International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:7, No:5, 2013

A New Effective Local Search Heuristic for the Maximum clique problem

S. Balaji

Abstract—An edge based local search algorithm, called ELS, is proposed for the maximum clique problem (MCP), a well-known combinatorial optimization problem. ELS is a two phased local search method effectively £nds the near optimal solutions for the MCP. A parameter 'support' of vertices de£ned in the ELS greatly reduces the more number of random selections among vertices and also the number of iterations and running times. Computational results on BHOSLIB and DIMACS benchmark graphs indicate that ELS is capable of achieving state-of-the-art-performance for the maximum clique with reasonable average running times.

Keywords—Maximum clique, local search, heuristic, NP-complete.

I. Introduction

LOCAL search (or local improvement) is a practical tool and a common technique for £nding near-optimal solutions in reasonable time for combinatorial optimization problems. In many cases, local search can be incorporated into more sophisticated methods called meta-heuristics, in order to obtain more high-quality solutions. The basic idea of local search is start from a feasible solution x and repeatedly replace x with better x' which is selected from neighborhood of x de£ned as the set of neighbor solutions that can be reached by making slight modi£cations to x. If no better solutions can be found in its neighborhood, local search immediately stops and returns as £nal the best solution found during search.

The concept of local search was £rst applied by Lin and Kernighan to the traveling salesman problem (TSP) in 1973 [1] and graph partitioning problem (GPP) in 1970 [2]. The basic concept is to search a portion of the large neighborhood within a reasonable amount of computation time. In the early 2000, for TSP and GPP, the variable depth search based heuristics have been incorporated into several metaheuristic frameworks, such as iterated local search [3,4] and evolutionary algorithm [5,6]. Generally, the performance of metaheuristics embedded with local search is remarkably effective for the hard problems TSP and GPP. For some other hard problems effective local search algorithms have been proposed. For the generalized assignment problem, Yagiura et al. [7] suggested an algorithm. For the unconstrained binary quadratic programming problem (UBQP), Merz and Katayama [8] proposed a memetic algorithm with the variant VDS-based local search and reported that the memetic algorithm is highly effective.

More recently K. Katayama et al. [9] proposed a local search algorithm inspired by VDS for the maximum clique problem (MCP) and they claimed that their algorithm capable of £nding better average solutions than compared metaheurstics. Pullan [10,24] proposed a phased local search algorithm for both MCP and weighted MCP and he claimed that it achieves state-of-the-art performance for those problems. Judging from these contributions, we can expect that metaheuristics embedded with local search heuristics to offer promising approaches to other hard problems, such as the minimum vertex cover problem and its associated decision problems.

Such an embedding into metaheuristic frame works would not be possible without developing backbone of local search for the maximum clique problem. In this paper an edge based local search proposed for the MCP. It is referred as ELS. Edges considered in the proposed algorithm are whole edge set of a graph, edges incident on a particular vertex of a graph and the set of all edges of an induced subgraph of G. ELS ef£ciently iterates searches for the best neighbor solution with the help of these edge based condition until better one is found.

To show the effectiveness of ELS for the MCP, ELS is repeatedly applied for each of several well known DIMACS benchmark graphs [11]. Based on extensive computational experiments, it is worthwhile to note that ELS is simple, capable of £nding better average solutions than those of state-of-the-art metaheuristics, in particular KLS [9], on a broad range of widely studied benchmark instances and hence represent an improvement in heuristic MCP solving algorithms. For most graphs, this approach is comparable to the best available metaheuristic PLS [10] that is based on vertex penalties.

II. Maximum Clique Problem

Let $G = (V, E)$ be an arbitrary undirected simple graph where V is the set of n vertices and $E \subseteq V \times V$ is the set of edges in G. For a subset $S \subseteq V$, let $G[S] = (S, E \cap S \times S)$ be a subgraph induced by S. A graph $G = (V, E)$ is complete if all its vertices are pairwise adjacent, i.e., $\forall v_i, v_j \in V$ with $v_i \neq v_j$, $\{v_i, v_j\} \in E$. A Clique C is a subset of V such that the induced subgraph $G[C]$ is complete. The cardinality of C is the number of vertices contained in C, denoted by $|C|$. The objective of maximum clique problem (MCP) is to £nd a clique of maximum cardinality in G. The MCP and MVC problems are related in that a graph G has a maximum clique of size k if and only if the complemented graph $\overline{G}$ has a minimum vertex cover of size n-k.

