A Simple Algorithm to Optimize Maximum Independent Set

S. Balaji [a,*] V. Swaminathan [b], K. Kannan [c]

[a, c] Department of Mathematics, SASTRA University, Thanjavur – 613 401, India.

[b] Ramanujan Research centre in Mathematics, Saraswathi Narayanan College, Madurai – 625 022, India.

Abstract:

The Maximum Independent Set Problem (MIS) is a classic graph optimization NP-hard problem with many real world applications. 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 an efficient algorithm, called Vertex Support Algorithm (VSA), is designed to find the maximum independent set of a graph. Our algorithm was tested on a large number of random graphs with upto 5,000 vertices and on DIMACS benchmark graphs and the result shows that VSA algorithm decidedly outperforms other existing algorithms found in the literature for solving the MIS.

Keywords – independent set, vertex cover, vertex support, algorithms, NP-hard problem.

1. Introduction:

An independent set of a graph is a subset of vertices in which no two vertices are adjacent. Given a set of vertices, the maximum independent set problem (MIS) calls for finding the independent set of maximum cardinality. The MIS is a classic one in computer science and in graph theory, and is known to be NP-hard [10]. MIS has many important applications, including combinatorial auctions [7], graph coloring, coding theory [9], geometric tiling, fault diagnosis, pattern recognition, molecular biology, and more recently its application in bioinformatics have become important [16].

The Minimum Vertex Cover (MVC) problem of a graph consists of identifying the vertex cover of the graph which has minimum cardinality. The MIS and MVC problems are related in that the maximum independent set contains all those vertices that are not in the minimum vertex cover of the graph. Due to computational intractability of the MIS (MVC) problem, many researchers have instead focused their attention on the design of approximation algorithm for delivering quality solutions in a reasonable time.

* Corresponding author.

[a,*] balaji_maths@yahoo.com, [b] sulanesri@yahoo.com, [c] kkannan@maths.sastra.edu.

AMO-Advanced Modeling and Optimization. ISSN: 1841-4311.

Pardalos and Xue [15] recently published a review with 260 references. Many algorithms for the MIS have been proposed [1, 4, 5, 6, 11, 17]. Recently, Ostergard [14] proposed a new maximum clique algorithm, which was supported by computational experiments.

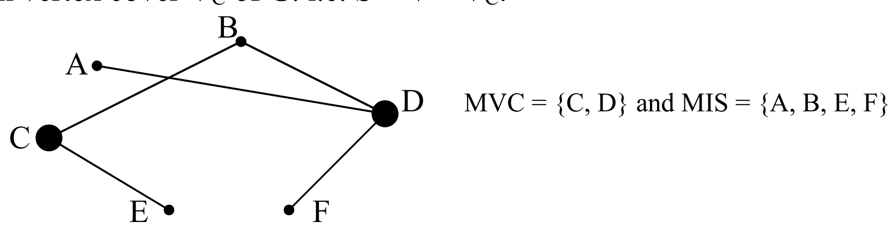
The MIS problem has a relation with the Maximum Clique Problem (MCP) i.e., an independent set of size k in a graph G (V, E) is a clique in the complemented graph $\overline{G}(V, \overline{E})$ and a graph has independent set of size k if and only if it has a vertex cover of size |V|-k. Clearly, S is an independent set of G if and only if it is a clique of $\overline{G}$. In this paper we are concerned with the problem of finding a maximum independent set in G, that is, independent set of maximum size. Equivalently, this problem can be viewed as asking for a minimum vertex cover in G or maximum clique in $\overline{G}$. The MIS (MVC) of a graph is approximated by the new approach support of a vertex which is calculated by summing up all the degrees of the vertices which are adjacent to the vertex.

In this paper a competent algorithm called Vertex Support Algorithm (VSA) is presented to find the maximum independent set of the graph, which calculates the MIS through MVC by support of the vertices. We compared our algorithm with the other existing algorithm namely [1, 4, 12]. The experimental result shows that our algorithm is very fast and yields better solutions than the compared algorithms for many random graphs and DIMACS benchmark graphs.

