*Mathematical and Computational Applications*, Vol. 18, No. 3, pp. 495-501, 2013

# A NEW HEURISTIC ALGORITHM TO SOLVE THE MAXIMUM INDEPENDENT SET PROBLEM

Onur Ugurlu

Department of Mathematics, Ege University, 35100, Bornova, Izmir, Turkey
onur__ugurlu@hotmail.com

**Abstract –**The Maximum Independent Set Problem is a classic graph optimization NP-hard problem. Given a graph G=(V,E), the independent set problem is that of finding a maximum-cardinality subset S of V such that no two vertices in S are adjacent. In this paper, the maximum independent set problem is discussed and a new heuristic algorithm is proposed to solve this problem. The performance of the algorithm has been tested on DIMACS benchmark instances and compared with the literature works. The experimental results show that the proposed approach can yield good solutions.

**Key Words –** maximum independent set problem, heuristics algorithms, optimization, NP-hard problems.

## 1. INTRODUCTION

An independent set in a graph G=(V,E) is a subset $V' \subseteq V$ such that, for all u, $v \in V'$, the edge (u,v) is not in E [1]. The maximum independent set problem calls for finding the independent set of maximum cardinality. The maximum independent set is a classic one in computer science and graph theory and is known to be NP-hard [1]. The maximum independent set problem has many important applications, including combinatorial auctions, graph coloring, coding theory, geometric tiling, fault diagnosis, pattern recognition, scheduling, computer vision, molecular biology, and more recently its application in bioinformatics have become important [2].

In solving the maximum independent set problem, we also have a solution for two other important graph problems: The minimum vertex cover problem and the maximal clique problem [1].

A vertex cover of an undirected graph G=(V,E) is a subset $V^* \subseteq V$ such that if $(u,v) \in E$, then $u \in V^*$ or $v \in V^*$(or both). That is, each vertex "covers" its incident edges, and a vertex cover for G is a set of vertices that covers all the edges in E. The size of a vertex cover is the number of vertices in it [3].

The clique in an undirected graph G=(V,E) and subset is $V^* \subseteq V$, a subset $V^* \subseteq V$ of vertices, each pair of which is connected by an edge in E. In other words, a clique is a complete subgraph of G [3].

The following relationship of independent sets, cliques, and vertex covers are easy to verify [1].

The *Lemma:* For any graph G=(V,E) and subset $V^* \subseteq V$, the following statements are equivalent:

- $V^*$ is a vertex cover for G.
- $V\text{-}V^*$ is an independent set for G.

- V-$V^*$ is a clique in the complement $G^C$ of G, where $G^C$ =(V,$E^C$) with $E^C$ = {(u,v): u, v $\in$ V and (u,v) $\notin$ E}.

Consequently, one can obtain a solution of the minimum vertex cover problem by taking the complement of the solution to the maximum independent set problem. A solution to the maximum clique problem is obtained by applying the maximum independent set heuristic to $G^C$=(V,$E^C$) [1].

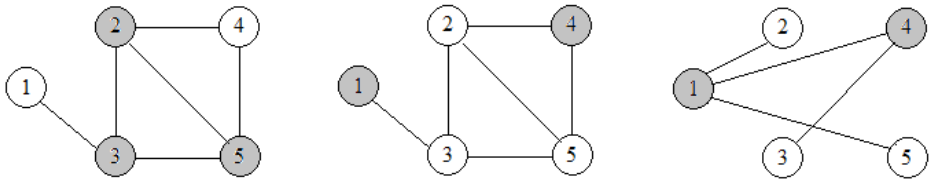


Figure 1. The relationship of independent set, clique, and vertex cover; a) Mininum vertex cover of G. b) Maximum independent of for G. c) Maximum clique of $G^C$.

A large number of exact, approximate, heuristic and metaheuristic algorithms have been proposed for the independent set problem. Some of these algorithms may be found in Balas and Yu (1986); Barbosa and Campos (2008); Bron and Kerbosch (1973); Busygin et al. (2002); Carraghan and Pardalos (1990); Boppana and Halldorsson (1992); Loukakis and Tsouros (1982); Gibbons et al. (1991); and a complete survey of all algorithms related to this problem may be found in Pardalos and Xue (1994).

In this paper, a new heuristic algorithm called Disassemble Algorithm (DA) is presented to find the maximum independent set of the graph. The proposed algorithm has been tested on the DIMACS benchmark instances and compared with other existing algorithm. The experimental results show that the proposed algorithm yields better solutions.