The MCP is a prominent combinatorial optimization problem with many applications. For example mobile networks, computer vision, cluster analysis, coding theory, tiling, fault diagnosis, biological analysis. More recently, applications in bioinformatics have become important [12,13].

S. Balaji is with the Department of Mathematics, SASTRA University, Thanjavur, TN, 613401 INDIA e-mail: (balaji_maths@yahoo.com).

The MCP is NP-hard and the associated decision problem is NP-complete [14]; furthermore, it is inapproximable in the sense that no deterministic polynomial-time algorithm can £nd cliques of size $n^{1-\epsilon}$ for any $\epsilon > 0$, unless $NP = ZPP$ [15]. The best polynomial-time approximation algorithm for maximum clique problem achieves an approximation ratio of $O(n/(logn)^2)$ [16].

To solve the MCP exactly, several exact methods such as branch and bound algorithm have been proposed [18,19], but their effectiveness and applicability are limited to relatively very small (or) sparse graphs. Therefore, large and hard instances of MCP are typically solved using heuristic approaches, in particular, greedy construction algorithms, stochastic local search algorithms such as simulated annealing [17], genetic algorithms [20], dynamic local search[23] and phased local search [24]. Other recent heuristics include a trust region technique [21], neural network [22], sampling technique combined with parameterized k-vertex cover algorithm [25], evolutionary formulaion [26] and more surrogate constraint heuristics [27]. These have been proposed for the MCP or strongly related problems, i.e., the maximum independent set problem and the minimum vertex cover problem.

From the recent literature on MCP algorithm, it seems that, most algorithms have been constructed by taking into account of some troublesome parameters and vertices with maximum degree or minimum degree were added into a feasible solution. These selection process yields more number of random selections (due to tie in maximum or minimum degree) and it indeed yields more number of combinations of feasible solution. Therefore time taken by an algorithm to get a near optimal solution from these more number of feasible solutions becomes very high. In order to avoid the more number of random selections as much as possible and to make a heuristic better a new parameter called support of a vertex is de£ned and implemented in our previous research [29].Based on our previous work, a novel edge based local search algorithm is proposed for MCP. This edge based local search algorithm re£nes the procedure with the modi£ed de£nition of the parameter support, in order to increase its strength, produce best solution for large set of instances. The modi£ed de£nition is given by adding degree of vertex with its s(v) value. i.e.

For each $v \in V$, $support$ of a vertex is de£ned by
$$s(v) = d(v) + \Sigma_{u \in N(v)} d_G(u)$$

The quantity $\Sigma_{u \in N(v)} d_G(u)$ is the sum of the degree of vertices which are adjacent to v.

It is worthy to note that the selection of vertices with maximum $support$ value or minimum $support$ value into a feasible solution decidedly reduce the number of random selections and the number of trials and also the execution time to get a near optimal solution [30].

In this paper we will look at some different approach to solve the maximum clique problem based on the relation between maximum clique and minimum vertex cover problem. i.e., MCP is approximated by the edge based local search approach proposed for the MVC. Edges considered in the proposed algorithm are whole edge set of a graph, edges incident on a particular vertex of a graph and the set of all edges of an induced subgraph of G. This local search method ef£ciently iterates searches for the best neighbour solution with the help of these edge based condition until better one is found.

To show the effectiveness of the proposed method for the MCP, it is repeatedly applied for each of several well known DIMACS benchmark graphs . Based on the extensive computational experiments, it is worthwhile to note that VSA is simple, capable of £nding better average solutions than those of state-of-the-art metaheuristics, in particular KLS [9], on a broad range of widely studied benchmark instances and hence represent an improvement in heuristic MCP solving algorithms. For most graphs, this approach is comparable to the best available metaheuristic PLS [10] that is based on vertex penalties.

III. Edge based local search algorithm (ELS)

Notations: $\overline{E}$ - edge set of a graph $\overline{G}(V, \overline{E})$;
$\overline{E'}(v)$- set of all edges incident with v in a given graph $\overline{G}$;
$\overline{E_S}(v)$ - set of all edges incident with v in a induced subgraph $\overline{G}[S]$;
V_c - minimum vertex cover of a graph G.
V_{wc} - minimum weighted vertex cover of a graph G.

