Finding Maximum Independent Set using Particle Swarm Optimization

Ritika Verma
Department of Computer Science and Engineering
National Institute of Technology Hamirpur
Hamirpur, Himachal Pradesh-177 005, India
24rcs001@nith.ac.in

Dharmendra Prasad Mahato
Department of Computer Science and Engineering
National Institute of Technology Hamirpur
Hamirpur, Himachal Pradesh-177 005, India
dpm@nith.ac.in

***Abstract*—The MaxIS (Maximum Independent Set) belongs to the category of NP-hard problems. To address this issue, we apply the PSO (particle swarm optimization) technique in this study. We evaluate the suggested method against other soft computing methods currently in use. The experimental testing is conducted using DIMACS10 benchmarks. The results demonstrate that the suggested method performs well and surpasses numerous other soft computing techniques, such as genetic algorithm (GA) and grey wolf optimization (GWO).**

***Index Terms*—Maximum Independent Sets (MaxIS) problem, Maximal Independent Set (MIS), Particle Swarm Optimization, Soft Computing methods.**

I. Introduction

The Maximal Independent Set (MIS) problem and the Maximum Independent Set (MaxIS) problem are core combinatorial optimization problems with applications in areas such as bioinformatics, scheduling, and network design. Both MIS and MaxIS are central to distributed computing and have been the focus of extensive research over the past four decades. Some important contributions to the distributed message forwarding paradigm are [5], [18], [20], [22]. The computational complexity of the problem frequently proves problematic for traditional exact algorithms, especially for the large graphs. Since the MaxIS problem is NP-hard. For solving this type of problem, we can apply soft computing (heuristic or meta-heuristic) methods to solve the problem.

In relation to the MaxIS problem, various algorithms have been created in recent years by tackling this issue using various heuristic techniques. A genetic method for the MaxIS problem was created by [6] which employed a graded penalty function to a few tiny MaxIS problem cases. Aggarawal *et al.*'s recent work [2] compares state-of-the-art approaches to a genetic algorithm for the MaxIS problem.

In this paper, we apply a soft computing approach such as the PSO to address the MaxIS problem. In the area of soft computing, Kennedy and Eberhart first proposed particle swarm optimization in 1995 [15]. This heuristic global optimization technique was inspired by swarm intelligence and is based on studies of the flock movement patterns of fish and birds, which are either dispersed or converge when looking for food. As the birds hunt and search, there is always one particular bird that has superior information about food resources and can smell the food extremely well, allowing it to be detected at the location of the food. As a result of their constant dissemination of information, particularly positive information, the birds will eventually congregate at the location of food because they will finally be searching for it everywhere.

In the Particle Swarm Optimization algorithm, the solution swarm resembles a flock of birds. The movement of the birds mirrors the swarm's evolution, the best information represents the optimal solution, and the food source signifies the best solution identified throughout the process. The food resource is comparable to the most optimal solution found throughout the entire process. By working together, each member of the particle swarm optimization algorithm can determine the most optimal solution [8].

For evaluation and testing the algorithms, we need to test them on some standard benchmarking datasets. The DIMACS and DIMACS10 [24] challenges provided the instances that were tested in this work [7]. In Resende *et al.* [13], GRASP is used on a collection of more complex and large-scale MaxIS problem instances that are produced using Bollobas's [10] technique for creating random graphs that can be predicted in advance.

A. Motivation

A maximal independent set might represent finding the largest set of non-interfering channels or nodes. Using PSO can provide an efficient way to determine these sets, thus optimizing network performance and resource allocation. Using particle swarm optimization (PSO), researchers and practitioners can efficiently address both the MIS and MaxIS problems, achieving near-optimal solutions within a practical timeframe. This approach is particularly valuable for applications in domains such as telecommunications, logistics, and social network analysis.

- **Heuristic Approach for Complex Problems:** The MIS as well as MaxIS problems are NP-hard, making them computationally challenging to solve exactly, especially for large graphs. PSO provides a heuristic approach that can find good solutions in reasonable time frames.