The paper is organized as follows: Section II outlines the proposed algorithm and its complexities. In Section III, computational efficiency of the proposed algorithm is discussed, and the proposed algorithm is compared with two other heuristics on the DIMACS benchmark instances. Section IV summarizes and concludes the paper.

## 2. BASIC DEFINITION, ALGORITHM AND COMPUTATIONAL COMPLEXITY

### 2.1 Basic Definition

Neighborhood of a vertex: Let G=(V,E) be an undirected graph. Then for each v $\in$ V, the neighborhood of v is defined by N(v) = { u $\in$ V | u is adjacent to v}.

Degree of vertex: The degree of vertex v $\in$ V is denoted by d(v) and is defined by the number of neighbors of v.

**2.2 . The Disassemble Algorithm (DA)**

The algorithm is designed to find the largest independent set of a graph. Algorithm starts selecting one initial vertex and tries to add its neighbors to independent set. If the neighbor vertex does not make the solution unfeasible, the vertex is added to the solution. After updating the adjacency matrix of graph by deleting all edges which are adjacent to vertex in solution and its neighbor, algorithm finds the vertex which has the maximum degree and tries to add its neighbor to solution alike. This procedure continuous until all vertices degrees become zero. After finding an initial solution, the algorithm search for the vertex that does not make solution unfeasible when added to it. If there is no vertex to add the solution, algorithm tries to create addible vertex by shifting the vertices that are in the solution. The pseudo-code of the disassemble algorithm is given below.

**Input:** G(V,E).
**Output:** A maximal independent set I* and its cardinality α.

1. $I^* \leftarrow 0$
2. **for** i=1 **to** n **begin**
3. update the adjacency matrix
4. add neighbors of $v_i$ to $I_i$ which do not make $I_i$ unfeasible
5. **do**
6. Find the maximum degree vertex $v_m$
7. add neighbors of $v_m$ to $I_i$ which do not make $I_i$ unfeasible
8. delete all edges which are adjacent to vertices in $I_i$ and their all neighbors
9. **while** ( $E \neq \phi$ )
10. **If** $v_k$ is not in $I_i$ and all vertices in $N(v_k)$ are not in $I_i$ as well **then** add $v_k$ to $I_i$
11. **if** $v_m$ is not in $I_i$ and only one neighbor $v_n$ in $I_i$ **then** remove $v_n$ from $I_i$, add $v_m$ to $I_i$ **and** go to step 11
12. **if** $I_i > I^*$ **then** $I^* = I_i$ and $\alpha = |I_i|$
13. **end**
14. **end.**

Figure 1. The pseudo-code of the proposed algorithm.

**2.3 . Computational Complexity**

The worst case complexity of the proposed algorithm can be obtained as follows: Assume that there are *n* vertices and the maximum independent sets cardinality *p*, after select an initial vertex and control its neighbors, picking the vertex which has maximum degree requires $O(n^2)$. Controlling the neighbors of the selected vertex to add solution requires $O(p)$. Use all vertices as initial vertex require $O(n)$. So the overall running time of the procedure of DA can be deduced as follows: $O(n(n^2.p)) = O(p.n^3)$.

The results of numerical experiments with the algorithm are reported in the next part of the paper.

## 3. EXPERIMENTAL RESULTS

This section presents results of computational experiments for the proposed heuristic. The algorithm has been tested on the complement graphs of DIMACS clique instances [13]. The heuristic has been compared with continuous based heuristic [4] (CBH) and heuristic based on optimization of a quadratic over a Sphere [5] (QSH). The comparison of the algorithm with previous work is summarized in table 1,2 and 3.

In table 1,2 and 3, the first four columns represent the name of the instance, number of its vertices, its density and cardinality of maximum independent set, respectively. This information is available from the DIMACS web site. The densities are specified for the original (maximum clique) instances and the last there column denotes CBH, QSH and the proposed algorithm DA, correspondingly.

In table 4, the performance of DA, CBH and QSH are compared, respectively. In each row of the table, first column contains a name of the family of graphs, where each family consists of a group of graphs with the same letter names. The second column represents the number of instances from each family for which CBH found the best solution. Similarly, the next two columns represent QSH found the best solution and DA found the best solution, respectively.