A. Algorithm

The Pseudocode of ELS as follows:
Input: $G(V, E)$ with adjacency matrix $A = (a_{ij})$
Generate: Complemented graph $\overline{G}(V, \overline{E})$ with an adjacency matrix $B = (b_i j)$
Output: Max. Clique $C(G) = V - V_c$
1. $V_c \leftarrow \phi$
2. $\forall v \in V$ and $\forall e \in \overline{E}$
3. ***while*** $\overline{E} \neq \phi$ ***do***
4. $d(v) = |N(v)|$ and $s(v) = d(v) + \Sigma_{u \in N(v)} d_G(u)$
5. $u \leftarrow max_{v \in V} s(v)$ if multiple vertices with same maximum $s(v)$ is found then select one vertex randomly among them.
6. $D \leftarrow D \cup \{u\}$
7. $\overline{E} \leftarrow \overline{E} - \overline{E'}(u)$
8. ***end while***
9. $(V, \overline{E}, D)$ /*Local search procedure*/
10. ***repeat*** $\forall v \in \overline{G}[D]$, update $d(v)$ and $s(v)$
11. $w \leftarrow max_{v \in D} s(v)$, apply the condition as in step 5, for selecting a vertex having minimum $s(v)$
12. ***if*** $\overline{E_D(w)} \subseteq \cup_{v \in D - \{w\}} \overline{E}(v)$
13. ***if*** $\forall v \in D - \{w\}, \overline{E} \subseteq \overline{E'}(v)$
14. ***then*** $V_c \leftarrow D - \{w\}; D \leftarrow D - \{w\}$
15. ***else if*** $\forall v \in D - \{N(w) \cap D\}, \overline{E} \subseteq \overline{E'}(v)$
16. ***then*** $V_c \leftarrow D - \{w\}; D \leftarrow D - \{w\}$
17. ***else*** $V_c \leftarrow V_c \cup \{w\}; D \leftarrow D - \{w\}$
18. ***else***
19. $V_c \leftarrow V_c \cup \{w\}; D \leftarrow D - \{w\}$
20. ***until*** $D \neq \phi$
21. ***end***
22. ***return*** updated V_c

The ELS operates as follows: The algorithm is working in two phases. First phase of the algorithm greedily select vertices in to a vertex cover, which is described in the lines 3-8 and the second phase of the algorithm is a local search (pruning) technique stated in the lines 9-21. In this technique an edge based local search procedure is applied and it re£nes the solution space, obtained in the £rst phase of the algorithm, in order to make it as a near optimal solution for the minimum vertex cover problem. The description of the algorithm follows:

For a given graph G, a complemented graph has been generated and this graph is being input to the ELS algorithm. After calculating $degree$ and $support$ of each vertices of $\overline{G}$ (line 4), to select a vertex in to a temporary vertex cover set D search starting in the line 5. This search space terminates when an edge set $\overline{E}$ becomes empty. In this search a vertex with maximum $support$ value added into the set D. Once a vertex is selected into D, the corresponding vertex and its incident edges removed from $\overline{G}$. i.e., in the line 7, the adjacency matrix of G updated by putting zero in to the row and column entries of the corresponding vertex which has been included in D. In selecting a vertex with maximum support value if multiple vertices have equivalent maximum support value, to add a vertex into the set D, a random selection is executed among them.

The second phase of the algorithm is a pruning technique (line 9-21). In this phase an edge based local search procedure is applied. This variant was inspired by the local search procedure used in the GRASP program for the maximum clique designed by Feo et al. [28]. Based on the above, a simple and effective procedure with minimum number of iterations is implemented in the local search. i.e., an edge based local search procedure is applied to determine whether it is possible to remove any of the vertices from D and replace them with one or no vertices while still remaining a minimum vertex cover.

In order to implement the above idea, $degree$ and $support$ values are updated for all the vertices in the subgraph induced by D. Among these vertices, a vertex with minimum support value is selected. If one or more vertices have the same criteria, a random selection is made among them. Then add or drop move adopted to achieve a near optimal solution in the local search. An add or drop moves based on the following cases:

case 1: For a vertex with minimum support value if its edges in the induced subgraph $\overline{G}[D]$ is a subset of union of all edges of the induced subgraph $\overline{G}[D]$ (D is a vertex set updated by excluding the minimum $support$ value vertex) then add or drop moves proceeded by the following three sub cases 1(a), 1(b) and 1(c). If the case 1 fails, the corresponding minimum support value vertex will be added in to the £nal MVC set V_c.