The paper is organized as follows. Section 2 briefly describes the maximum independent set (MIS) problem and the minimum vertex cover problem (MVC) and its theoretical background. Section 3 outlines the proposed algorithm VSA. In Section 4 graph model used in the experiments is briefly described. Section 5 provides experiments done and their results. Section 6 summarizes and concludes the paper.

2. Maximum Independent Set and Minimum Vertex Cover

Let G = (V, E) be an arbitrary undirected graph, where V = {1, 2, …, n} is the set of vertices and $E \subseteq V \times V$ (not in ordered pairs) is the set of edges. Two distinct vertices u and v are called adjacent if they are connected by an edge, an independent set S of G is a subset of V whose elements are pairwise non-adjacent. The MIS problem seeks to find an independent set with large number of vertices. The size of the maximum independent set of G is the stability number of G and is denoted by α. A vertex cover for G is a subset V_C of V such that for each edge (u, v) $\in$ E, at least one of u or v or both belongs to V_C. The MVC problem consists of identifying the vertex cover V_C of G which has minimum cardinality. The MIS and MVC problems are related in that the maximum independent set S of G contains all those vertices that are not in the minimum vertex cover V_C of G. i.e. $S = V - V_C$.



Graph G(V, E)
Figure 1.

Fig. 1 depicts the above statement briefly. There are two versions of the vertex cover problem: the decision and optimization versions. In the decision version, the task is to verify for a given graph G whether there exists a vertex cover of a specified size but in the optimization version the task is to find a vertex cover of minimum size. In this paper we consider the optimization version of the minimum vertex cover with the goal of obtaining optimum solution. Now the minimum vertex cover problem is formulated as an integer programming problem by using the following conditions:

Binary variables a_{ij} (i = 1,2,3,…,n; j = 1,2,3,…,n) form the adjacency matrix of the graph G. Each variable has only two values (1 or 0) according as an edge exists or not. In other words, if an edge (v_i, v_j) is in E then $a_{ij} = 1$ else $a_{ij} = 0$. The output of the program expresses the vertex v_i is in the independent set or not. $v_i = 1$ if it is in the independent set otherwise $v_i = 0$. Thus the number of vertices in the independent set can be expressed by $Z = \sum v_i$ and any one or none of the vertex of the edge (v_i, v_j) is included in the independent set, we can write the constrained condition of the MIS as $v_i + v_j \le 1$. Thus the problem can be mathematically transformed into the following optimization integer programming problem as follows.

MIS:

$$\begin{aligned} &\text{Max } Z = \sum v_i \\ &\text{Subject to} \\ &v_i + v_j \le 1 \quad \forall (v_i, v_j) \in E \\ &v_i \in \{0,1\} \quad \forall v_i \in V \end{aligned}$$

In this paper a competent algorithm VSA is proposed for the MIS (MVC) problems. The simulation results show that the new VSA can yield better solutions, to random graphs, than other existing algorithms. The iterations are much less than the original model. Moreover, for some DIMACS graphs, the new VSA can yield 100% convergence rate to optimal solutions.

3. Terminologies & Proposed Algorithm

Neighborhood of a vertex:

Let G = (V, E), V is a vertex set and E is an edge set, be an undirected graph and let $|V| = n$ *and* $|E| = m$. Then for each $v \in V$, the neighborhood of v is defined by $N(v) = \{u \in V / u \text{ is adjacent to } v\}$ and $N[v] = v \cup N(v)$.

Degree of a vertex:

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

Support of a vertex:

The support of a vertex $v \in V$ is defined by the sum of the degree of the vertices which are adjacent to v, i.e., support (v) = s (v) = $\sum_{u \in N(v)} d(u)$.

3.1. Algorithm

The following algorithm is designed to find the maximum independent set of a graph G. Adjacency matrix A = (a_{ij}) of the given graph G of n vertices and m edges is given as the input of the program and the degree $d(v_i)$, support $s(v_i)$ of the each vertex $v_i \in V$ of the graph G are calculated. Support of the vertex $v \in V$ is found by the relation $\sum_{u \in N(v)} d(u)$. Add the vertex which has the maximum value of the support into the vertex cover V_C. If one or more vertices have equal maximum value of the support, in this case if (degree (v_i)>=degree (v_j)), add the vertex v_i into the vertex cover V_C otherwise add v_j into V_C. Update the adjacency matrix by putting zero in to the row and column entries of the corresponding vertex $v \in V_C$. Proceed the above process until G has no edges. i.e., up to $a_{ij} \neq 0 \; \forall$ i, j. Finally the maximum independent set S of the graph G is extracted from the minimum vertex cover V_C of the graph G by S = V - V_C. The pseudo-code of the proposed algorithm is described in Fig. 2.