- **Exploration and Exploitation Balance:** PSO's ability to balance exploitation (refining existing solutions) and exploration (investigating new areas of the search space) makes it well-suited for navigating the complex search space of the Maximum Independent Set problem.
- **Adaptability:** PSO can adapt to different types of graphs (e.g., sparse, dense) and problem constraints without significant changes to the algorithm structure.
- **Parallelism and Scalability:** The naturally parallel structure of PSO enables its implementation in parallel computing environments, enhancing its scalability for solving large problem instances.
- **Simplicity and Flexibility:** PSO is relatively simple to implement and can be easily combined with other optimization techniques or domain-specific heuristics to enhance performance.

B. Our Contributions

1) In this paper, we use the PSO algorithm to solve the MaxIS problem.
2) We have simulated other combinatorial algorithms for comparison, e.g., the grey wolf optimizer and the genetic algorithm.
3) We have used DIMACS10 benchmark datasets for testing and evaluation of the experimental results.

C. Organization of paper

Section II presents the literature survey on MaxIS and soft computing approaches. In Section III, we have the problem formulation. Section IV outlines the methodology for addressing the problem, while Section V presents the analysis of the results. Finally, in Section VI, the paper concludes and also discusses the future directions in this field of research.

II. Literature Survey

The development of heuristic-based approaches stems from the significant computing complexity involved in determining the maximal independent set, which rises with the size of the graph. The accuracy and degree of graph complexity of heuristic algorithms allow them to find the suboptimal solutions in polynomial time.

Blelloch *et al.* proposed the greedy sequential approach to solve the maximal independent set problem, which shows polylogarithmic bounds for random graphs [9]. Krivelevich *et al.* presented a general framework for computing the asymptotic density of the random greedy independent set for sequences of possibly random graphs by employing a notion of local convergence [16]. Das *et al.* presented a critical review of different existing approaches in evolutionary methods to solve the maximum independent set problem [12].] Almara and Suleiman applied the Min_Max algorithm to solve the maximum independent set problem [4]. Adil *et al.* proposed a new approach to solve the problem to find the maximum independent set in a given Graph, known also as max-stable set problem (MSSP) [1].

M. Hia proposed a genetic algorithm-based heuristic, especially for the weighted maximum independent set problem [14]. Sakamato *et al.* presented a genetic algorithm for maximum independent set problem and permutation encoding with a greedy decoding [25]. Aggarwal *et al.* discussed a classical combinatorial problem called the independent set problem [2]. Nayeem and Pal presented the genetic algorithm (GA) to find the Maximum Weight Independent Set (MWIS) of a graph [21]. Moisés Silva *et al.* presented a new artificially generated algorithm for the maximum independent set problem [27]. The automatic production of algorithms, a method that enables the creation of new hybrid algorithms by utilizing pre-existing algorithms, produces the new algorithm. Taranenko and Vesel presented a new genetic algorithm for the maximum independent set problem based on the elitist strategy [28].

The ant colony optimization technique has also been applied for solving the maximum independent set problem [17], [29].

The particle swarm optimization approach has also been used to solve the MaxIS problem [3], [26].

III. Problem Formulation

Many similar formulations of the maximum independent set issue exist, including those as a continuous non-convex optimization problem and an integer programming problem [11], [23].

Let $G = (V, E)$, where $|V| = n$, is an example of an undirected graph. A stable set (also known as an independent set) of the graph G is a subset of the nodes in which there are no edges connecting any two of them. Let us look at $\alpha(G)$ which represents the size of a graph G's biggest stable set and is also known as the graph's stability number. Determining whether $\alpha(G)$ exceeds a given integer k is an NP-hard issue.

According to [19], one natural integer programming formulation of MaxIS is the following:

$$\begin{aligned} &\max \sum_{i=1}^{n} x_i \\ &\qquad s.t. x_i + x_j \leq 1, if(i,j) \in E, \\ &\qquad x_i \in \{0,1\}, i = 1, \ldots, n. \end{aligned} \tag{1}$$

IV. Methodology

This section presents the proposed algorithms for the MaxIS problem.

