Performance Comparison of Randomized Local Search, (1+1)-Evolutionary Algorithm and Genetic Algorithm for Graph Isomorphism Problem using Permutation Matrix Search Space

Akshitha Puri, Gunveen Batra, and Ajitha Shenoy K B*, *Member, IAENG*

***Abstract*—Graph isomorphism problem (GIP) is a class NP Problem and is a well-studied problem in graph and complexity theory. It is still an open problem, whether GIP is in class P or NP-complete. In the literature, no meta-heuristics are applied to GIP. However, few studies in the literature show that the meta-heuristic algorithms are applied to the subgraph isomorphism problem, which is a known NP-Complete problem and extension of GIP. This article proposes a new search space called permutation matrix space for GIP. Two search space elements are neighbors if one is obtained from the other by swapping any two distinct rows. The work shows that the magnification of permutation matrix space is bounded below by n/2. Magnification is one of the main structural properties of a search graph which implies the number of arcs going out from any cut-set in the search graph. The article studies the performance of Randomized Local Search (RLS), (1+1)-Evolutionary Algorithm (EA), and Genetic Algorithm (GA) for the GIP. Using the concepts of Markov chain Coupling, this article proves that the Markov chains associated with the randomized local search mixes rapidly, i.e., the mixing time is bounded above by $O(n^2)$. Experimentally, RLS outperforms (1+1)-EA and GA on the permutation matrix space.**

***Index Terms*—Meta-heuristic, Search Space, Evolutionary Algorithm, Randomized Local Search, Genetic Algorithm, Markov Chain, Mixing Time, Graph Isomorphism, Magnification, Coupling.**

I. Introduction

GRAPH Isomorphism Problem (GIP) is a well-studied combinatorial optimization problem in the field of computational complexity. It is a class NP problem [1]. It continues to be an open question whether the problem is in class P or NP-complete. The recent result says that GIP is in the complexity class SPP [2]. GIP has many applications in the area such as image processing [3]–[5], protein structure analysis [6]–[9], fingerprint authentication [10], [11] and pattern recognition [12], [13] etc.

GIP is relabeling of the vertices of one graph to obtain another graph. Relabeling takes all permutations of the vertices of graph G, thus it has a complexity of $O(n.n!))$, where n-denotes the number of vertices in the graph. Till date, there is no algorithm to solve this problem in polynomial time. L Babai, in 2016 proved that GIP can be solved in quasi-polynomial time i.e. $\exp\left((\log n)^{O(1)}\right)$ [14] and this is the best-known complexity for GIP till date. The claimed results had some flaws which L. Babai fixed in the year 2017 [15]. Daniel Wiebking extended the result of L. Babai in the year 2020. He proved that the run time of the GIP test is $n^{polylogk}$, where n and k denote the number of vertices and the minimum treewidth of the given graph respectively [16]. K. Liu *et al.*, in the year 2019 designed Discrete-Time Quantum Walk (DTQW) for GIP and proved that the computational complexity of simulating DTQW on the classical computer is $O(n^6)$ (the quantum walk is the quantum analogue of the classical random walk) [17].

Meta-heuristics like Randomized Local Search (RLS), (1+1)-Evolutionary Algorithm (EA) ([18], [19]) and Genetic algorithm (GA) [20]–[23], Particle Swarm Optimization (PSO) [24], Ant Colony Optimization (ACO) [25] etc., are, used to get optimum or near to optimum solutions for hard optimization problem [26]–[43]. In the literature, meta-heuristics are applied to solve the subgraph isomorphism problem (which is an extension of GIP) and it is a known NP-complete problem [1], [44], [45]. This is the first attempt where a meta-heuristic algorithm is being applied exclusively to GIP. The paper proposes a new search space for GIP called permutation matrix space and, studies the performance of three meta-heuristic algorithms, namely, RLS, (1+1)-EA, and GA on it. The major contribution of the paper is listed below:

1) Defined a new search space called permutation matrix space for GIP (Definition II.4).
2) Proved that the magnification of the permutation matrix space is at least $\frac{n}{2}$ (Theorem II.6), which indicates more edges or arcs are going out from any cut-set.
3) Designed RLS algorithm, (1+1)-EA, GA for GIP using permutation matrix space (Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm 4).
4) Using the concept of Markov chain and Coupling it is proved that the Mixing time of the Markov chains associated with the RLS algorithm is bounded by $O(n^2)$ (Theorem III.3).
5) Experimental results show that the RLS algorithm outperforms (1+1)EA and both the versions of GA (GA-1, GA-2) for GIP using permutation matrix space (Section V).

In the next section, the permutation matrix space is defined

Manuscript received June 07, 2021; revised March 15, 2022.

Akshitha Puri is an Undergraduate Student of the Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal - 576 104, Karnataka, INDIA e-mail: akshitapuri@gmail.com

Gunveen Batra is an Undergraduate Student of the Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal - 576 104, Karnataka, INDIA e-mail: emerald165@gmail.com

Ajitha Shenoy K B is a Senior Associate Professor in the Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal - 576 104, Karnataka, INDIA (Corresponding Author e-mail: ajith.shenoy@manipal.edu)

and the structural property of the same is studied.

II. Permutation Matrix Space for GIP

Graph isomorphism problem [46], [47] is to check whether the two graphs are structurally similar, more formally:

Definition II.1. *GIP Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are two graphs with vertex and edge set V_1, E_1 and V_2, E_2 respectively. An isomorphism from graph G_1 to graph G_2) is a bijective (one-one and onto) function $f : V_1 \rightarrow V_2$ such that, for each edge $(u, v) \in E_1$, there is an edge $(f(u), f(v)) \in E_2$ and vice versa. If there is an isomorphism between two graphs G_1 and G_2 then they are isomorphic.*

From the definition of GIP, the isomorphism preserves adjacency and non-adjacency between two graphs and implies that they are structurally the same. Note that, the concept of permutation matrix is required to define permutation matrix space for GIP.

Definition II.2. *(Permutation Matrix [48], [49]) Permutation matrix B is an $n \times n$ square binary matrix derived from permuting (or swapping) rows of identity matrix I. Permutation matrix has the following properties:*

- *Permutation matrix is non singular.*
- *Determinant of permutation matrix is always ± 1.*
- *$B^{-1} = B^T$ i.e. $B \cdot B^T = I$, where B^{-1} and B^T denote inverse and transpose of matrix B.*
- *If B_1 and B_2 are permutation matrices then, $B_1 \cdot B_2$ and $B_2 \cdot B_1$ are also permutation matrices.*
- *If B is a permutation matrix and A is a square matrix then $B \cdot A$ permutes rows of A and $A \cdot B$ permutes columns of A.*

The following proposition summarizes the relationship between permutation matrices and adjacency matrices of two isomorphic graphs.

Proposition II.3. *[50] Let A_1 and A_2 denote adjacency matrix of two graphs G_1 and G_2 respectively. If G_1 and G_2 are isomorphic, then there exists a permutation matrix B such that $A_2 = B \times A_1 \times B^T$.*

Next, the permutation matrix space for GIP is defined.

Definition II.4. *(Permutation Matrix Space) Given are two graphs, G_1 and G_2, and their adjacency matrices A_1 and A_2 respectively.*

- *Search Space Elements: Permutation Matrices (as defined in Definition II.2).*
- *Neighbourhood Structure: Two search space elements, B_1 and B_2, are* neighbours *if B_2 is obtained by swapping any two rows of B_1.*
- *Fitness Function: For any search space element B_1,* fitness *$f(B_1)$ is defined as number of corresponding identical elements between the matrices $A' = (B_1 \times A_1 \times B_1^T)$ and A_2. Formally,*

$$f(B_1) = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j},$$

where

$$x_{i,j} = \begin{cases} 1, & \text{if } A'(i,j) = A_2(i,j) \\ 0, & \text{otherwise} \end{cases}$$

Note that if two graphs are isomorphic then search space has at least one element B_i for which fitness $f(B_i) = n^2$. This implies that by pre and post multiplying adjacency matrix A_1 with permutation matrices B_i and B_i^T, $A' = A_2$ (i.e., $A' = (B_i \times A_1 \times B_i^T) = A_2$) matrix is obtained. This implies A' is obtained by permuting rows and columns of matrix A_1. From the definition of the permutation matrix space (Definition II.4), it is clear that each permutation matrix (i.e., each search space element) has $d = \frac{n(n-1)}{2}$ neighbours and the total number of permutation matrices (search space elements) are $n!$. If Ω denotes permutation matrix space then $|\Omega| = n!$.