Algorithm 3.1.1: Vertex Support Algorithm (VSA)

Input: G (V, E)
Output: Max. Independent Set S (G) = V - V_C where V_C is the minimum vertex cover of G

1. $V_C \leftarrow \phi$; selection = 0;
2. $\forall v_i \in V$
3. do
4. $d(v_i) = \sum_j a_{ij}$; $s(v_i) = \sum_{v_j \in N(v_i)} d(v_j)$;
5. Max = $s(v_1)$, k = 1;
6. **for** $i \leftarrow 2$ to n
7. **if** max < $s(v_i)$ then
8. t = i;
9. **else if** max = $s(v_i)$ and $d(v_{i-k})$ <= $d(v_i)$ then
10. t = i;
11. **else** max = $s(v_i)$ and $d(v_{i-k})$ > $d(v_i)$ then
12. t = i-k;
13. k = k + 1;
14. end for
15. $V_C = V_C \cup \{v_t\}$
16. selection = selection + 1;
17. assign the value zero to the t^{th} row and t^{th} column of the matrix A = (a_{ij});
18. while $A \neq (0)$
19. | V_C | = selection;
20. end.

Figure 2: The pseudo-code of the proposed algorithm

4. Experimental Results & analysis

All the procedures of VSA have been coded in C++ language. The experiments were carried out on an Intel Pentium Core2 Duo 1.6 GHz CPU and 1 GB of RAM. The effectiveness of the VSA heuristic was evaluated using 136 instances. These instances are divided into 3 sets as shown in the Table 1. Simulations are carried out on three types of graphs: the randomly generated small size, moderate and large scale graphs for the maximum independent set problem.

Table 1
MIS Instances

Set	No. of Instances	Scale	Graph Model	Optimal Solution
1	36	small-large	G (n, p)	Unknown
2	80	small-large	DIMACS	Known
3	20	moderate	G(n, m)	Unknown

4.1 G (n, p) Model

The G (n, p) model is also called Erdös Renyi random graph model [2], consists of graphs of n vertices for which the probability of an edge between any pair of nodes is given by a constant p > 0. To ensure that graphs are almost always connected, p is chosen so that $p >> \frac{\log(n)}{n}$. To generate a G (n, p) graph we start with an empty graph. Then we iterate through all pairs of nodes and connect each of these pairs with probability p. The pseudo code for generating G (n, p) graph is shown in the Fig. 3.

Algorithm 4.1: Generate (G, n, p)

```
Initialize graph G (V, E)
for i ← 1 to n
for j ← i+1 to n
add edge (i, j) to E with probability p
return (G)
```

Figure 3: The pseudo-code for generating G(n, p) graphs

The expected number of edges of G (n, p) graph is $\binom{n}{2}p$ and expected degree is np. Graphs are generated for different p and n values.

4.2 G(n, m) Model

The G(n, m) model consists of all graphs with n vertices and m edges. The number of vertices, n and the number of edges, m are related by m = nc, where c > 0 is constant.

To generate a random G(n, m) graph, we start with a graph with no edges. Then, cn edges are generated randomly using uniform distribution over all possible graphs with cn edges. Each node is thus expected to connect to 2c other nodes on average. The pseudo-code for the random graph generation is shown in Fig. 4.

Algorithm 4.2: Generate (G, n, c)

Initialize graph G(V, E)
$m \leftarrow n * c$
for $i \leftarrow 1 \text{ to } m$
repeat
$e \leftarrow \text{random edge}$
until e not present in E
$E \leftarrow E \cup \{e\}$
return (G)

Figure 4: The pseudo-code for generating G (n, m) graphs

4.2 Results for random graphs

We first tested the VSA on 36 random graphs generated based on the concept explained in Section 4.1. The result we recorded for each test graph and their information are shown in the Table 2 and these results are compared with the theoretical evaluation of expected maximum clique for G(n, p) random graphs, shown in [14], and it is guaranteed that the proposed algorithm estimations are quite well to the expected size of the maximum independent set. In the 36 instances tested the maximum time taken of 29 seconds, (3000, 0.8; 4000, 0.9 & 5000, 0.8), is an encouraging one but also it is comparatively very less time for finding the maximum clique of random graphs of large number of vertices with high density. So, it is interest to see the performance of the proposed algorithm on benchmark graphs with known optimal (best known) solutions.