PSO is an evolutionary method that draws inspiration from natural animal behavior found in the wild. In literature, Kennedy and Eberhart were the ones who first suggested it (1995) [15]. PSO was initially developed to solve continuous space optimization problems, although it is also helpful for issues with combinatorial optimization. Particles, or potential solutions, make up a swarm, traveling through the search area.

There are three possible movements for each particle: traveling in its own direction, returning to its peak local position from before P_i^{Best} personal best position based on an assessment function known as fitness. The best (i.e., maximum fitness) position a particle has attained thus far during the algorithm is its personal best P_i^{Best}. In addition to the global best G_i^{Best}, which is the highest position any particle in the entire swarm has ever encountered, each particle maintains track of its own personal best position and direction. The best (i.e., maximum fitness) position discovered by any particle across the swarm is the global best G_i^{Best}. The global best G_i^{best} is the best (i.e., highest fitness) position that has been found by any particle in the entire swarm. It is a global reference point that all particles use to guide their movement. The throng investigates the search space using the formulas (Eq. 2 and Eq. 3).

$$V_i^{t+1} = W.V_i^t + c_i.r_i.(P_i^{Best} - X_i^t) + c_2.r_2.(G_i^{Best} - X_i^t) \quad (2)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (3)$$

The particle position at an instant t is represented by X_i^t. The updated velocity of the particle at time $t+1$. The current velocity of particle V_i^t at time t The distance that this particle must travel from X_i^t is represented by V_i^t, which is called the velocity of particle i at instant t. In $[0,1]$, the variables r_1 and r_2 are created at random. Inertia, social factor, and cognitive factor are the names given to the parameters c_1, c_2, and W, respectively [15].

A. Steps of PSO Algorithm

- **Initial Swarm Generation:** The PSO method begins by generating a swarm of particles at random. Each particle is represented as a node in the graph, and the number of maximal independent sets it includes indicates how well it fits the data. This process begins with adding a randomly selected vertex to an empty current Maximal Independent Set.
- **Particle Position Update:** In this phase, the MaxIs is found by calculating the maximum number of elements among all the fit Maximal Independent Sets. In each iteration, we find the maximum independent set (MaxIS) as G^{Best}.
-

B. Pseudocode of PSO

This section presents the pseudocode of PSO (as shown in **Algorithm 1**) for finding the MaxIS.

Algorithm 1: Particle swarm optimization for Maximum Independent Set

```
1  Input: Graph G, swarm size S, number of iterations I
2  Output: Maximum Independent Set $S_{max}$
3  int main()
4  G ← ReadGraph(filePath) $S_{max}$ ← {}
     $swarm$ ← Initialize Swarm(G, S)
5  for $iter$ ← 1 to I do
6      for each particle in swarm do
7          UpdateParticle(particle)
8          LocalSearch(particle.Position)
             $particle.Fitness$ ←
             EvaluateFitness($particle.Position$)
             Update personal and global bests
9      end
10     Print current best fitness and independent set
11 end
12 return global best position as $S_{max}$
13 InitializeSwarm(G, S)
14 for i ← 1 to S do
15     Initialize particle position and velocity randomly
         Evaluate fitness
         Update personal and global bests
16 end
17 Return swarm
18 UpdateParticleparticle
19 for each dimension d do
20     Update velocity
21     Update position based on velocity and sigmoid function
22 end
23 EvaluateFitnessposition
24 Calculate size and validity of the independent set
   return fitness value (size if valid, else 0)
25 LocalSearchposition
26 Initialize empty independent set and inclusion array
     for each vertex v in position do
27     if vertex v can be included then
28         Add v to the independent set and mark as included
29     end
30 end
31 Update position to reflect the refined independent set
```

C. How does the proposed algorithm work?

Let us consider a graph (as shown in Fig. 1) to address the MaxIS problem. This section outlines the step-by-step process of our proposed algorithm using the example of a network represented as a graph in Fig. 1.

In **step 1: Graph initialization**, we begin by defining the number of vertices (V) and the number of edges (E). In the given MaxIS graph (as shown in Fig. 1), the set of vertices is $|V| = 6$, with the vertices labeled as $\{1,2,3,4,5,6\}$.