One of the important structural properties of any search space is Magnification. Finding a lower bound on magnification gives a clear idea about the minimum number of arcs (or edges) going out from any cut set. Formally, magnification is defined as:

Definition II.5. *(Magnification [51], [52]) Let Ω denote permutation matrix space. For any subset $S \subset \Omega$ and $S \neq \emptyset$, let $E(S, \overline{S})$ denote the cut set in Ω defined by S, that is, $E(S, \overline{S})$ is the set of edges with one end point in S and other end point in $\overline{S} = \Omega - S$, then magnification ($\mu(\Omega)$) is:*

$$\mu(\Omega) = \min_{0<|S|\leq|\Omega|/2} \frac{E(S, \overline{S})}{|S|} \tag{1}$$

Note that, large magnification implies that there is more number of arcs (or edges) going out from any cut set $E(S, \overline{S})$ in the permutation matrix space. Therefore, any meta-heuristic algorithm may use the permutation matrix space profitably to overcome local optimum. Now to find a lower bound for magnification of permutation matrix space, the concept of canonical path method (as defined in [51], [52]) is used, which is explained in the following theorem.

Theorem II.6. *Permutation matrix space Ω has magnification at least $\frac{n}{2}$.*

Proof: For each pair of states B_1 and B_2 in the permutation matrix space, define a unique path $\eta_{B_1 B_2}$ called canonical path as follows: Let $l_1, l_2, \ldots, l_k$ ($k \leq n$), denote row indices, arranged in increasing order, where permutation matrix B_1 differs from B_2. The canonical path $\eta_{B_1 B_2}$ is a sequence $P_0 = B_1, P_1, \ldots, P_k = B_2$, where P_i is a permutation matrix obtained by swapping i^{th} and j^{th} rows of $P_{(i-1)}$, and $j > i$. That is, at i^{th} stage, swap i^{th} and j^{th} ($j > i$) row of $P_{(i-1)}$ so that first i-rows of P_i match with the first i-rows of B_2. At the i^{th} stage, swap takes place only if they differ at the i^{th} row. By the definition of permutation matrices (Definition II.2) such a row-j always exists. The way in which the canonical path is defined, it is clear that all the edges in the canonical path between B_1 and B_2 are distinct and appear exactly once.

Let $E_{R,S} = (R, S)$ denote an edge in the canonical path $\eta_{B_1 B_2}$, where S is obtained by interchanging or swapping the i th and j th row of R ($j > i$). Let $P(E_{R,S})$ be the set of all canonical paths η_{B_i, B_j} which use edge $E_{R,S}$, for some B_i and B_j in Ω. Now to prove that a function $f : P(E_{R,S}) \rightarrow \Omega$ is an injective function:

Let B_2 and B_3 denote two permutation matrices and $B_2 \neq B_3$. Let $\eta_{B_1 B_2}, \eta_{B_1 B_3} \in P(E_{R,S})$ be canonical paths from B_1 to B_2 and B_1 to B_3 respectively, which make use of edge $E_{R,S}$.

- If $B_2 = B_3$ then, from the definition of canonical path stated above, it is evident that the canonical path sequence is equivalent. i.e., $\eta_{B_1B_2} = \eta_{B_1B_3}$.
- Next, if $\eta_{B_1B_2} \neq \eta_{B_1B_3}$, i.e.,

  $$B_1, P_1, \ldots, (P_k = B_2) \neq B_1, P'_1, \ldots, (P'_k = B_3)$$

  which implies that there exists at least one edge

  $$(P_{(i-1)}, P_i) \in \eta_{B_1B_2}$$

  which is not equal to $(P'_{(i-1)}, P'_i) \in \eta_{B_1B_3}$. Thus, it can be inferred that, at i-th stage, the swaps while moving from state B_1 to B_2 are different than that from state B_1 to B_3 (Refer the way in which canonical path defined). Therefore, $B_2 \neq B_3$.

Therefore, function f is injective. Hence, the number of canonical paths which pass through an edge are bounded above by $|\Omega| = n!$. To get tighter bound consider the path from permutation matrix

$$B_1 = (r_1, r_2, \ldots, r_n) \text{ to } B_2 = (r'_1, r'_2, \ldots, r'_n)$$

, where r_i and r'_i for all $1 \leq i \leq n$ denote rows of permutation matrix B_1 and B_2 respectively. Consider an edge $(P_i, P_{(i+1)}) \in \eta_{B_1B_2}$. Assume that P_{i+1} obtained by P_i by swapping i-th and j-th row, where $j > i$. Then,
$P_i = (r'_1, r'_2, \ldots, r'_{i-1}$, remaining rows of B_2 in any random order)
$P_{i+1} = (r'_1, r'_2, \ldots, r'_{i-1}, r'_i$, remaining rows of B_2 in same order of P_i except at i and j-th position)
Therefore, number of canonical paths that pass through above edge P_i and P_{i+1} is less than or equal to $(i-1)!\cdot(n-i-1)!$ i.e.,

$$(i-1)!\cdot(n-i-1)! \leq i!\cdot(n-1-i)! = \frac{(n-1)!}{C(n-1,i)} \leq (n-1)!$$

(since $C(n-1,i) = \frac{(n-1)!}{i!(n-i-1)!} \geq 1$.)

Now to bound the magnification, consider any non-empty set $S \subseteq \Omega$ and $|S| \leq \frac{|\Omega|}{2}$. Let $\Gamma_{S\overline{S}}$ represent all canonical paths η_{PQ}, which start at search space element P in S and end at search space element Q in $\overline{S}$. Then,

$$|\Gamma_{S\overline{S}}| = |S| \times |\overline{S}| \geq \frac{|S| \times |\Omega|}{2}$$
$$\left(\text{since } |S| \leq \frac{|\Omega|}{2},\ |\overline{S}| > \frac{|\Omega|}{2}\right)$$

Every path in $\Gamma_{S\overline{S}}$ has to traverse an edge (R, S) in $E(S, \overline{S})$. It is proved that through any edge (say (R, S)), at most $(n-1)!$ canonical paths can pass through. Therefore,

$$|E(S,\overline{S})| \cdot (n-1)! \geq \Gamma_{S\overline{S}} \geq \frac{|S| \times |\Omega|}{2}$$

This gives,

$$\mu(\Omega) = \frac{|E(S,\overline{S})}{|S|} \geq \frac{n!}{2(n-1)!} \geq \frac{n}{2} \tag{2}$$

■

From the way canonical path is defined, it is clear that between any two search space elements in Ω there is a $O(n)$ length path. Hence, the diameter (i.e., longest shortest path) of the search graph is bounded above by $O(n)$. The next section describes the design of the RLS algorithm for GIP using permutation matrix space and it is analyzed using the concept of the Markov chains and mixing time.

III. RLS for GIP

Randomized local search (RLS) is a meta-heuristic search strategy. It greedily searches the neighborhood for optimum/near to optimum solutions in the search space. The detailed RLS algorithm for GIP is given in Algorithm 1. Mutation for each individual (search space element) is defined as follows:

Definition III.1. *(Mutation) For any individual (i.e. search space element) $B_1 \in \Omega$, mutation is defined as swapping of any two distinct rows of B_1 uniformly, at random. i.e., $swap(B_{1i}, B_{1j})$, where $i \neq j$ and B_{1i} and B_{1j} denote i-th and j-th row of B_1 respectively.*

Algorithm 1 RLS for GIP

1: Let $B_1 = I$ be initial starting state, where I is an $n \times n$ identity matrix.
2: No. of steps $= 0$ and Best Cost $= f(B_1)$
3: **while** ((No. of steps $<$ Max iterations) and (Best Cost $< n^2$)) **do**
4: Select a row of B_1 uniformly, at random and mutate it as defined in Definition III.1. [i.e., select any one of the neighbour of B_1 uniformly, at random.]
5: Calculate the fitness $f(B_1)$ and $f(B_2)$ of B_1 and B_2 as defined in Definition II.4.
6: **if** $f(B_2) \geq f(B_1)$ **then**
7: Set $B_1 = B_2$ with probability $\frac{1}{2}$.
8: Best Cost $= f(B_2)$
9: **end if**
10: $step = step + 1$
11: **end while**

The concept of the Markov chain is used to analyze the performance of RLS. For basic definitions and concepts related to Markov chains, Coupling and Mixing Time refer [51]–[55]. The RLS algorithm for GIP (Algorithm 1) induces a Markov chain on the permutation matrix space Ω with transition probabilities defined as:

$$P_{B_1,B_2} = \begin{cases} 0, & \text{if } B_2 \notin N(B_1) \\ 0, & \text{if } B_2 \in N(B_1), B_2 \neq B_1 \\ & \&f(B_2) < f(B_1) \\ \frac{1}{2d}, & \text{if } B_2 \in N(B_1), B_2 \neq B_1 \\ & \&f(B_2) \geq f(B_1) \\ 1 - \sum_{B_i \neq B_1} P_{B_1,B_i}, & \text{if } B_2 = B_1, \\ & B_i \in N(B_1) \end{cases}$$

where $N(B_1)$ denotes neighbours of B_1 and $d = \frac{n(n-1)}{2}$ denotes degree (or number of neighbors) of each node in the permutation matrix space. Note that the proposed RLS Algorithm 1 is a slightly modified version of RLS. In this modified version of RLS for GIP, with probability $\frac{1}{2}$, the Markov chain remains in the same state. This property is required to make the chain aperiodic. Also, permutation matrix space is connected. Therefore, the Markov chain running on state space is ergodic (i.e. irreducible and aperiodic). Thus using the fundamental theorem of Markov chains ([51]–[55]), the chain associated with RLS has unique stationary distribution (say π). Note that for the Markov chain induced by RLS, $\pi_{B_1}P_{B_1,B2} \neq \pi_{B_2}P_{B_2,B_1}$. Hence the Markov chain

is not reversible. The next section describes the concept of Coupling to bound the mixing time of the chain.