Table 2.
Simulation results for the G(n, p) random graphs.

n	p	Proposed VSA		n	p	Proposed VSA		n	p	Proposed VSA	
		|C|	Time (s)			|C|	Time (s)			|C|	Time (s)
100	0.7	15	<1	400	0.7	23	<1	2000	0.7	129	23
	0.8	20	<1		0.8	31	2		0.8	142	18
	0.9	31	<1		0.9	53	4		0.9	159	27
150	0.8	23	<1	500	0.7	32	<1	3000	0.7	143	15
	0.9	37	3		0.8	41	5		0.8	167	29
	0.95	54	2		0.9	59	3		0.9	189	17
200	0.7	19	<1	700	0.7	46	6	4000	0.7	173	28
	0.8	26	3		0.8	52	3		0.8	206	27
	0.9	43	5		0.9	72	12		0.9	236	29
300	0.7	21	2	1000	0.7	83	17	5000	0.7	227	23
	0.8	29	<1		0.8	107	8		0.8	249	29
	0.9	51	5		0.9	112	28		0.9	283	24

4.3 Results for DIMACS benchmark graphs

To test the performance of VSA approach, further we have tested the proposed algorithm on benchmark graphs with known results, they have been extracted from DIMACS [8] challenge suite. That suite structured from the perspective of finding maximum cliques, so we considered the benchmark graphs as $\overline{G}$. We compare the heuristic performance with implementation of the algorithms SQUEEZE [4], KLS [12], OCH [1] and the results were shown in the Table 3.

The first three columns reports the type of the instances such as name, cardinality and density of the instances; the fourth gives the best results obtained in the challenge, the fifth, sixth and seventh gives the maximum size of the cliques found by corresponding algorithms. Eighth column reports the optimality achieved by proposed algorithm, in which * indicates the instances where proposed algorithm fail to reach the optimality, mostly in *MANN* type of instances. Table 3 shows that proposed algorithm could find the optimal solution for most of the DIMACS benchmark graphs i.e., out of 79 instances tested the proposed algorithm reaches the optimum value for 73 instances.

Table 3
Simulation results for the DIMACS benchmark instances

$\overline{G}$	|V|	Density	Optimum	SQUEEZE	KLS	OCH	Proposed VSA		
			α (G)	α (G)	α (G)	α (G)	α (G)	Time (s)	Success (%)
brock200_1	200	0.745	21	-	19	-	21	<1	100
brock200_2	200	0.496	12	12	10	12	12	<1	100
brock200_3	200	0.605	15	15	13	-	15	<1	100
brock200_4	200	0.658	17	17	14	17	17	<1	100
brock400_1	400	0.748	27	-	20	27	27	<1	100
brock400_2	400	0.749	29	-	23	29	29	<1	100
brock400_3	400	0.748	31	-	23	31	31	<1	100
brock400_4	400	0.749	33	-	23	33	33	<1	100
brock800_1	800	0.649	23	-	23	20	23	2	100
brock800_2	800	0.651	24	-	24	24	24	2	100
brock800_3	800	0.649	25	-	25	25	25	5	100
brock800_4	800	0.65	26	-	26	23	26	4	100
C125.9	125	0.898	34	34	-	34	34	<1	100
C250.9	250	0.899	44	-	-	44	44	<1	100
C500.9	500	0.9	≥57	-	-	53	57	6	100
C1000.9	1000	0.901	≥68	-	-	68	68	13	100
C2000.5	2000	0.5	≥16	-	-	16	16	18	100
C2000.9	2000	0.9	≥77	-	-	77	77	26	100
C4000.5	4000	0.5	≥18	-	-	-	18	30	100
c-fat200-1	200	0.077	12	12	12	-	12	<1	100
c-fat200-2	200	0.163	24	24	24	-	24	<1	100
c-fat200-5	200	0.426	58	-	-	-	56*	<1	96