case 1(a): If the edge set $\overline{E}$ of $\overline{G}$ is a subset of set of all edges of $\overline{G}[D]$ where D is a vertex set updated by excluding the minimum support value vertex then the corresponding minimum support value vertex is dropped from the £nal MVC set V_c. If this condition fails, search move follows *case 1(b)*.

case 1(b): If the set of all edges $\overline{E}$ of $\overline{G}$ is a subset of set of all edges of $\overline{G}[D]$, where D is a vertex set updated by removing a vertex set $\{N(min.\ support\ value\ vertex) \cap D\}$ from itself, then the corresponding minimum support value vertex dropped from the £nal MVC set V_c.

case 1(c): If both the above two conditions 1(a) and 1(b) fails then simply add the corresponding minimum $support$ value vertex in to the £nal MVC set V_c.

In all the above main and sub cases after adding a vertex in the £nal MVC set V_c or dropping a vertex from the £nal MVC set V_c, the corresponding vertex removed from the temporary vertex cover set D. These add or drop moves repeated until the temporary vertex cover set D is non empty. The £nal step (line 23) of second phase of the algorithm returns a £nal MVC set V_c. Finally the maximum clique C of a graph G is extracted from the MVC set V_c of a graph $\overline{G}$ by $C = V - V_c$ and $\omega = n - |V_c|$.

A vertex with maximum support value in a complemented graph (less support in the original graph) is selected into a vertex cover of a complemented graph and exceedingly which is not included in the clique of the original graph. As a consequence a vertex with maximum support value in the original graph, which in turn adjacent with more number of vertices and forms maximum sized complete subgraph, has been added into the clique of a given graph. In such a way the ELS £nds the maximum clique of a graph.

IV. Experimental Results and Analysis

In order to evaluate the performance of VSA extensive computational experiments were carried out on following two sets of benchmark instances.

A. BHOSLIB benchmark

Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring) (BHOSLIB)[1]. The maximum independent set / minimum vertex cover benchmark instances are directly transformed from forced satis£able SAT benchmarks, with the set of vertices and the set of edges respectively corresponding to the set of variables and the set of binary clauses in SAT instances. The benchmark clique instances are the complements of above mentioned graph instances and these instances range in size from less than 500 vertices and 83500 edges to greater than 1500 vertices and 7400000 edges.

B. DIMACS benchmarks

These benchmarks were constructed using the maximum clique instances from the second DIMACS Implementation Challenge [11] which has been used extensively for benchmarking purposes in the recent literature on maximum clique algorithm. The 80 DIMACS maximum clique instances were generated from problems in coding theory, fault diagnosis problems, Keller's conjuncture on tilings using hypercubes and the Steiner triple problem, in addition to randomly generated graphs and graphs where the maximum clique has been 'hidden' by incorporating low-degree vertices. These problems

[1]http://www.nlsde.buaa.edu.cn/kexu/benchmarks/graph-bencmarks.htm

range in size from less than 50 vertices and 1000 edges to greater than 3300 vertices and 500000 edges.

All experiments for this study were performed on a Dell Vostro 1400 workstation with Intel Pentium Core2 Duo 1.6 GHz CPU and 1 GB of RAM. When executing the required user times, for DIMACS benchmark instances are 0.24 CPU seconds for r300.5, 1.02 CPU seconds for r400.5 and 5.32 CPU seconds for r500.5. In the performed experiments of ELS, described local search is repeatedly applied with different feasible solutions. i.e., in a single trial of local search method of ELS (line 9) is repeatedly executed up to $|D|$ times where $D \subseteq V$ is a temporary vertex cover which is obtained during the greedy search of £rst phase of the ELS.