Definition III.2. *Coupling: A coupling of a Markov chain with transition matrix P is a pair process (X_t, Y_t) such that,*

- $Pr[X_{t+1} = b|X_t = a] = P(a, b) = Pr[Y_{t+1} = b|Y_t = a]$.
- *if $X_t = Y_t$, then $X_{t+1} = Y_{t+1}$.*

Stopping time (i.e. until the time two process X_t and Y_t meet) is defined as $T_{xy} = \min\{t : X_t = Y_t|X_0 = x, Y_0 = y\}$. The relationship between Coupling time (stopping time) and mixing time is given as:

$$t_{\text{mix}}(\varepsilon) \leq \max_{x,y} Pr[T_{xy} > t]$$

Theorem III.3. *The mixing time $t_{\text{mix}}(\varepsilon)$ of the chain associated with RLS is bounded by $O(n^2)$*

Proof: Consider the scenario of swapping two rows of a permutation matrix say r_i and r_j. Define a Coupling as X_t and Y_t, choosing same r_i and r_j at each step. This coupling ensures that the distance between X and Y is non-increasing. Let d_t denote the distance between X_t and Y_t for the number of positions at which two permutation matrices differ. Then the following two cases arise:

1) If row r_i is in the same position in both the permutation matrices, then $d_{t+1} = d_t$.
2) If row r_i is in different positions in the two permutation matrices, there are two cases:
   a) If row r_j is in the same position in both the permutation matrices, then $d_{t+1} = d_t$.
   b) Otherwise $d_{t+1} \leq d_t - 1$.

This implies that there is a decrease in the distance only in case 2(b). Therefore,

$$Pr[d_{t+1} < d_t] = \left(\frac{d_t}{n}\right)^2$$

It can be deduced that the time for d_t to decrease from value d is dominated by a geometric random variable with mean $(n/d)^2$. Therefore,

$$E[T_{xy}] \leq \sum_{d=1}^{n} \left(\frac{n}{d}\right)^2 = O(n^2).$$

Using Markov inequality,

$$Pr[T_{xy} > cn^2] < c' = \frac{1}{2\varepsilon} \text{ for a suitable constant } c.$$

Thus, $t_{\text{mix}}(\varepsilon) \leq cn^2$. ∎

In the next section (1+1)-EA and GA for GIP using permutation matrix space are discussed.

IV. (1+1)-EA and GA for GIP

(1+1)-EA and GA are evolutionary algorithms that are extensively used meta-heuristics in the literature to get *good* solutions to hard optimization problems. The first subsection defines (1+1)-EA for GIP and the second subsection defines GA for GIP.

A. (1+1)-EA for GIP

(1+1)-EA is a slightly modified version of RLS. (1+1)-EA mutates every row with probability $1/n$ in each iteration, unlike RLS which selects one row uniformly, at random for mutation. This slight modification makes (1+1)-EA more capable of escaping from local optima compared to RLS. Both RLS and (1+1)-EA are well studied in the literature for the analysis of the evolutionary algorithm. The detailed (1+1)-EA for GIP is given in Algorithm 2. The next subsection describes GA for GIP using permutation matrix space.

Algorithm 2 (1+1)-EA for GIP

```
Let B_1 = I be initial starting state, where I is an n × n identity matrix.
No. of steps = 0 and Best Cost = f(B_1)
while ((No. of steps < Max iterations) and (Best Cost < n^2)) do
    Mutate each row of B_1 as defined in Definition III.1 with probability 1/n and obtain B_2.
    Calculate the fitness f(B_1) and f(B_2) of B_1 and B_2 as defined in Definition II.4.
    if f(B_2) ≥ f(B_1) then
        Set B_1 = B_2
        Best Cost = f(B_2)
    end if
    step = step + 1
end while
```

B. GA for GIP

Genetic algorithm is a popular evolutionary algorithm technique used for finding near to optimum or *good* solutions for many hard optimization problems. GA is unlikely to get stuck at local optima if parameter settings are appropriate. This work proposes two versions of GA for GIP. In GA-1 (version1: simple basic version) population size equals 2 and the number of offspring generated is 2 and in GA-2 (version2: generic) population size equals k and the number of offspring generated is m. GA-2 uses tournament selection with tournament size 2 for selecting the best $k/2$ individuals from the population. Using the $\ell = k/2$ individuals, create $C(\ell, 2)$ pairs, and each pair can produce two offspring (as defined in Definition IV.1). Therefore, number of offspring per generation is $m = 2 \times C(\ell, 2)$. Crossover and mutation for genetic algorithm are defined as follows:

Definition IV.1. *(Crossover and Mutation)*

- *Crossover: Create two children/offspring C_1 and C_2 from parents B_1 and B_2 as $C_1 = B_1 \cdot B_2$ and $C_2 = B_2 \cdot B_1$ [Note that the product of two permutation matrices is again a permutation matrix (refer Definition II.2)].*
- *Mutation: Swap rows of B_1 to mutate B_1 i.e., select any neighbour of B_1 uniformly at random (as defined in Definition II.4).*

The two versions of genetic algorithm GA-1 and GA-2 for GIP is using permutation matrix space is discussed in the Algorithm 3 and Algorithm 4 respectively.

Algorithm 3 GA-1 for GIP

Population Size $k = 2$, Number of children per generation $m = 2$.
Select any two individuals (two permutation matrices) as population say B_1 and B_2.
Calculate fitness of population as defined in Definition II.4. i.e. $f(B_1)$ and $f(B_2)$.
initialize step $= 0$ and Best Cost $= \min\{f(B_1), f(B_2)\}$.
while ((step $<$ Max iterations) and (Best Cost $< n^2$)) **do**
    Select both individuals from the population
    Generate 2 offspring using cross over as defined in Definition IV.1 with crossover probability $P_{crossover}$.
    Apply mutation on 2 offspring with probability $P_{mutation}$
    Calculate fitness of all the offspring.
    Out of 4 individuals (2 individuals in the population and 2 individuals are offspring generated) , the best 2 candidates will go for next generation. Rename them as B_1, B_2.
    if Best Cost $< \max\{f(B_1), f(B_2)\}$ **then**
        Best Cost $= \max\{f(B_1), f(B_2)\}$
    end if
    $step = step + 1$
end while

Algorithm 4 GA-2 for GIP

Population Size k, Number of children per generation m.
Initially select k individuals by performing row swapping operation $B_1 = I$ (as defined in Definition II.4), where I is $n \times n$ identity matrix (i.e., select any k permutation matrices as starting states).
Calculate fitness of population as defined in Definition II.4. i.e. $f(B_1), f(B_2), \ldots, f(B_k)$.
initialize step $= 0$ and Best Cost $= \min f(B_1), f(B_2), \ldots, f(B_k)$.
while ((step $<$ Max iterations) and (Best Cost $< n^2$)) **do**
    Select the best $\ell = \frac{k}{2}$ candidates using tournament selection with tournament size 2.
    Generate $m = C(\ell, 2) \times 2 = (\ell(\ell-1))$ offspring using cross over as defined in Definition IV.1 with crossover probability $P_{crossover}$.
    Apply mutation on m offspring with probability $P_{mutation}$
    Calculate fitness of all the m offspring
    Out of the $(k+m)$ individuals (k being the population size and m being the number of offspring generated) , the best k candidates will go for next generation. Rename them as $B_1, B_2, \ldots, B_k$.
    if Best Cost $< \max\{f(B_1), f(B_2), \ldots, f(B_k)\}$ **then**
        Best Cost $= \max\{f(B_1), f(B_2), \ldots, f(B_k)\}$
    end if
    $step = step + 1$
end while

The next section experimentally compares the performances of RLS, (1+1)-EA, GA-1, and GA-2 for GIP using permutation matrix space.