Table 1. The simulation results for DIMACS bencmark graphs

| Graph Name | \|V\| | Density | α(G) | CBH | QSH | DA |
|---|---|---|---|---|---|---|
| brock200_1 | 200 | 0.745 | 21 | 20 | 21 | 20 |
| brock200_2 | 200 | 0.496 | 12 | 12 | 12 | 10 |
| brock200_3 | 200 | 0.605 | 15 | 14 | 15 | 13 |
| brock200_4 | 200 | 0.658 | 17 | 16 | 17 | 12 |
| brock400_1 | 400 | 0.748 | 27 | 23 | 27 | 24 |
| brock400_2 | 400 | 0.749 | 29 | 24 | 29 | 25 |
| brock400_3 | 400 | 0.748 | 31 | 23 | 31 | 31 |
| brock400_4 | 400 | 0.749 | 33 | 24 | 33 | 24 |
| brock800_1 | 800 | 0.649 | 23 | 20 | 17 | 20 |
| brock800_2 | 800 | 0.651 | 24 | 19 | 24 | 20 |
| brock800_3 | 800 | 0.649 | 25 | 20 | 25 | 20 |
| brock800_4 | 800 | 0.650 | 26 | 19 | 26 | 20 |

Table 2. The simulation results for DIMACS bencmark graphs

| Graph Name | \|V\| | Density | α(G) | CBH | QSH | DA |
|---|---|---|---|---|---|---|
| c-fat200-1 | 200 | 0.077 | 12 | 12 | 12 | 12 |
| c-fat200-2 | 200 | 0.163 | 24 | 24 | 24 | 24 |
| c-fat200-5 | 200 | 0.426 | 58 | 58 | 58 | 58 |
| c-fat500-1 | 500 | 0.036 | 14 | 14 | 14 | 14 |
| c-fat500-2 | 500 | 0.374 | 26 | 26 | 26 | 26 |

| | | | | | | |
|---|---|---|---|---|---|---|
| c-fat500-5 | 500 | 0.073 | 64 | 64 | 64 | 64 |
| c-fat500-10 | 500 | 0.186 | 126 | 126 | 126 | 126 |
| hamming6-2 | 64 | 0.905 | 32 | 32 | 32 | 32 |
| hamming6-4 | 64 | 0.349 | 4 | 4 | 4 | 4 |
| hamming8-2 | 256 | 0.969 | 128 | 128 | 128 | 128 |
| hamming8-4 | 256 | 0.639 | 16 | 16 | 16 | 16 |
| johnson8-2-4 | 28 | 0.556 | 4 | 4 | 4 | 4 |
| johnson8-4-4 | 70 | 0.768 | 14 | 14 | 14 | 14 |
| johnson16-2-4 | 120 | 0.765 | 8 | 8 | 8 | 8 |
| johnson32-2-4 | 496 | 0.879 | 16 | 16 | 16 | 16 |
| keller4 | 171 | 0.649 | 11 | 10 | 11 | 11 |
| keller5 | 776 | 0.751 | 27 | 21 | 24 | 26 |
| MANN_a9 | 45 | 0.927 | 16 | 16 | 16 | 16 |
| MANN_a27 | 378 | 0.990 | 126 | 121 | 125 | 126 |
| p_hat300-1 | 300 | 0.244 | 8 | 8 | 7 | 8 |
| p_hat300-2 | 300 | 0.489 | 25 | 25 | 24 | 16 |
| p_hat300-3 | 300 | 0.744 | 36 | 36 | 33 | 11 |
| p_hat500-1 | 500 | 0.253 | 9 | 9 | 9 | 9 |
| p_hat500-2 | 500 | 0.505 | 36 | 35 | 33 | 36 |
| p_hat500-3 | 500 | 0.752 | 50 | 49 | 46 | 50 |
| p_hat700-1 | 700 | 0.249 | 11 | 11 | 8 | 10 |
| p_hat700-2 | 700 | 0.498 | 44 | 44 | 42 | 44 |
| p_hat700-3 | 700 | 0.748 | 62 | 60 | 59 | 62 |
| san200_0.7_1 | 200 | 0.700 | 30 | 15 | 30 | 17 |
| san200_0.7_2 | 200 | 0.700 | 18 | 12 | 18 | 18 |
| san200_0.9_1 | 200 | 0.900 | 70 | 46 | 70 | 70 |
| san200_0.9_2 | 200 | 0.900 | 60 | 36 | 60 | 60 |
| san200_0.9_3 | 200 | 0.900 | 44 | 30 | 35 | 44 |
| san400_0.5_1 | 400 | 0.500 | 13 | 8 | 9 | 8 |
| san400_0.7_1 | 400 | 0.700 | 40 | 20 | 40 | 21 |
| san400_0.7_2 | 400 | 0.700 | 30 | 15 | 30 | 30 |
| san400_0.7_3 | 400 | 0.700 | 22 | 14 | 16 | 22 |
| san400_0.9_1 | 400 | 0.900 | 100 | 50 | 100 | 100 |