Continued on next page

$\overline{G}$	|V|	Density	Optimum	SQUEEZE	KLS	OCH	Proposed VSA		
			α (G)	α (G)	α (G)	α (G)	α (G)	Time (s)	Success (%)
c-fat500-1	500	0.036	14	14	14	-	14	<1	100
c-fat500-2	500	0.073	26	26	26	-	26	<1	100
c-fat500-5	500	0.186	54	-	52	-	54	<1	100
DSJC500.5	500	0.5	≥13	13	13	13	13	13	100
DSJC1000.5	1000	0.5	≥15	15	15	15	15	20	100
gen200_p0.9_44	200	0.9	44	-	-	44	44	<1	100
gen200_p0.9_55	200	0.9	55	-	-	55	55	<1	100
gen400_p0.9_55	400	0.9	55	-	-	53	55	6	100
gen400_p0.9_65	400	0.9	65	-	-	65	65	9	100
gen400_p0.9_75	400	0.9	75	-	-	75	75	8	100
Hamming6-2	64	0.905	32	30	30	32	32	<1	100
Hamming6-4	64	0.349	4	4	4	4	4	<1	100
Hamming8-2	256	0.969	128	-	-	-	128	3	100
Hamming8-4	256	0.639	16	16	16	16	16	2	100
Hamming10-2	1024	0.99	512	512	512	512	512	9	100
Hamming10-4	1024	0.829	40	-	-	40	40	23	100
Johnson8-2-4	28	0.556	4	4	4	4	4	<1	100
Johnson8-4-4	70	0.768	14	14	14	14	14	<1	100
Johnson16-2-4	120	0.765	8	7	8	8	8	<1	100
Johnson32-2-4	496	0.879	16	-	-	15	16	6	100
keller4	171	0.649	11	11	8	11	11	<1	100
keller5	776	0.751	27	-	16	27	27	10	100
keller6	3361	0.818	≥58	-	-	58	54*	25	91
MANN_a9	45	0.927	16	16	16	16	16	<1	100
MANN_a27	378	0.99	126	-	117	120	125*	12	99
MANN_a45	1035	0.996	345	-	-	338	343*	33	99
MANN_a81	3321	0.999	≥1093	-	-	1093	1084*	43	98
p_hat300-1	300	0.244	8	8	8	8	8	<1	100
p_hat300-2	300	0.489	25	25	25	25	25	<1	100
p_hat300-3	300	0.744	36	36	36	36	36	<1	100
p_hat500-1	500	0.253	9	9	9	9	9	2	100
p_hat500-2	500	0.505	36	36	36	36	36	5	100
p_hat500-3	500	0.752	50	-	47	47	50	3	100
p_hat700-1	700	0.249	11	11	7	11	11	<1	100
p_hat700-2	700	0.498	44	-	44	43	44	15	100
p_hat700-3	700	0.748	62	-	59	60	61*	18	98
p_hat1000-1	1000	0.245	10	10	10	10	10	8	100
p_hat1000-2	1000	0.49	46	-	44	45	46	23	100
p_hat1000-3	1000	0.744	66	-	62	63	65*	30	98
p_hat1500-1	1500	0.253	12	12	10	12	12	23	100
p_hat1500-1	1500	0.506	≥64l	52	64	64	65	26	100
p_hat1500-1	1500	0.754	≥94	-	91	91	94	24	100

Continued on next page

$\overline{G}$	|V|	Density	Optimum	SQUEEZE	KLS	OCH	Proposed VSA		
			α (G)	α (G)	α (G)	α (G)	α (G)	Time (s)	Success (%)
san200-0.7.1	200	0.7	30	30	15	30	30	<1	100
san200-0.7.2	200	0.7	18	18	12	18	18	<1	100
san200-0.9.1	200	0.9	70	-	45	65	70	5	100
san200-0.9.2	200	0.9	60	-	39	57	60	17	100
san200-0.9.3	200	0.9	44	44	31	44	44	23	100
san400-0.5.1	400	0.5	13	13	7	13	13	6	100
san400-0.7.1	400	0.7	40	-	20	40	40	<1	100
san400-0.7.2	400	0.7	30	30	15	30	30	<1	100
san400-0.7.3	400	0.7	22	-	12	22	22	19	100
san400-0.9.1	400	0.9	100	-	50	96	100	8	100
san1000	1000	0.502	10	10	8	10	10	<1	100
sanr200-0.7	200	0.697	18	18	16	18	18	<1	100
sanr200-0.9	200	0.898	42	-	41	42	42	<1	100
sanr400-0.5	400	0.501	13	13	13	13	13	<1	100
sanr400-0.7	400	0.7	21	-	21	21	21	<1	100