V. Results and Discussion

To test RLS, (1+1)-EA and GA algorithms, random graphs are used and they are constructed using Erdös Rényi [56] model $G(n, p)$. In $G(n, p)$, each edge in the graph has a probability p, independent from every other edge. The parameter p is a weight function. As p increases from 0 to 1, the graph is likely to have more edges. If $p > \frac{(1+\varepsilon)\ln n}{n}$, then a graph will almost surely be connected. For comparing the performance of different meta-heuristics considered, created the graph G_1 by taking different values of n and p. The isomorphic graph G_2 is constructed by permuting the corresponding rows and columns of the adjacency matrix of G_1 randomly for n iterations.

In Theorem III.3, it is proved that the mixing time of the Markov chain associated with the proposed RLS is bounded by $O(n^2)$. Therefore, executed RLS and (1+1)-EA for $10n^2$ iterations. GA-1 and GA-2 generate 2, and m individuals per iteration respectively, whereas both RLS and (1+1)-EA generates 1 individual per iteration. Therefore, to have a fair comparison, executed GA-1 and GA-2 for $5 \times n^2$ and $10n^2/m$ iterations respectively. This ensures that the total number of search space elements searched using all the algorithms is $10n^2$. Parameter settings used for For GA-1 and GA-2 are given in Table I. Various combinations of mutation and cross-over probabilities are considered (i.e., mutation probability in the range from 0 to 0.5 and cross-over probability in the range of 0.6 to 1) and, based on the performance, finally selected crossover and mutation probability as 0.8 and 0.5 respectively. For GA-2, different population size (6, 10, and 20) is set to see the effect of population size on the performance. The observation indicates that considering a larger population size will not give a better solution for GA-2. As population size increases in GA-2, slight or negligible improvement in the cost of the best solution. To report the result in this paper, a population size equal to 10 for GA-2 is selected.

Specification of the system used for running RLS, (1+1)-EA, GA-1, and GA-2 is Asus FS702D Ryzen 5 Quad Core AMD R5 2500 U (up to 3.6 GHz), 8 GB RAM, and HDD 1 TB. Figures [Fig. 1, Fig. 2, Fig. 3, Fig. 4, Fig. 5, and Fig. 6] and Tables [Table II, Table III, Table III, Table V, Table VI] shows comparison of RLS, (1+1)-EA, GA-1 and GA-2 for different values of p (0.04, 0.25, 0.5, 0.75, 0.9 and $\frac{\ln n}{n}$). Figures Fig. 1a, Fig. 2a, Fig. 3a, Fig. 4a, Fig. 5a, and Fig. 6a show a comparison of different algorithms concerning the cost of the solution obtained and the number of nodes in the graph. It is evident from the result that the cost obtained using the RLS algorithm is better than other algorithms considered in all the cases. (1+1)-EA is slightly better than GA if the cost of the solution obtained is considered. Figure Fig. 1b, Fig. 2b, Fig. 3b, Fig. 4b, Fig. 5b, and Fig. 6b show a comparison of different algorithms concerning CPU time needed to search $10n^2$ elements in the search space and number of nodes in the graph. It is clear from the figure that RLS is faster as compared to all other algorithms considered in all the cases (i.e., for different values of p).

From Tables [Table II, Table III, Table IV, Table V and Table VI] it is evident that taking one sample is not sufficient

TABLE I: Parameter Settings for GA

Version	Crossover Probability	Mutation Probability	Selection Method	Offspring per iteration	Number of iterations	Total individual Searched
GA-1	0.8	0.5	–	2	$5n^2$	$10n^2$
GA-2	0.8	0.5	Tournament	m	$\frac{10n^2}{m}$	$10n^2$
(1+1)-EA	–	–	–	1	$10n^2$	$10n^2$
RLS	–	–	–	1	$10n^2$	$10n^2$

TABLE II: Results of RLS, (1+1)-EA, GA-1 and GA-2 for p=0.04

No.of vertices	Optimum Cost	RLS		(1+1)-EA		GA-1		GA-2	
		Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time
10	100	100	0.02	84	0.8	81	0.06	92	2.44
20	400	400	0.44	304	6.73	356	4.86	376	6.95
25	625	625	1.4	457	14.5	481	12.16	497	9.99
50	2500	2500	9.55	2020	145.83	2152	197.54	2224	131.4
75	5625	5625	76.89	4729	639.041	4813	763.67	4921	672.04
100	10000	10000	276.02	8524	1825.6	8600	3110.9	8664	4055.8

TABLE III: Results of RLS, (1+1)-EA, GA-1 and GA-2 for p=0.25

No.of vertices	Optimum Cost	RLS		(1+1)-EA		GA-1		GA-2	
		Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time
10	100	96	0.24	84	0.41	72	0.75	80	2.06
20	400	344	3.13	300	3.76	312	8.24	312	11.35
25	625	509	7.16	465	18.25	481	18.9	465	21.34
50	2500	1940	105.67	1712	93.73	1696	156.78	1720	185.69
75	5625	4257	522.86	3745	460.66	3657	915.13	3793	1161.1
100	10000	7300	1635.16	6528	1300.14	6260	2758.78	6404	2922.44

TABLE IV: Results of RLS, (1+1)-EA, GA-1 and GA-2 for p=0.5

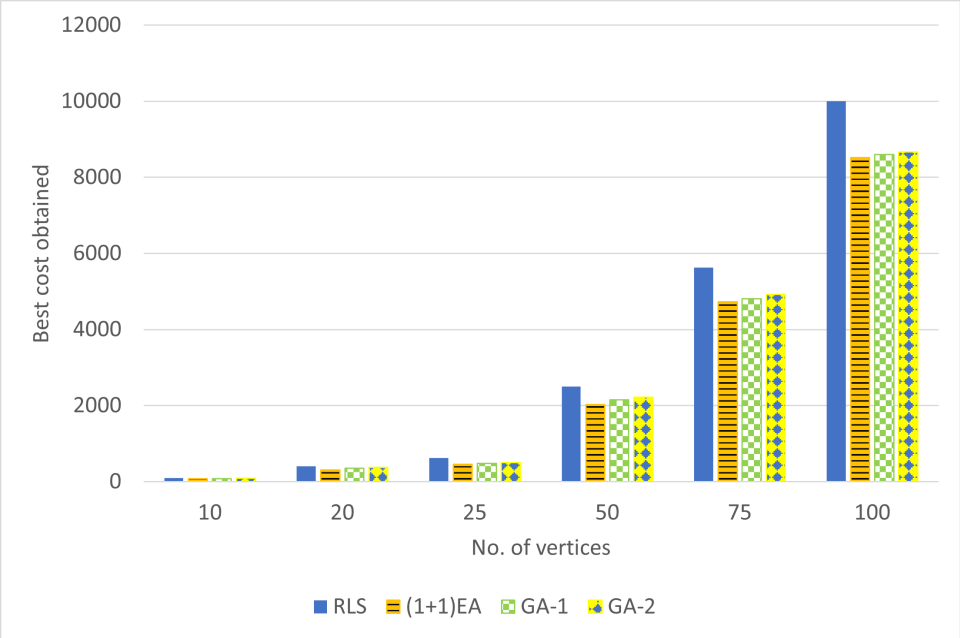
No.of vertices	Optimum Cost	RLS		(1+1)-EA		GA-1		GA-2	
		Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time
10	100	88	0.23	80	0.43	68	0.47	76	2.12
20	400	312	3.14	268	3.94	256	5.52	260	11.43
25	625	473	7.25	389	8.12	369	13.28	397	20.9
50	2500	1732	104.8	1436	94.52	1360	171.94	1440	305.96
75	5625	3713	330.98	3049	422.47	2913	965.42	3045	1273.26
100	10000	6344	1043.97	5635	1246.27	5372	2730.62	5536	3272.99