Table 3. The simulation results for DIMACS bencmark graphs

| **Graph Name** | **\|V\|** | **Density** | **α(G)** | **CBH** | **QSH** | **DA** |
|---|---|---|---|---|---|---|
| sanr200_0.7 | 200 | 0.700 | 18 | 18 | 15 | 18 |
| sanr200_0.9 | 200 | 0.900 | 42 | 41 | 37 | 42 |
| sanr400_0.5 | 400 | 0.500 | 13 | 12 | 11 | 12 |
| sanr400_0.7 | 400 | 0.700 | 21 | 20 | 18 | 21 |

As one can see, DA performed better than CBH and QSH on four families (MANN a, p-hat, keller, sanr) and did worse than QSH on two (brock, san). The proposed algorithm found the optimal solution 38 of 54 instances whereas QSH found 36 and CBH found 24. The results of numerical experiments show that DA can yield good solutions inasmuch as to CBH and QSH.

Table 4. Comparison of the results on benchmark instances.

| Graphs Family Name | CBH | QSH | DA |
|---|---|---|---|
| brock | 1 | 11 | 2 |
| c-fat | 7 | 7 | 7 |
| hamming | 4 | 4 | 4 |
| johnson | 4 | 4 | 4 |
| keller | 0 | 1 | 2 |
| MANN_a | 1 | 1 | 2 |
| p-hat | 6 | 1 | 8 |
| san | 0 | 8 | 7 |
| sanr | 2 | 0 | 4 |
| Total | 25 | 37 | 40 |

## 4. CONCLUSION

In this work, a new heuristic algorithm for the maximum independent set problem is presented. The implementations of the algorithm were coded in C++ language and the performance of the proposed algorithm is tested on DIMACS instances. The computational experiments show that DA can yield good solutions as much as the existing heuristics in the literature.

## 5. REFERENCES

1. M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the theory NP–completeness*, San Francisco: Freeman, 1979.
2. S. Balaji, V. Swaminathan, K. Kannan, Simple Algorithm to Optimize Maximum Independent Set, *Advanced Modeling and Optimization* **12(1)**, 2010.
3. T. H. Cormen, C. E. Lieserson, R. L. Rivest and C. Stein, *Introduction to Algorithms, Second Edition*, The MIT Press, 2001.
4. L. E. Gibbons, D. W. Hearn and P. M. Pardalos, A Continuous Based Heuristic for the Maximum Clique Problem, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **26**, 103-124, 1994.
5. S. Busgin, S. Butenko and P. M. Pardalos, A Heuristic for the Maximum Independent Set Problem Based on Optimization of a Quadratic Over a Sphere, *Journal of Combinatorial Optimization* **6**, 287-297, 2002.
6. E. Loukakis and C. Tsouros, Determining the Number of Internal Stability of a Graph, *Journal of Computational Mathematics* **11**, 232-248, 1982.
7. R. Carraghan and P. M. Pardalos, An Exact Algorithm for the Maximum Clique Problem, *Operation Research Letter* **9**, 375-38, 1990.

8. P. M. Pardalos and J. Xue, The Maximum Clique Problem, *Journal of Global Optimization* **4**, 301-328, 1994.
9. V. C. Barbosa and L. C. D. Campos, *A Novel Evolutionary Formulation of the Independent Set Problem*, 2008.
10. R. Boppana and M. M. Halldorsson, Approximating Maximum Independent Set by Excluding Subgraphs, *Bit Numerical Mathematics* **32**, 180-196, 1992.
11. E. Balas and C. S. Yu, Finding the Maximum Clique in an Arbitrary Graph, *SIAM Journal on Computing* **15**, 1054-1068, 1986.
12. C. Bron and J. Kerbosch, Finding All Cliques of an Undirected Graph, *Communications of the ACM* **16**, 575-577, 1973.
13. DIMACS clique benchmarks, Benchmark instances made available by electronic transfer at dimacs.rutgers.edu, Rutgers Univ., Piscataway. NJ. 1993.