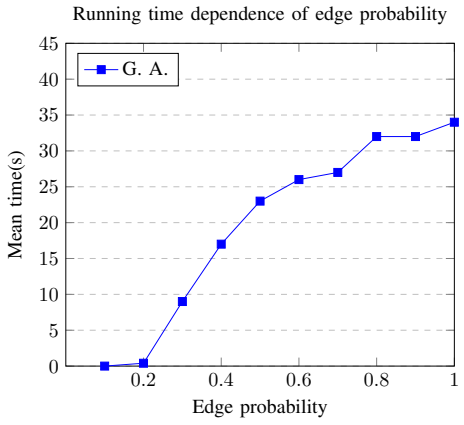

Fig. 11. Running time of the genetic algorithm for randomly generated graph of order 100 with different edge probabilities.

Table 1

Graphs order	#of non-dp graphs	#of non-dp graphs $1 \notin \mathrm{dp}'(G)$	#of non-dp graphs $1 \in \mathrm{dp}'(G)$, $2 \notin \mathrm{dp}'(G)$
5	1	1	0
6	1	1	0
7	4	4	0
8	19	14	4
9	183	168	15

Theorem V.1. *[24, Theorem 4.1] Let G be a graph such that $\mathrm{girth}(G) \geq 5$ and such that every vertex is either a cut vertex or in a cycle. Then G is not dp.*

Proposition V.2. *The C_5 and C_6 are the only non-dp graphs of order less than 7.*

Lemma V.3. *[15, Theorem 3.3] For nonempty subsets A and B in the vertex set of graphs G and H respectively, $A \times B \in \mathrm{DP}'(G \,\square\, H)$ if and only if $A \in \mathrm{DP}'(G)$ and $B \in \mathrm{DP}'(H)$.*

Therefore:

Corollary V.4. *Let G and H be arbitrary graphs,*
(a) If $1 \notin \mathrm{dp}'(G)$ then $1 \notin \mathrm{dp}'(G \,\square\, H)$.
(b) If $2 \notin \mathrm{dp}'(G \,\square\, H)$ then $2 \notin \mathrm{dp}'(G)$ or $2 \notin \mathrm{dp}'(H)$.

By the Proposition V.2 and Corollary V.4, any dp Cartesian product graph cannot have a non-dp factor of order smaller than 7.

Lemma V.5. *Let G' be an induced subgraph of G such that $\mathrm{diam}(G') \leq 2$ then $G' \leq G$.*

Lemma V.6. *[24, Lemma 4.3] Let G contain a vertex v such that every pair of non-adjacent $u, w \in \mathcal{N}(v)$ are in a 4-cycle in G. If $G - v$ is dp then so is G.*

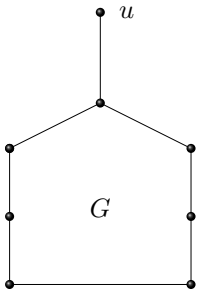

Fig. 12. A non-dp graph G with order 8 such that $1 \in \mathrm{dp}'(G)$.

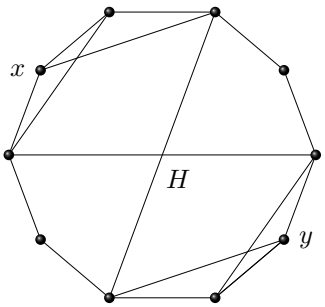

Fig. 13. A non-dp graph H of order 10 such that $1 \in \mathrm{dp}'(H)$ and $2 \in \mathrm{dp}'(H)$.

Proposition V.7. *If G is a non-dp graph of order 7 then $1 \notin \mathrm{dp}'(G)$.*

By the Corollary V.4 and Proposition V.7, any dp Cartesian product graph cannot have a non-dp factor of order 7. But there is a non-dp graph G of order 8 such that $1 \in \mathrm{dp}'(G)$, that is shown in Figure 12 in which G does not have any isometric subgraph of order 6 so it is not dp and we use this graph as a factor of our desired dp Cartesian product graph.

Proposition V.8. *If G is a non-dp graph of order 8 and $1 \in \mathrm{dp}'(G)$ then $2 \notin \mathrm{dp}'(G)$.*

Now if we use the mentioned graph G as the first factor of our desired Cartesian product graph, the Proposition V.8 gives $2 \notin \mathrm{dp}'(G)$. On the other hand by Corollary V.4(b), if $2 \notin \mathrm{dp}'(G)$ and $2 \notin \mathrm{dp}'(H)$ then $2 \notin \mathrm{dp}'(G \,\square\, H)$. Therefore in the second factor H of our Cartesian product graph, we must have $1 \in \mathrm{dp}'(H)$ and also $2 \in \mathrm{dp}'(H)$.