TABLE V: Results of RLS, (1+1)-EA, GA-1 and GA-2 for p=0.75

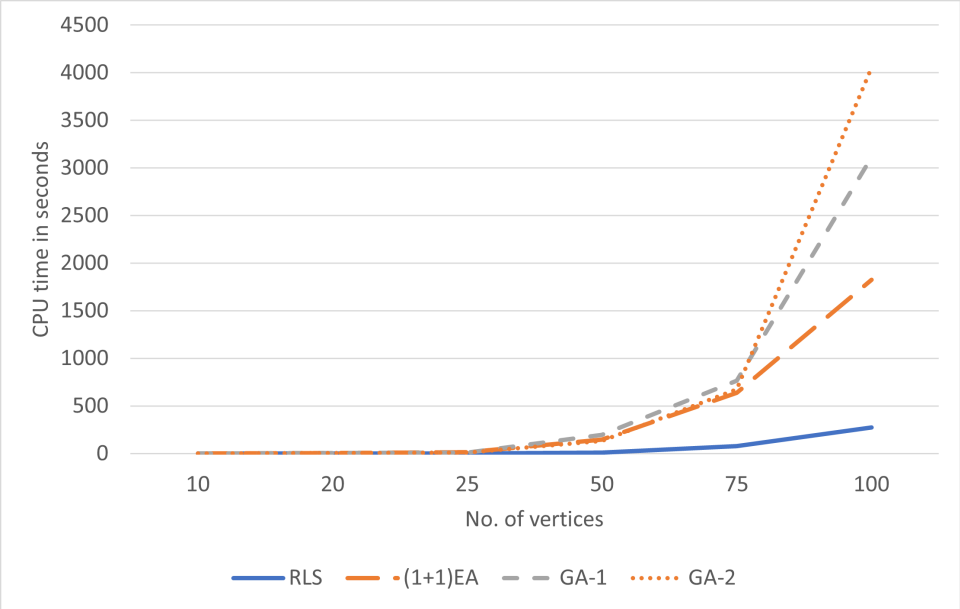
No.of vertices	Optimum Cost	RLS		(1+1)-EA		GA-1		GA-2	
		Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time
10	100	92	0.26	84	0.61	80	0.57	84	2.11
20	400	328	2.04	284	6.03	264	5.77	272	11.38
25	625	513	4.83	449	13.83	425	12.54	453	21.31
50	2500	1920	68.24	1708	94.52	1652	169.78	1720	176.99
75	5625	4265	346.48	3737	416.1	3653	949.41	3777	1418.42
100	10000	7316	1645.2	6519	1248.64	6516	2714.2	6668	4932.97

TABLE VI: Results of RLS, (1+1)-EA, GA-1 and GA-2 for p=0.9

No.of vertices	Optimum Cost	RLS		(1+1)-EA		GA-1		GA-2	
		Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time	Best Cost	CPU Time
10	100	96	0.2	92	0.66	88	0.68	96	2.06
20	400	360	3.05	344	6.23	328	8.19	332	11.46
25	625	565	7.17	525	13.34	517	17.57	521	21.0
50	2500	2228	107.56	2136	155.86	2112	263.84	2140	205.84
75	5625	4945	535.39	4637	679.96	4597	1236.08	4685	1555.3
100	10000	8808	1641.19	8372	1924.48	8372	3584.17	8416	6839.26



(a) No. nodes Vs Best Cost obtained



(b) No. nodes Vs CPU Time

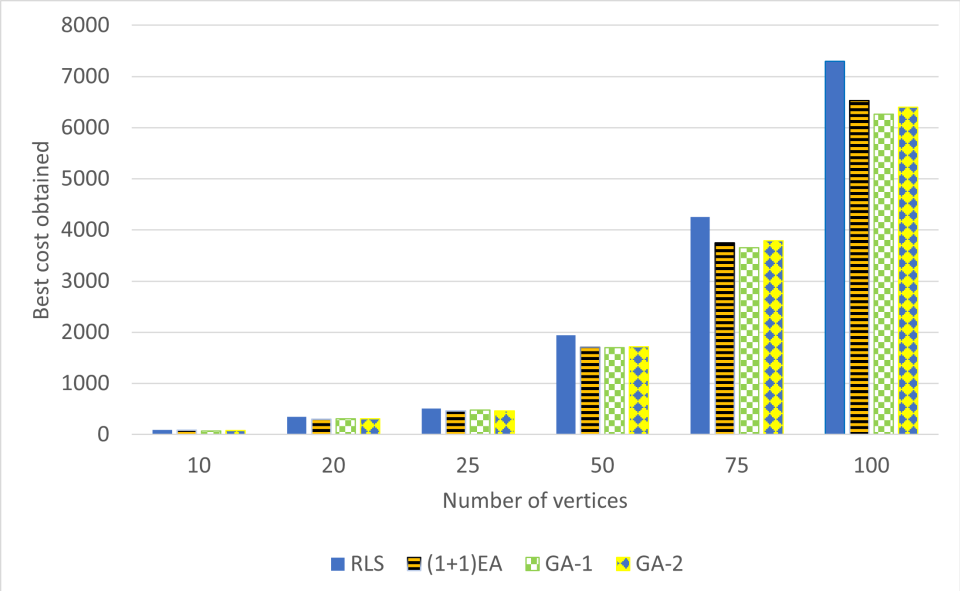
Fig. 1: Comparison of RLS, (1+1)EA, GA-1 and GA-2 for Erdös Rényi model $G(n, p = 0.04)$

to get a good solution when we make use of randomized search heuristics like RLS, EA and GA etc. A total of 100 samples are generated for each algorithm, where each sample is obtained by searching $10n^2$ individuals in the search space, and the cost of the best solution obtained is listed in Table VII. From Table VII, it is evident that RLS outperforms (1+1)-EA, GA-1, and GA-2 on permutation matrix space. (1+1)-EA, GA-1, and GA-2 failed to produce an optimum solution in all the samples obtained by searching $10n^2$ individuals, whereas RLS can produce an optimum solution in one of the 100 samples drawn by searching $10n^2$ individuals. For $n = 200$, getting one sample by running RLS for $10n^2$ iterations takes nearly half an hour of CPU time (1800 seconds). Therefore, taking 100 samples will approximately take 2 days. But even 100 samples are not sufficient to locate optimum solution when $n = 200$. The cost of the best solution obtained in this case is 39100 (optimum cost is 40000). As n increases, one has to take more samples to get the optimum solution for the problem at hand.

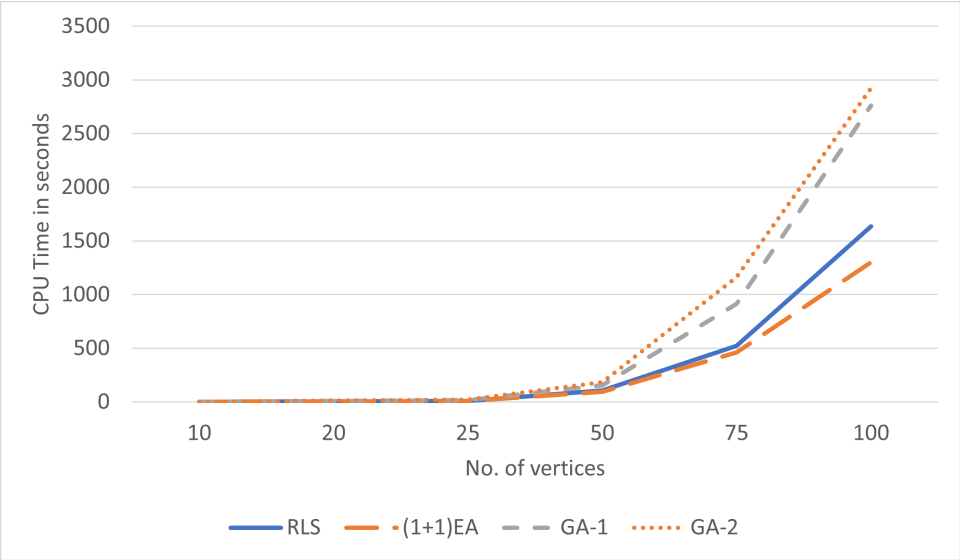
VI. Conclusion and Future Work

Conclusion

The paper proposes a new permutation matrix search space for GIP. The results obtained show that the magnification of permutation matrix space is at least $\frac{n}{2}$, where n denotes the number of search space elements. This work studies three meta-heuristic algorithms, RLS, (1+1)-EA, and GA for GIP using permutation matrix search space. The proposed work shows that the mixing time of the Markov chains associated with RLS has bounded above by $O(n^2)$. The experimental results show that the RLS algorithm outperforms (1+1)-EA



(a) No. nodes Vs Best Cost obtained



(b) No. nodes Vs CPU Time

Fig. 2: Comparison of RLS, (1+1)EA, GA-1 and GA-2 for Erdös Rényi model $G(n, p = 0.25)$

and GA on permutation matrix space. As a future work one can check the suitability of other meta-heuristics like ant colony optimization (ACO), Simulated Annealing (SA), Particle Swarm Optimization (PSO), etc., on permutation matrix space for GIP. It will be interesting to study the theoretical analysis of GA and other meta-heuristic algorithms for GIP using permutation matrix space. The proposed permutation matrix space can also be used by different meta-heuristic algorithms for subgraph isomorphism problems.

REFERENCES

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990.

[2] V. Arvind and P. P. Kurur, "Graph isomorphism is in spp," *Information and Computation*, vol. 204, no. 5, pp. 835 – 852, 2006.

[3] M. A. Abdulrahim and M. Misra, "A graph isomorphism algorithm for object recognition," *Pattern Analysis and Applications*, vol. 1, no. 3, pp. 189–201, Sep 1998.

[4] H. Bunke and B. T. Messmer, "Efficient attributed graph matching and its application to image analysis," in *Image Analysis and Processing*, C. Braccini, L. DeFloriani, and G. Vernazza, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 44–55.