In this graph, the edge connecting vertex 1 to vertex 2 can be represented as $\{1-2\}$. Similarly, the other edges are

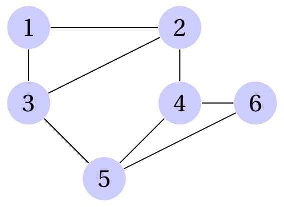

Fig. 1: **Example of MaxIS**

represented as follows: {1 − 3}, {2 − 4}, {3 − 5}, {4 − 5}, {4 − 6}, and {5 − 6}.

In **step 2: Swarm Initialization** we initialize the particle randomly. Each particle's position is a binary array representing 1 represents a node is included, while 0 represents that the node is not included.

For Particle 1: Position is {0, 1, 0, 1, 0, 0} and Velocity is {0.2, -0.5, 0.3, -0.1, 0.4, 0.6}.

For Particle 2: Position is {1, 0, 1, 0, 0, 1} and Velocity is {-0.4, 0.5, -0.2, 0.1, -0.3, 0.7}.

For Particle 3: Position is {0, 0, 1, 0, 1, 0} and Velocity is {0.1, -0.4, 0.2, 0.5, -0.2, 0.3}.

For Particle 4: Position is {1, 1, 0, 1, 0, 0} and Velocity is {-0.1, 0.3, -0.5, 0.2, -0.4, 0.6}.

For Particle 5: Position is {0, 1, 0, 0, 1, 1} and Velocity is {0.3, -0.6, 0.2, 0.4, -0.3, 0.5}.

In **step 3:** The particle's velocity is determined using this formula. According to Eq. 2, let us assume, the value of W is 0.5, c_1 is 1.5, c_2 is 1.5, r_1 is 0.8 and r_2 is 0.6. Then, we calculate the updated velocity for each dimension:

- **For Dimension 1:**

  $V_1^{(t+1)} = 0.5 * 0.2 + 1.5 * 0.8(0 - 0) + 1.5 * 0.6(1 - 0) = 1.0.$ So the updated velocity for the first dimension of particle 1 is, 0.1 and we will use the same dimension for each dimension.
- **For Dimension 2:** $V_2^{(t+1)} = 0.5 * (-0.5) + 1.5 * 0.6(1 - 1) + 1.5 * 0.4(1 - 1) = -0.25.$
- **For Dimension 3:** $V_3^{(t+1)} = 0.5 * 0.3 + 1.5 * 0.9(1 - 0) + 1.5 * 0.7(0 - 0) = 1.5.$
- **For Dimension 4:** $V_4^{(t+1)} = 0.5 * (0.1) + 1.5 * 0.4(0 - 1) + 1.5 * 0.6(1 - 1) = -0.65.$
- **For Dimension 5:** $V_5^{(t+1)} = 0.5 * 0.4 + 1.5 * 0.7(1 - 0) + 1.5 * 0.3(1 - 0) = 1.7.$
- **For Dimension 6:** $V_6^{(t+1)} = 0.5 * 0.6 + 1.5 * 0.3(0 - 0) + 1.5 * 0.3(0 - 0) = 0.3.$ So the updated velocity is {1.0, -0.25, 1.5, -0.65, 1.7, 0.3}.

In **step 4**, we use the sigmoid function to determine the new position from the updated velocity. Now the updated velocity is {1.0, -0.25, 1.5, -0.65, 1.7, 0.3} and the current position is {0, 1, 0, 1, 0, 0}. We need to update this based on the sigmoid function applied to updated velocity.

$$\text{Sigmoid function } f(v) = \frac{1}{1 + e^{v}} \tag{4}$$

The position becomes 1 if the sigmoid value exceeds a random threshold; otherwise, it becomes 0. We use a random threshold value which is used for all dimensions. Let us suppose the threshold value is 0.5.