TABLE I
ELS PERFORMANCE ON BHOSLIB BENCHMARKS

Instance	$\omega(G)$	avg	worst	CPU(s)	S
frb30-15-1	30	30	30	0.04	100
frb30-15-2	30	30	30	0.09	100
frb30-15-3	30	30	30	0.18	100
frb30-15-4	30	30	30	0.15	100
frb30-15-5	30	30	30	0.26	100
frb35-17-1	35	35	35	2.58	100
frb35-17-2	35	35	35	2.79	100
frb35-17-3	35	35	35	2.62	100
frb35-17-4	35	35	35	2.84	100
frb35-17-5	35	35	35	2.92	100
frb40-19-1	40	40	40	3.59	100
frb40-19-2	40	39.86	39	3.47	87
frb40-19-3	40	40	40	3.73	100
frb40-19-4	40	39.67	39	3.92	82
frb40-19-5	40	40	40	4.52	100
frb45-21-1	45	45	45	4.56	100
frb45-21-2	45	44.89	44	5.32	89
frb45-21-3	45	45	45	6.41	100
frb45-21-4	45	45	45	8.43	100
frb45-21-5	45	45	45	9.17	100

C. BHOSLIB benchmark results

To evaluate the performance of ELS on the BHOSLIB benchmark instances, 100 independent trials were performed for each instance using different feasible solution of temporary vertex cover set D (line 9 of ELS). The results from these experiments are displayed in TABLE I and TABLE II. The ELS performance results (averaged over 100 independent trials) are shown for the complete set of 40 BHOSLIB benchmark instances. For each instance, maximum clique size is given by the 2 digits immediately following 'frb' in the instance name; '$w(G)$' gives the maximum clique size, 'avg' gives the average maximum clique size and 'worst' gives the minimum clique size for the 100 ELS trials; 'CPU(s)' is the run-time in CPU seconds, averaged over all successful runs, for each instance; 'S' gives the number of successful trials (from a total of 100) in which the optimal maximum clique size was located. It is to be noted that ELS reaches best known solutions with a success rate of 100 over all 100 trials per instance for 20 of the 40 instances and £nds the best known solution for remaining instances in some of the least trials. Moreover ELS 100 independent trials were performed on the newly introduced benchmark instance frb100-40-1 of 4000 vertices, the ELS found the clique size of 96 instead of 100. As can be seen, the results for ELS are encouraging if not an improvement on these results.

TABLE II
ELS PERFORMANCE ON BHOSLIB BENCHMARKS

Instance	$\omega(G)$	avg	worst	CPU(s)	S
frb50-23-1	50	50	50	10.49	100
frb50-23-2	50	50	50	14.62	100
frb50-23-3	50	49.89	49	16.34	78
frb50-23-4	50	49.73	49	17.43	73
frb50-23-5	50	49.69	49	19.48	56
frb53-24-1	53	52.94	52	25.43	23
frb53-24-2	53	52.81	52	28.32	9
frb53-24-3	53	53	53	34.25	100
frb53-24-4	53	52.76	52	29.46	49
frb53-24-5	53	52.25	51	33.48	93
frb56-25-1	56	55.84	55	58.63	57
frb56-25-2	56	55.16	54	62.65	23
frb56-25-3	56	55.04	54	60.48	46
frb56-25-4	56	55.83	55	68.93	6
frb56-25-5	56	55.64	55	70.43	38
frb59-26-1	59	58.56	58	89.92	5
frb59-26-2	59	58.34	58	84.68	3
frb59-26-3	59	58.12	57	88.26	15
frb59-26-4	59	58.78	58	90.17	9
frb59-26-5	59	58.02	57	90.86	2

D. DIMACS benchmark results

To evaluate the performance of ELS on the DIMACS benchmark instances, 100 independent trials were performed for each instance using different feasible solution. The results from these experiments are displayed in TABLES III, IV, V and VI. The results in the tables indicate that the ELS reach the best known solution for all 80 DIMACS instances with a success rate of 100 over all 100 trials per instance for 66 of the 80 instances and for the remaining one of the least success rate of 3 over all 100 trials for the MANN_a81 instance.

For a comparative study on performance of ELS, published results of state-of-art-heuristics PLS [10], KLS [9], SAA [17] and Q-MS [30] are shown in the TABLE III, IV, V and VI. In the displayed tables '$\omega(G)$' is the best known maximum clique. The maximum clique size found by ELS, PLS [10], KLS [9], SAA [17] and Q-MS [30] are shown in the correspondingly labeled '$\omega(G)$' columns; 'avg' gives the average maximum clique size and 'worst' gives the minimum clique size for the 100 ELS trials; 'CPU(s)' is the run-time in CPU seconds of ELS, averaged over all successful runs, for each instance; 'SCP(s)' lists the scaled (to the reference computer used in this study) CPU time for the PLS algorithm; 'S' gives the number of successful trials (from a total of 100) in which the optimal maximum clique size was located. Entries of '$< \epsilon$' represents that the average CPU time is less than 0.005 seconds. Each of the heuristics PLS and KLS were