[5] S. Pan, X. Zhu, C. Zhang, and P. S. Yu, "Graph stream classification using labeled and unlabeled graphs," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, April 2013, pp. 398–409.

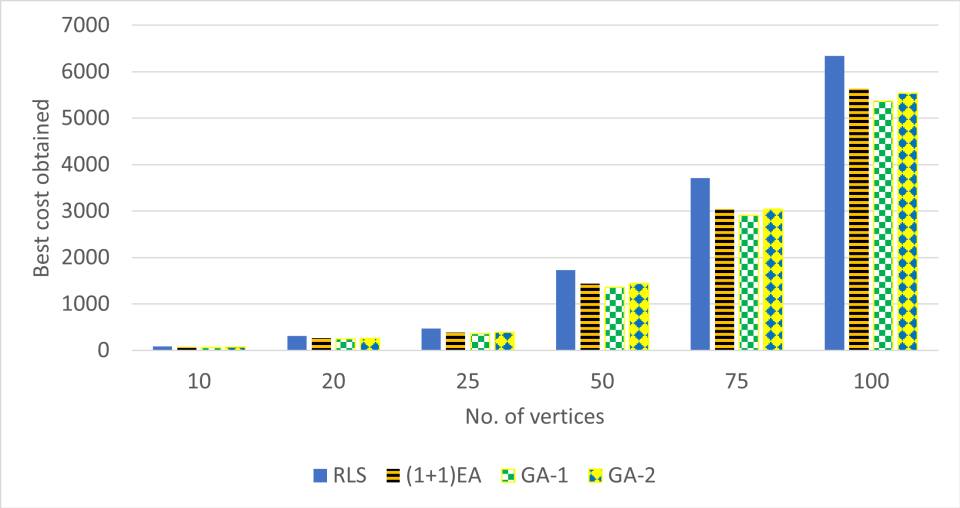
[6] A. Elmsallati, C. Clark, and J. Kalita, "Global alignment of protein-protein interaction networks: A survey," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 13, no. 4, pp. 689–705, July 2016.

[7] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," *Proceedings of the National Academy of Sciences*, vol. 105, no. 35, pp. 12 763–12 768, 2008.

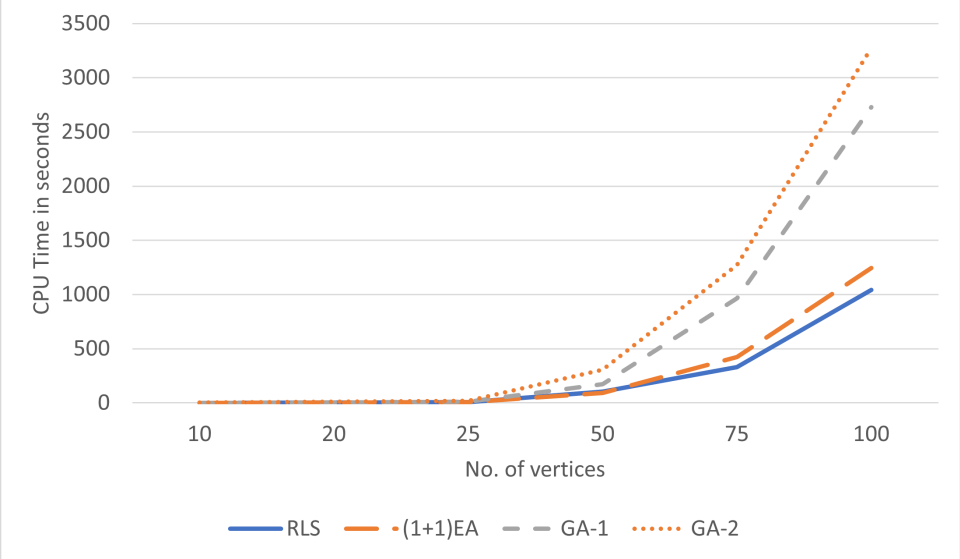
[8] C. Henneges, C. Behle, and A. Zell, "Practical graph isomorphism for graphlet data mining in protein structures," in *Computational Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 345–360.

[9] J. Huan, D. Bandyopadhyay, W. Wang, J. Snoeyink, J. Prins, and A. Tropsha, "Comparing graph representations of protein structure for mining family-specific residue-based packing motifs," *Journal of Computational Biology*, vol. 12, no. 6, pp. 657–671, 2005.

[10] D. Isenor and S. Zaky, "Fingerprint identification using graph matching," *Pattern Recognition*, vol. 19, no. 2, pp. 113 – 122, 1986.



(a) No. nodes Vs Best Cost obtained



(b) No. nodes Vs CPU Time

Fig. 3: Comparison of RLS, (1+1)EA, GA-1 and GA-2 for Erdös Rényi model $G(n, p = 0.5)$

TABLE VII: Results of RLS, (1+1)-EA, GA-1 and GA-2 using permutation matrix space

No.of vertices	Optimum Cost	RLS Best cost Obtained	(1+1)-EA Best cost Obtained	GA-1 Best cost Obtained	GA-2 (population size = 10) Best cost Obtained
10	100	100	80	81	92
20	400	400	304	356	376
25	625	625	457	481	497
50	2500	2500	2020	2152	2224
75	5625	5625	4729	4813	4921
100	10000	10000	8524	8600	8664

[11] D. R. Kisku, P. Gupta, and J. K. Sing, "Feature level fusion of face and palmprint biometrics by isomorphic graph-based improved k-medoids partitioning," in *Advances in Computer Science and Information Technology*, T.-h. Kim and H. Adeli, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 70–81.

[12] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *IJPRAI*, vol. 18, no. 3, pp. 265–298, 2004.

[13] K. Madi, "Inexact graph matching : application to 2D and 3D Pattern Recognition," Theses, Université de Lyon, Dec. 2016.

[14] L. Babai, "Graph isomorphism in quasipolynomial time [extended abstract]," in *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 684–697.

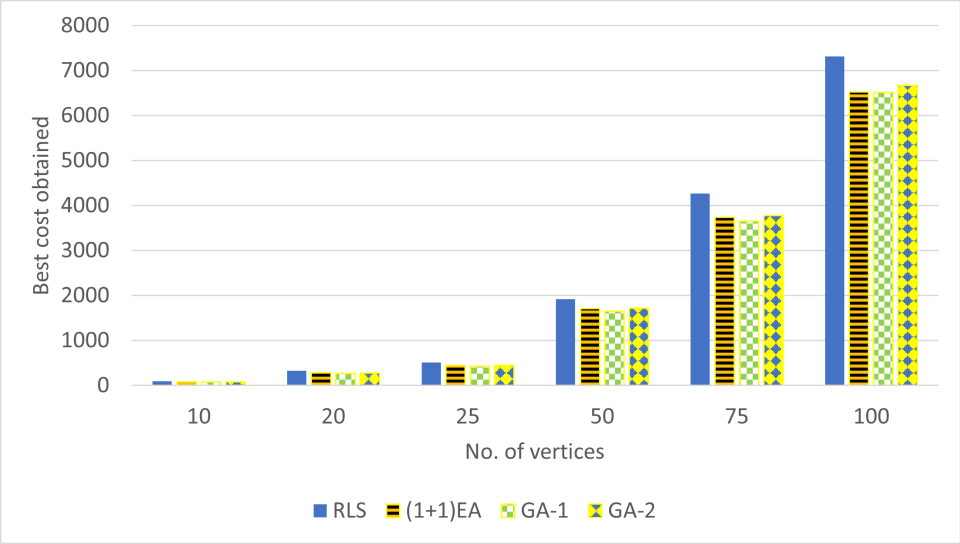
[15] L Babai, "Graph isomorphism algorithm updated, quasipolynomial-time claim restored," http://people.cs.uchicago.edu/~laci/update.html, 2017.

[16] D. Wiebking, "Graph Isomorphism in Quasipolynomial Time Parameterized by Treewidth," in *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), A. Czumaj, A. Dawar, and E. Merelli, Eds., vol. 168. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 103:1–103:16.