- **For Dimension 1:** Velocity is {1.0}, Sigmoid function $f(1.0) = \frac{1}{1+e^{1.0}} \approx 0.731$ and threshold value is 0.5. Here, $(0.731 > 0.5)$, So the new position is 1.
- **For Dimension 2:** Velocity is -0.25, then $f(-0.25) \approx 0.44$ and threshold is 0.5. Here, since $\{0.44 < 0.5\}$. So, the new position is 0.
- **For Dimension 3:** Velocity is 1.5, then $f(1.5) \approx 0.818$ and threshold is 0.5. Here, since $\{0.818 > 0.5\}$. So, the new position is 1.
- **For Dimension 4:** Velocity is -0.65, then $f(-0.65) \approx 0.343$ and threshold is 0.5. Here, since $\{0.343 < 0.5\}$. So, the new position is 0.
- **For Dimension 5:** Velocity is 0.3, then $f(0.3) \approx 0.574$ and threshold is 0.5. Here, since $\{0.574 > 0.5\}$. So, the new position is 1.
- **For Dimension 6:** Velocity is 1.7, then $f(1.7) \approx 0.846$ and threshold is 0.5. Here, since $\{0.846 > 0.5\}$. So, the new position is 1.

Now the updated position is {1, 0, 1, 0, 1, 1}.

In **step 5**, we calculate fitness for the updated position, i.e {1, 0, 1, 0, 1, 1}. Since the position does not form a valid independent set because nodes 3, 5, and 6 are all connected, this independent set is not a valid independent set because nodes 1, 3, 5 and 6 are mutually not adjacent. So the fitness score is zero. Now, we calculate fitness for all the particles.

In **step 6**, we apply local search; it refines the position to ensure that it represents a valid independent set {1, 0, 1, 0, 1, 1}. From this independent set, we analyze the provided graph depicted in **Fig.1**. In the updated position set, the position vector for node 1 is observed to be 1. So, the node 1 will be considered for inclusion in the independent set. Here, it is evident that node 1 is connected to nodes 2 and node 3 (which we do not consider for inclusion in the independent set).

For the node 2, we see the position vector is zero, so this node will not be considered for inclusion in the independent set.

For the node 3, as its value is 1 in the updated position vector. But it also cannot be included in the independent set because it is adjacent to node 1.

For the node 4, the position vector is zero in the updated position. So, this node cannot be considered for inclusion in the independent set.

For the node 5, we see the position vector is 1 in the updated position.We will add this node since it is not adjacent to any nodes currently in the independent set.

For the node 6 whose position value is 1 in the position vector. But this node cannot be included as it is connected to the included node 5.

Step 7: Update the Personal Best Position: The particle's personal best position and its present location are compared for fitness. The particle updates its personal best position and fitness if the current position's fitness exceeds its personal best fitness.

Step 8: Update the Global Best Position: The algorithm determines whether the personal best location of every particle is more fit than the present global best. If so, the fitness and global best position are changed to reflect the new best values. Until a maximal independent set is discovered, same procedures are carried out for every particle.

TABLE I: DIMACS10 Benchmarks Dataset Results

DIMACS10 Benchmarks Dataset	**Nodes**	**Edges**	**Lower Bound (Maximum Clique)**	**GWO**	**GA**	**PSO**
chesapeake	39	170	5	8	9	17
delaunay_n10	1K	3K	4	28	196	287
delaunay_n11	2K	6.1K	4	58	393	455
delaunay_n12	4.1K	12.3K	4	71	815	1112
delaunay_n13	8.2K	24.5K	4	135	1662	1765
delaunay_n14	16.4K	49.1K	4	186	3235	3482
fe-4elt2	11K	33K	4	155	2315	2504
rgg_n_2_15_s0	33K	160K	10	315	5148	5307
wing_nodal	10.9K	75.5K	6	83	1320	1401
cti	17K	48K	3	346	4077	4532

V. Result Analysis

This section presents the result analysis.