Table 1 shows that there is no non-dp graph G, of order less than 10 such that $1 \in \mathrm{dp}'(G)$ and $2 \in \mathrm{dp}'(G)$. But such a this graph H, of order 10 exists which is shown in Figure 13. It is easy to see that $\{x\} \in \mathrm{DP}'(H)$ and $\{x, y\} \in \mathrm{DP}'(H)$, but $3 \notin \mathrm{dp}'(H)$.

Theorem V.9. *The smallest order of a dp Cartesian product of graphs whose factors are not dp is 80.*

Proof. Let G and H be graphs in Figure 12 and 13 respectively. We show that $G \,\square\, H$ is dp. As isometric property is transitive, using Lemma V.3 one can see

$$\big(A \times \{x\}\big) \cup \big(B \times \{y\}\big) \cup \big(C \times S\big) \subseteq \mathrm{DP}'(G \,\square\, H),$$

where $A, B, C \in \mathrm{DP}'(G)$ and $x, y \notin S \in \mathrm{DP}'(H)$. Thus

$$(a \times 1) + (b \times 1) + (c \times s) \subseteq \mathrm{dp}'(G \,\square\, H), \qquad (1)$$

where $a, b, c \in \mathrm{dp}'(G) = [8]_0 - \{2\}$ and $s \in \mathrm{dp}'(H) - \{1, 2\} = [10]_0 - \{1, 2, 3\}$. Using Equation 1 one can find $[80]_0 \subseteq \mathrm{dp}'(G \,\square\, H)$. Moreover by above aforementioned in this section the order of $G \,\square\, H$ is minimum. □

References

[1] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[2] H.-J. Bandelt and H. M. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

[3] B. Bollobás. The diameter of random graphs. *Transactions of the American Mathematical Society*, 267(1):41–52, 1981.

[4] J. A. Bondy and U. S. R. Murty. *Graph theory with applications*, volume 6. Macmillan London, 1976.

[5] K. M. Bryden, D. A. Ashlock, S. Corns, and S. J. Willson. Graph-based evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 10(5):550–567, 2006.

[6] D. Coudert, G. Ducoffe, N. Nisse, and M. Soto. *Distance-preserving orderings in graphs*. PhD thesis, Inria Sophia Antipolis, 2016.

[7] G. Damiand, M. Habib, and C. Paul. A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. *Theoretical Computer Science*, 263(1):99–111, 2001.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[9] M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. Winfield. *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings*, volume 5217. Springer, 2008.

[10] R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[11] R. Graham and P. Winkler. On isometric embeddings of graphs. *Transactions of the American mathematical Society*, 288(2):527–536, 1985.

[12] E. Howorka. A characterization of distance-hereditary graphs. *The quarterly journal of mathematics*, 28(4):417–420, 1977.

[13] W. Imrich and S. Klavzar. *Product graphs*. Wiley, 2000.

[14] M. Jenkinson and S. Smith. A global optimisation method for robust affine registration of brain images. *Medical image analysis*, 5(2):143–156, 2001.

[15] M. Khalifeh, B. E. Sagan, and E. Zahedi. Distance preserving graphs and graph products. *arXiv preprint arXiv:1507.04800*, 2015.

[16] S. Klavzar. Applications of isometric embeddings to chemical graphs. *DIMACS Ser. Discrete Math. Theoret. Comput. Sci*, 51:249–259, 2000.

[17] J. W. Moon and L. Moser. *Almost all (0, 1) matrices are primitive*. springer, 1965.

[18] R. Nowakowski and I. Rival. On a class of isometric subgraphs of a graph. *Combinatorica*, 2(1):79–90, 1982.

[19] R. Nussbaum and A.-H. Esfahanian. Preliminary results on distance-preserving graphs. *Congressus Numerantium*, 211:141–149, 2012.

[20] R. Nussbaum, A.-H. Esfahanian, and P.-N. Tan. Clustering social networks using distance-preserving subgraphs. In *The Influence of Technology on Social Network Analysis and Mining*, pages 331–349. Springer, 2013.

[21] J. P. Smith and E. Zahedi. On distance preserving and sequentially distance preserving graphs. *arXiv preprint arXiv:1701.06404*, 2017.

[22] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[23] D. Yan, J. Cheng, W. Ng, and S. Liu. Finding distance-preserving subgraphs in large road networks. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 625–636. IEEE, 2013.

[24] E. Zahedi. Distance preserving graphs. *arXiv preprint arXiv:1507.03615*, 2015.