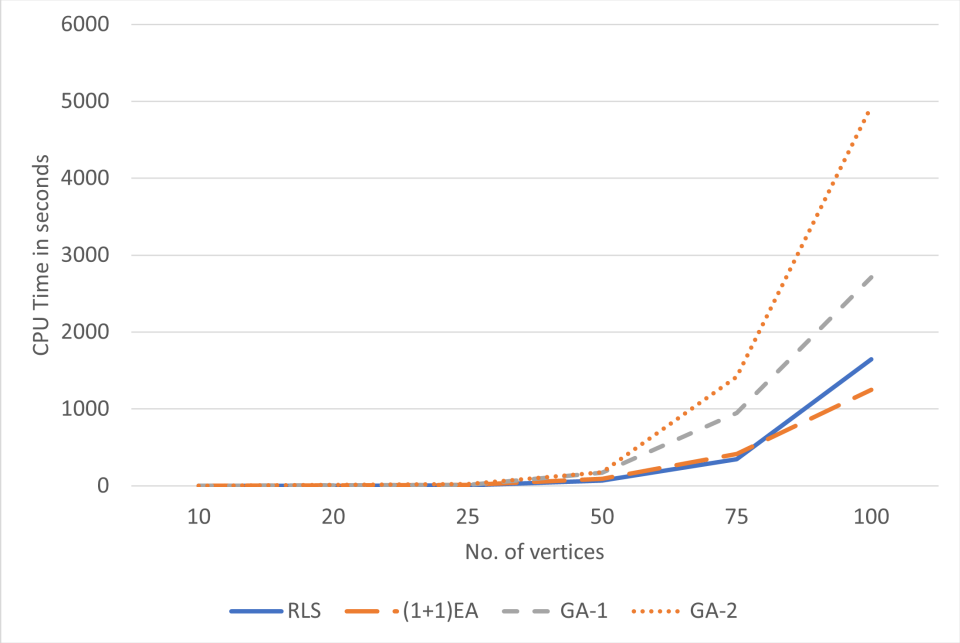
[17] K. Liu, Y. Zhang, K. Lu, X. Wang, X. Wang, and G. Tian, "Mapeff: An effective graph isomorphism agorithm based on the discrete-time quantum walk," *Entropy*, vol. 21, no. 6, p. 569, Jun 2019.

[18] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+ 1) evolutionary algorithm," *Theor. Comput. Sci.*, vol. 276, no. 1-2, pp. 51–81, apr 2002.

[19] P. Jain, L. Kanesh, J. Madathil, and S. Saurabh, "A parameterized



(a) No. nodes Vs Best Cost obtained



(b) No. nodes Vs CPU Time

Fig. 4: Comparison of RLS, (1+1)EA, GA-1 and GA-2 for Erdös Rényi model $G(n, p = 0.75)$

runtime analysis of randomized local search and evolutionary algorithm for max uncut," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, H. E. Aguirre and K. Takadama, Eds. ACM, 2018, pp. 326–327.

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

[21] J. McCall, "Genetic algorithms for modelling and optimisation," *Journal of Computational and Applied Mathematics*, vol. 184, no. 1, pp. 205 – 222, 2005, special Issue on Mathematics Applied to Immunology.

[22] J. A. Vasconcelos, J. A. Ramirez, R. H. C. Takahashi, and R. R. Saldanha, "Improvements in genetic algorithms," *IEEE Transactions on Magnetics*, vol. 37, no. 5, pp. 3414–3417, 2001.

[23] C. García-Martínez, F. J. Rodriguez, and M. Lozano, *Genetic Algorithms*. Cham: Springer International Publishing, 2018, pp. 431–464.

[24] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[25] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Scituate, MA, USA: Bradford Company, 2004.

[26] X. Zhang and D. Yuan, "A niche ant colony algorithm for parameter identification of space fractional order diffusion equation," *IAENG International Journal of Applied Mathematics*, vol. 47, no. 2, pp. 197–208, 2017.

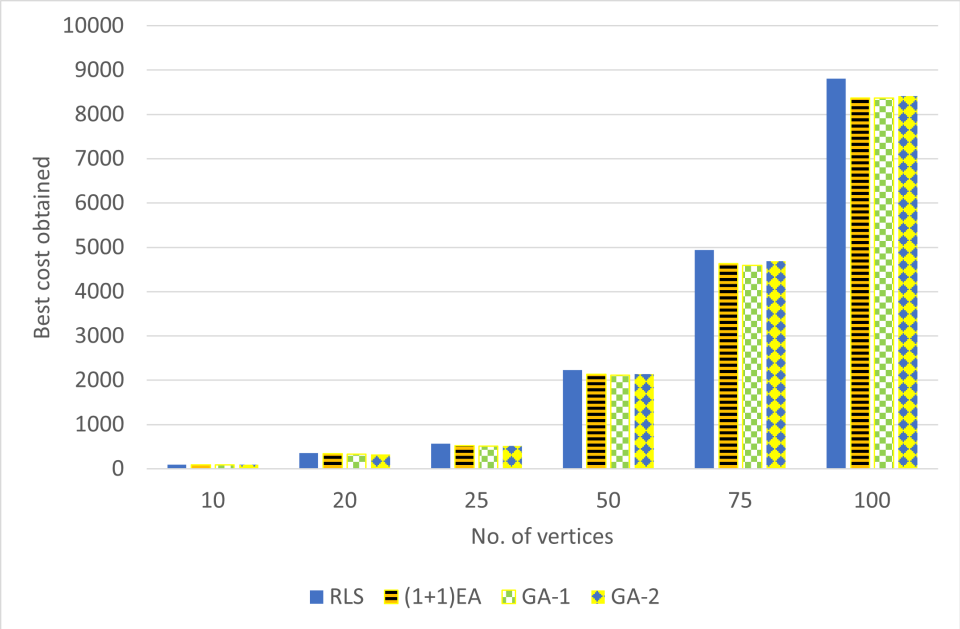
[27] R. M. Son Duy Dao, Kazem Abhary, "An adaptive restarting genetic algorithm for global optimization," in *Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering and Computer Science 2015*, San Francisco, USA, 21-23 October 2015, pp. 455–459.

[28] M. A. R. M. A. H. Akhand, Shahina Akter and S. B. Yaakob, "Velocity tentative pso: An optimal velocity implementation based particle swarm optimization to solve traveling salesman problem," *IAENG International Journal of Computer Science*, vol. 42, no. 3, pp. 221–232, 2015.

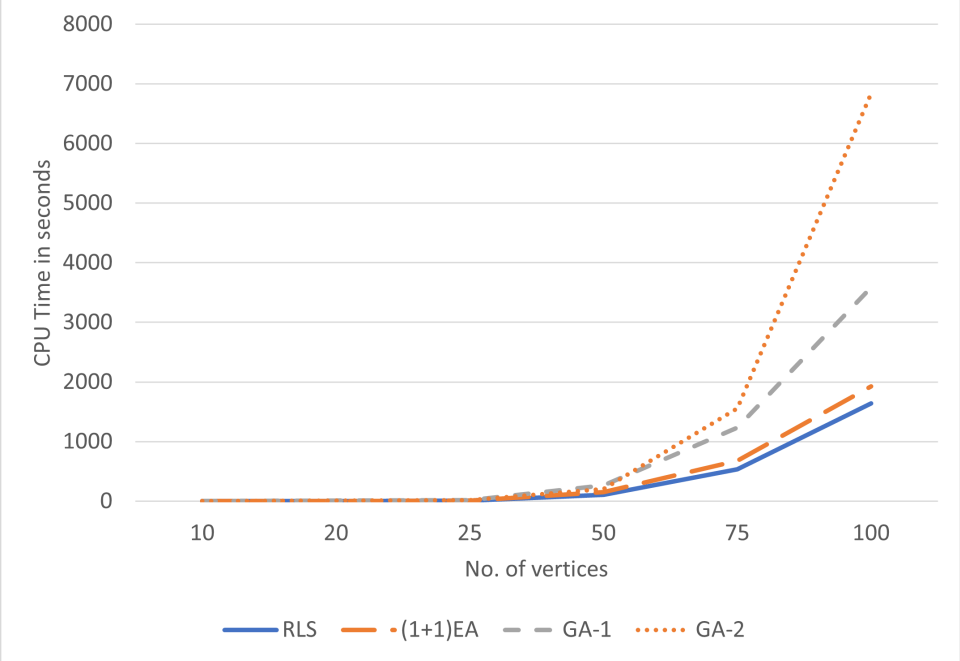
[29] D. X. Lei Jiang, "An efficient differential memetic algorithm for clustering problem," *IAENG International Journal of Computer Science*, vol. 45, no. 1, pp. 118–129, 2018.

[30] Y. D. Shiyong Li, Huan Liu, "An optimal resource allocation scheme for elastic applications in multipath networks via particle swarm optimization," *IAENG International Journal of Computer Science*, vol. 47, no. 2, pp. 278–283, 2020.

[31] Z. Y. Xin Shen, Xiaoxia Zhang, "A new hybrid cuckoo search for the resourcesesources-constrained project scheduling problem," *IAENG International Journal of Computer Science*, vol. 48, no. 2, pp. 304–



(a) No. nodes Vs Best Cost obtained



(b) No. nodes Vs CPU Time

Fig. 5: Comparison of RLS, (1+1)EA, GA-1 and GA-2 for Erdös Rényi model $G(n, p = 0.9)$

315, 2021.

[32] S.-S. G. Yi-Xuan Lu, Jie-Sheng Wang, "Solving path planning problem based on particle swarm optimization algorithm with improved inertia weights," *IAENG International Journal of Computer Science*, vol. 46, no. 4, pp. 628–636, 2019.