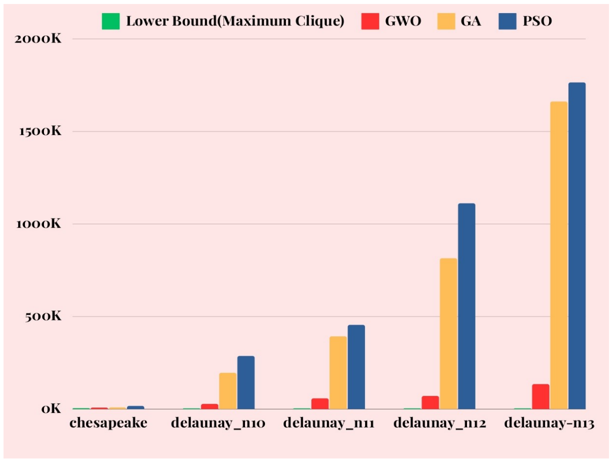

Fig. 2: Comparison of results simulated on different DIMACS10 datasets [24]

We simulated our proposed algorithm by using *C#* coding language on the Windows 11 Pro operating system, Intel (R) Core (TM) $i7/8550UCPU@$1.80 GHz 1.99 GHz (8^{th} Generation, RAM 8GB, System Type 64 Bit Operating System. We have evaluated the algorithms by testing on the various DIMACS benchmark datasets [24].

Here **TABLE I** and **Fig. 2** - **Fig. 3** Present the simulation results from applying the algorithm to the DIMACS10

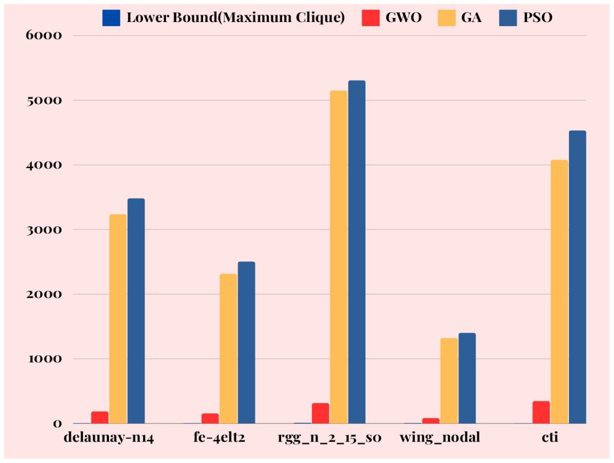

Fig. 3: Comparison of results simulated on different DIMACS10 datasets [24]

benchmark graph datasets.Here we see that our proposed algorithm PSO performs better as compared to GWO and GA algorithms. We simulated all the algorithms and then compared the results on the same platform.

VI. Conclusions and Future Direction

This paper proposes a PSO-based algorithm for tackling the NP-hard problem of finding Maximum Independent Sets (MaxIS). We simulated our algorithm on DIMACS10 benchmark datasets. The paper also compared the results with other soft computing algorithms, such as GWO and GA. We found that our proposed algorithm performs better than GWO and GA.

References

[1] Adil, B., Chakir, L., et al.: Chn and swap heuristic to solve the maximum independent set problem. International Journal of Electrical and Computer Engineering **7**(6), 3583 (2017)

[2] Aggarwal, C.C., Orlin, J.B., Tai, R.P.: Optimized crossover for the independent set problem. Operations research **45**(2), 226–234 (1997)

[3] Agrawal, J., Agrawal, S.: Acceleration based particle swarm optimization for graph coloring problem. Procedia Computer Science **60**, 714–721 (2015)

[4] Almara'beh, H., Suleiman, A.: Heuristic algorithm for graph coloring based on maximum independent set. Journal of Applied Computer Science & Mathematics **13**(6), 19–23 (2012)

[5] Alon, N., Babai, L., Itai, A.: A fast and simple randomized parallel algorithm for the maximal independent set problem. Journal of Algorithms **7**(4), 567–583 (1986). https://doi.org/https://doi.org/10.1016/0196-6774(86)90019-2

[6] Back, T., Khuri, S.: An evolutionary heuristic for the maximum independent set problem. In: Proceedings of the first IEEE conference on evolutionary computation. IEEE World Congress on Computational Intelligence. pp. 531–535. IEEE (1994)

[7] Bader, D.A., Meyerhenke, H., Sanders, P., Wagner, D.: 10th dimacs implementation challenge-graph partitioning and graph clustering. Georgia: Georgia Institute of Technology (2011)