Since we know the optimal solution value for each instance we tested, we can measure the quality of the solution derived by an algorithm by computing ratio between them. That is, we define the quality measure ratio as value/optimum, where value is the value of a solution found by an algorithm and optimum is the optimal solution value. We note that smaller the ratio indicates that the performance of an algorithm is guaranteed one. In Table 4 we sum up the information concerning the ratios.

Table 4
Averages and standard deviations of the ratio values

Algorithm	Min	Average	Max	Std. dev
VSA	1.00	1.06	1.18	0.06
OCH	1.00	1.26	1.45	0.13
KLS	1.15	1.40	1.70	0.17
SQUEEZE	2.80	3.75	4.94	1.21

4.4 Results for G(n, m) random graphs

In this experiment the parameter set opted like moderate scale problems, that is V varied from 50 to 1000. Here we used the G(n, m) graph model to generate the random graphs. All of the heuristics implemented in the previous experiment were examined in this experiment. For most of the test instances the optimal solutions are unknown, we obtained the relative performance of the other heuristics with the VSA by calculating the percentage of deviation of other heuristics from the VSA. These results are shown in the figure 5 where the major axis represents the 20 test instances and for each test instances error rate of other heuristics with VSA were plotted as points and for each algorithm their points are linked by a line. With these figures we show that, for the set instances we used, the VSA produced better solutions than other heuristics compared and the other heuristics get higher deviation from VSA when the size of the problem increases.

5. Concluding Remarks

A new VSA for MIS of graphs using vertex cover has been proposed and its effectiveness has been shown by simulation experiments. The terminology support of a vertex introduced in the new model, with that, the new model can find the maximum Independent Set effectively. Experimental result shows that this approach greatly reduce the execution time and in addition, the simulation results show that the new VSA can yield better solutions than SQUEEZE, KLS and OCH heuristics found in the literature. At the same time, our approach gives best solutions for DIMACS benchmark graph instances and also for random graphs. The proposed algorithm has led to give near optimal solutions for most of the test instances where we know the optimal solutions. Furthermore attractiveness of this heuristic is its outstanding performance for solving MIS.