[33] Y. H. D. K. Wang and Q. Zhao, "A hybrid metaheuristic algorithm for the heterogeneous school bus routing problem and a real case study," *IAENG International Journal of Computer Science*, vol. 47, no. 4, pp. 775–785, 2020.

[34] S. A. Rezzy Eko, Rung Ching, "Employing best input svr robust lost function with nature-inspired metaheuristics in wind speed energy forecasting," *IAENG International Journal of Computer Science*, vol. 47, no. 3, pp. 572–584, 2020.

[35] X.-L. S. Hua-Ping Wu, Hui Li, "Evolutionary game for enterprise cloud accounting resource sharing behavior based on the cloud sharing platform," *IAENG International Journal of Applied Mathematics*, vol. 51, no. 1, pp. 125–132,, 2021.

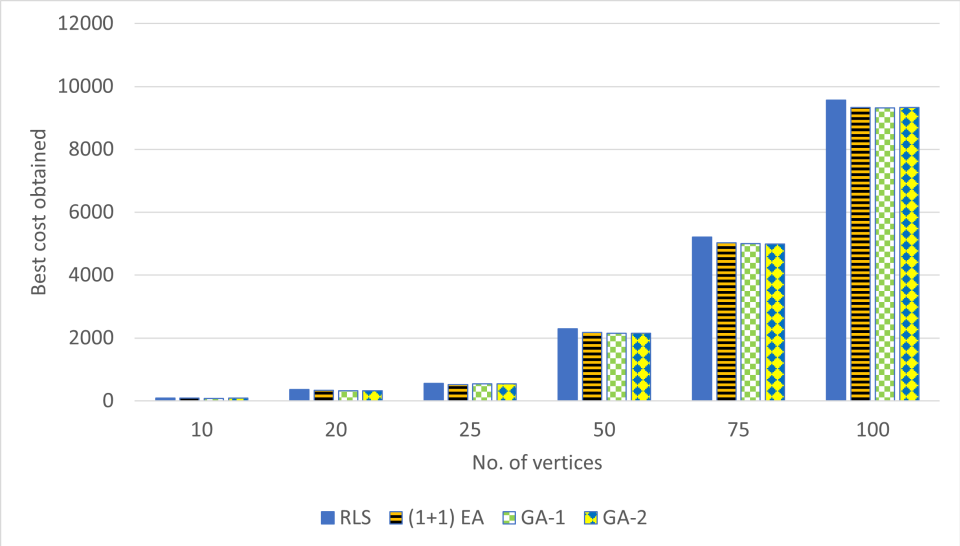
[36] A. Raghavan, P. Maan, and A. K. B. Shenoy, "Optimization of day-ahead energy storage system scheduling in microgrid using genetic algorithm and particle swarm optimization," *IEEE Access*, vol. 8, pp. 173 068–173 078, 2020.

[37] S. K. B. Ajitha, S. Biswas, and P. P. Kurur, "Metropolis algorithm for solving shortest lattice vector problem (svp)," in *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, 2011, pp. 442–447.

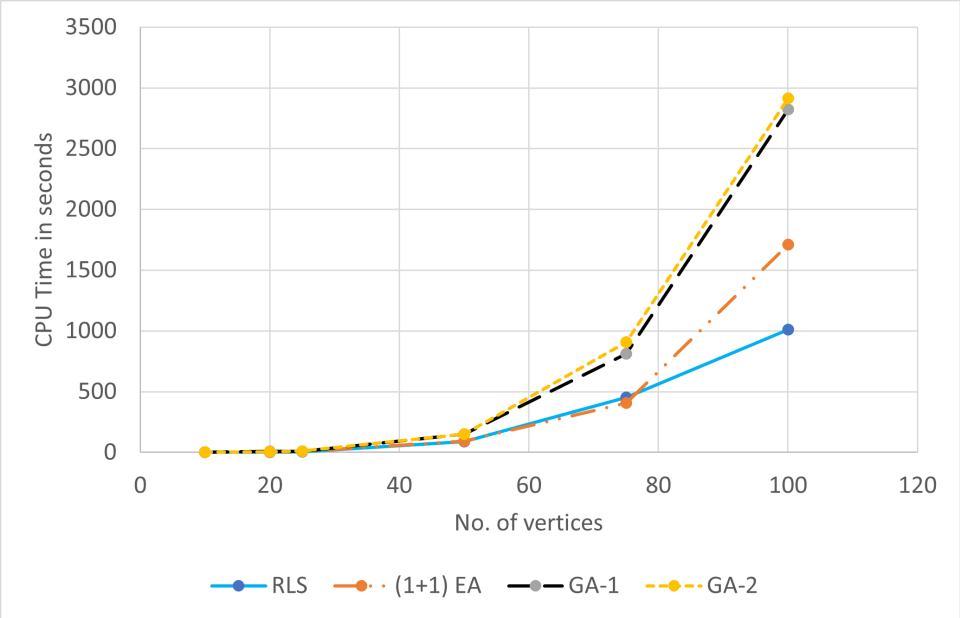
[38] A. Shenoy K B, S. Biswas, and P. P. Kurur, "Performance of metropolis algorithm for the minimum weight code word problem," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 485–492. [Online]. Available: https://doi.org/10.1145/2576768.2598274

[39] Y. X. Z. Nan Wang, Jie Sheng Wang, "Two dimensional bin packing problem with rectangular and circular regions solved by genetic algorithm," *IAENG International Journal of Applied Mathematics*, vol. 51, no. 2, pp. 268–278, 2021.

[40] J. H. Rayner Alfred, Loo Yew, "Social media mining: A genetic based multiobjective clustering approach to topic modelling," *IAENG International Journal of Computer Science*, vol. 48, no. 1, pp. 32–42,



(a) No. nodes Vs Best Cost obtained



(b) No. nodes Vs CPU Time

Fig. 6: Comparison of RLS, (1+1)EA, GA-1 and GA-2 for Erdös Rényi model $G(n, p = \ln n/n)$

2021.

[41] M. R. H. Abdallah Aissou, Abdelhamid Daamouche, "Components assignment problem for flow networks using mopso," *IAENG International Journal of Computer Science*, vol. 48, no. 1, pp. 96–108, 2021.

[42] C. Z. Gonggui Chen, Ao Zhang, "Optimal placement and capacity of combined dgs and scs in radial distribution networks based on pso-os algorithm," *IAENG International Journal of Computer Science*, vol. 48, no. 2, pp. 236–249, 2021.

[43] Y. H. Xiaoxia Zhang, Ziqiao Yu, "Milling force prediction of titanium alloy based on support vector machine and ant colony optimization," *IAENG International Journal of Computer Science*, vol. 48, no. 2, pp. 223–235, 2021.

[44] J. Choi, Y. Yoon, and B.-R. Moon, "An efficient genetic algorithm for subgraph isomorphism," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 361–368. [Online]. Available: https://doi.org/10.1145/2330163.2330216

[45] H. Choi, J. Kim, Y. Yoon, and B.-R. Moon, "Investigation of incremental hybrid genetic algorithm with subgraph isomorphism problem," *Swarm and Evolutionary Computation*, vol. 49, pp. 75–86, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650218308071

[46] L. Gonz´alez, "Edges, chains, shadows, neighbors and subgraphs in the intrinsic order graph," *IAENG International Journal of Applied Mathematics*, vol. 42, no. 1, pp. 66–73, 2012.

[47] S. Wang Jing-yu, "Some results of the compact graph," *IAENG International Journal of Applied Mathematics*, vol. 49, no. 4, pp. 588–594, 2019.

[48] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*. Cambridge University Press, 1991.

[49] S. Pissanetzky, ""linear algebraic equations"," in *Sparse Matrix Technology*, S. Pissanetzky, Ed. Academic Press, 1984, pp. 38 – 68.

[50] L. Chen, "Graph isomorphism and identification matrices: Sequential algorithms," *Journal of Computer and System Sciences*, vol. 59, no. 3, pp. 450 – 475, 1999.

[51] A. Sinclair, *Algorithms for Random Generation and Counting A Markov Chain Approach*. Cambridge, MA 02139, USA.: Birkhauser Boston, 1993.

[52] A. K. B. Shenoy, S. Biswas, and P. P. Kurur, "Efficacy of the metropolis algorithm for the minimum weight codeword problem using codeword and generator search spaces," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 664 – 678, 2020.

[53] J. R. Norris, *Markov chains.*, ser. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998.

[54] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. New York, USA:

Cambridge University Press, 2005.
[55] D. Levin, Y. Peres, and E. Wilmer, *Markov Chains and Mixing Times*. American Mathematical Soc., 2008.
[56] P. Erdös and A. Rényi, "On random graphs," *Publicationes Mathematicae Debrecen*, vol. 6, p. 290, 1959.