[8] Bai, Q.: Analysis of particle swarm optimization algorithm. Computer and information science **3**(1), 180 (2010)

[9] Blelloch, G.E., Fineman, J.T., Shun, J.: Greedy sequential maximal independent set and matching are parallel on average. In: Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures. pp. 308–317 (2012)

[10] Bollobás, B., Bollobás, B.: Random graphs. Springer (1998)

[11] Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The maximum clique problem. Handbook of Combinatorial Optimization: Supplement Volume A pp. 1–74 (1999)

[12] Das, K.N., Chaudhuri, B.: Heuristics to find maximum independent set: An overview. In: Deep, K., Nagar, A., Pant, M., Bansal, J.C. (eds.) Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011. pp. 881–892. Springer India, India (2012)

[13] Feo, T.A., Resende, M.G., Smith, S.H.: A greedy randomized adaptive search procedure for maximum independent set. Operations Research **42**(5), 860–878 (1994)

[14] Hifi, M.: A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems. Journal of the Operational Research Society **48**(6), 612–622 (1997). https://doi.org/10.1057/palgrave.jors.2600405, `https://doi.org/10.1057/palgrave.jors.2600405`

[15] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. vol. 4, pp. 1942–1948. ieee (1995)

[16] Krivelevich, M., Mészáros, T., Michaeli, P., Shikhelman, C.: Greedy maximal independent sets via local limits. Random Structures & Algorithms **64**(4), 986–1015 (2024)

[17] Li, Y., Xul, Z.: An ant colony optimization heuristic for solving maximum independent set problems. In: Proceedings Fifth International Conference on Computational Intelligence and Multimedia Applications. ICCIMA 2003. pp. 206–211 (2003). https://doi.org/10.1109/ICCIMA.2003.1238126

[18] Linial, N.: Locality in distributed graph algorithms. SIAM Journal on Computing **21**(1), 193–201 (1992). https://doi.org/10.1137/0221015, `https://doi.org/10.1137/0221015`

[19] Lovász, L.: On the shannon capacity of a graph. IEEE Transactions on Information theory **25**(1), 1–7 (1979)

[20] Luby, M.: A simple parallel algorithm for the maximal independent set problem. In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing. p. 1–10. STOC '85, Association for Computing Machinery, New York, NY, USA (1985). https://doi.org/10.1145/22145.22146

[21] Nayeem, S.M.A., Pal, M.: Genetic algorithmic approach to find the maximum weight independent set of a graph. Journal of Applied Mathematics and Computing **25**, 217–229 (2007)

[22] Panconesi, A., Srinivasan, A.: On the complexity of distributed network decomposition. J. Algorithms **20**(2), 356–374 (mar 1996). https://doi.org/10.1006/jagm.1996.0017

[23] Pardalos, P.M., Xue, J.: The maximum clique problem. Journal of global Optimization **4**, 301–328 (1994)

[24] Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015), `https://networkrepository.com`

[25] Sakamoto, A., Liu, X., Shimamoto, T.: A genetic approach for maximum independent set problems. IEICE transactions on fundamentals of electronics, communications and computer sciences **80**(3), 551–556 (1997)

[26] Shi, Y., Krohling, R.: Co-evolutionary particle swarm optimization to solve min-max problems. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). vol. 2, pp. 1682–1687 vol.2 (2002). https://doi.org/10.1109/CEC.2002.1004495

[27] Silva-Muñoz, M., Contreras-Bolton, C., Rey, C., Parada, V.: Automatic generation of a hybrid algorithm for the maximum independent set problem using genetic programming. Applied Soft Computing **144**, 110474 (2023)

[28] Taranenko, A., Vesel, A.: An elitist genetic algorithm for the maximum independent set problem. In: Proceedings of the 23rd International Conference on Information Technology Interfaces, 2001. ITI 2001. pp. 373–378. IEEE (2001)

[29] Xia, X., Peng, X., Liao, W.: On the analysis of ant colony optimization for the maximum independent set problem. Frontiers of Computer Science **15**(4), 154329 (2021)