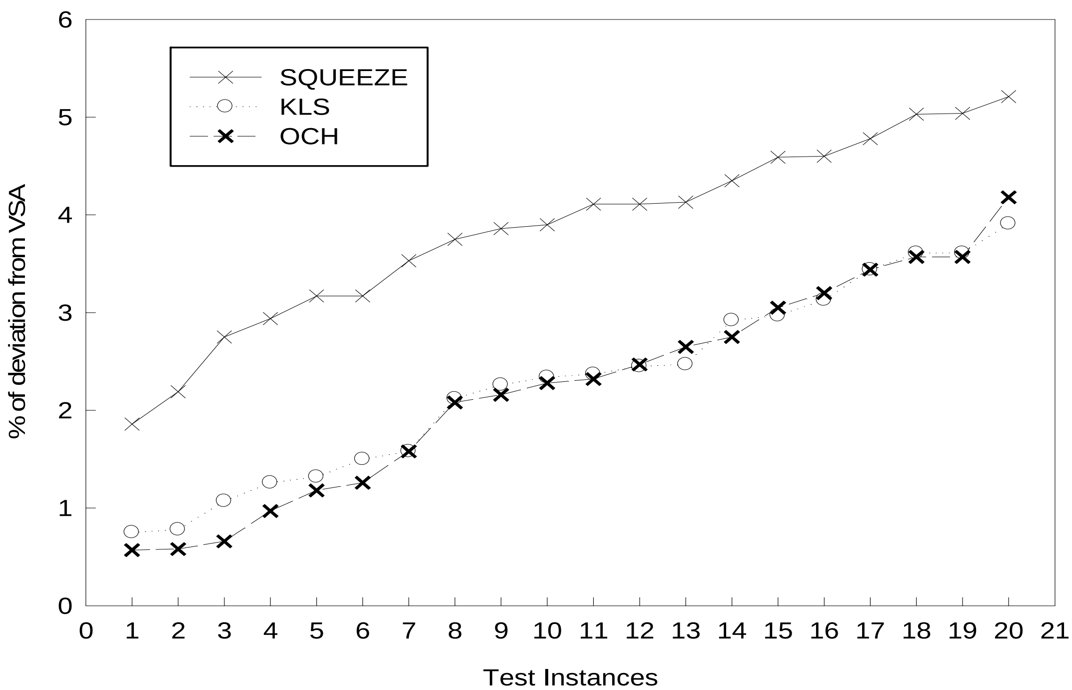


Figure 5: Error rate (%) of other heuristics with VSA in 3rd set of test instances

References:

[1] Aggarwal C., Orlin J. B & Tai R. P, "Optimized cross cover for the independent set problem," Operations Research, (1997), Vol. 45, pp. 226-234.

[2] Bollobas. B: Random graphs (2nd Ed.). Cambridge, UK: Cambridge University press (2001).

[3] Bollobas B, Erdös P., "Cliques in Random Graphs", mathematical Proceedings of the Cambridge Philosophical Society, (1976), Vol.80, pp. 419-427.

[4] Bourjolly J. M., Gill P., Laporte G & Mercure H, " An exact quadratic 0-1 algorithm for the stable set problem," in cliques, coloring and satisfiability, D. S. Johnson and M. A. Trick (Eds.). American Mathematical Society Providence, RI. 1996, pp. 53-73.

[5] Busygin S., Butenko S & Pardolos P. M, "A heuristic for the maximum independent set problembased on optimization of a quadratic over a sphere," Jounal of combinatorial optimization, (2002), vol. 6, pp. 287-297.

[6] Carraghan R., & Pardolos PM. "An exact algorithm for the maximum clique problem", Operations Research Letters, (1990), vol. 9, pp. 375-382.

[7] De Vries S & Vohra R "Combinatorial auctions: a survey" INFORMS Journal on Computing (2003), vol. 15, pp. 284-309.

[8] DIMACS clique benchmarks. Benchmark instances made available by electronic transfer at dimacs.rutgers.edu, Rutgers Univ., Piscataway. NJ. (1993).

[9] Etzion T & Östergård P. R. J., "Greedy and heuristic algorithms for codes and colorings," IEEE Transactions on Information Theory, (1998), vol. 44, pp. 382-388.

[10] Garey. M. R, Johnson. D. S: Computers and Intractability: A Guide to the theory NP – completeness. San Francisco: Freeman (1979).

[11] Gibbons L., Hearn D & Pardolos P, "A continuous based heuristic for the maximum clique problem" in cliques, coloring and satisfiability, D. S. Johnson and M. A. Trick (Eds.). American Mathematical Society Providence, RI. 1996, pp. 103-124.

[12] Katayama K, Hamamoto A, Narihisa H, "An effective local search for the maximum clique problem", Information Processing Letters, (2005), vol. 95, pp. 503-511.

[13] Matula D., “On the complete subgraph of a random graph”, Combinatory mathematics and its Applications, (1970), pp. 356-369.

[14] Östergård P R J., “A fast algorithm for the maximum clique problem”, Discrete Applied Mathematics, (2002), vol. 120, pp.197-207.

[15] Pardolos P., & Xue J. “The maximum clique problem”, Journal of Global optimization, (1994), vol.4, pp. 301-328.

[16] Pevzner, P.A. & Sze, S-H., Combinatorial approaches to finding subtle signals in DNA sequences, in proceedings of Eighth International Conference on intelligent systems for Molecular Biology, AAAI Press, (2000), pp.269-278.

[17] Pullan W, “Optimisation of unweighted/weighted maximum independent sets and minimum vertex covers” Discrete Optimization, (2009), vol. 6, pp. 214-219.