TABLE III
SIMULATION RESULTS FOR DIMACS BENCHMARK GRAPHS

G	Opt.	ELS	PLS	KLS	SAA	Q-MS
brock200_1	21	21	21	-	21	21
brock200_2	12	12	12	11	12	12
brock200_3	15	15	15	-	14	15
brock200_4	17	17	17	16	16	17
brock400_1	27	27	27	-	25	27
brock400_2	29	29	29	25	25	29
brock400_3	31	31	31	-	25	31
brock400_4	33	33	33	25	25	33
brock800_1	23	23	23	-	21	23
brock800_2	24	24	24	21	21	24
brock800_3	25	25	25	-	21	25
brock800_4	26	26	26	21	21	26
C125.9	34	34	34	34	34	34
C250.9	44	44	44	44	44	44
C500.9	57	57	57	57	57	55
C1000.9	68	68	68	68	68	64
C2000.5	16	16	16	16	16	16
C2000.9	77	77	77	77	74	72
C4000.5	18	18	18	18	17	17
c-fat200-1	12	12	12	-	12	12
c-fat200-2	24	24	24	-	24	24
c-fat200-5	58	58	58	-	58	58
c-fat500-1	14	14	14	-	14	14
c-fat500-2	26	26	26	-	26	26
c-fat500-5	54	54	54	-	64	64
c-fat500-10	126	126	126	-	126	126
DSJC500.5	13	13	13	13	13	13
DSJC1000.5	15	15	15	15	15	14
gen200_p0.9_44	44	44	44	44	44	42
gen200_p0.9_55	55	55	55	55	55	55
gen400_p0.9_55	55	55	55	53	55	51
gen400_p0.9_65	65	65	65	65	65	65
gen400_p0.9_75	75	75	75	75	75	75
Hamming6-2	32	32	32	-	32	32
Hamming6-4	4	4	4	-	4	4
Hamming8-2	128	128	128	-	128	128
Hamming8-4	16	16	16	16	16	16
Hamming10-2	512	512	512	-	512	512
Hamming10-4	40	40	40	40	40	36
Johnson8-2-4	4	4	4	-	4	4
Johnson8-4-4	14	14	14	-	14	14

TABLE IV
SIMULATION RESULTS FOR DIMACS BENCHMARK GRAPHS

G	Opt.	ELS	PLS	KLS	SAA	Q-MS
Johnson16-2-4	8	8	8	-	8	8
Johnson32-2-4	16	16	16	-	16	16
keller4	11	11	11	11	11	11
keller5	27	27	27	27	27	26
keller6	59	59	59	59	51	53
MANN_a9	16	16	16	-	16	16
MANN_a27	126	126	126	126	126	125
MANN_a45	345	345	344	345	334	342
MANN_a81	1100	1100	1098	1100	1080	1096
p_hat300-1	8	8	8	8	8	8
p_hat300-2	25	25	25	25	25	25
p_hat300-3	36	36	36	36	36	35
p_hat500-1	9	9	9	-	9	9
p_hat500-2	36	36	36	-	36	36
p_hat500-3	50	50	50	-	50	48
p_hat700-1	11	11	11	11	11	11
p_hat700-2	44	44	44	44	44	44
p_hat700-3	62	62	62	62	62	62
p_hat1000-1	10	10	10	-	10	10
p_hat1000-2	46	46	46	-	46	45
p_hat1000-3	68	68	68	-	68	65
p_hat1500-1	12	12	12	12	12	12
p_hat1500-2	65	65	65	65	65	64
p_hat1500-3	94	94	94	94	94	91
san200-0.7.1	30	30	30	-	17	30
san200-0.7.2	18	18	18	-	15	18
san200-0.9.1	70	70	70	-	61	70
san200-0.9.2	60	60	60	-	60	60
san200-0.9.3	44	44	44	-	44	40
san400-0.5.1	13	13	13	-	7	13
san400-0.7.1	40	40	40	-	21	40
san400-0.7.2	30	30	30	-	16	30
san400-0.7.3	22	22	22	-	17	18
san400-0.9.1	100	100	100	-	57	100
san1000	10	10	10	-	8	15
sanr200-0.7	18	18	18	-	18	18
sanr200-0.9	42	42	42	-	42	41
sanr400-0.5	13	13	13	-	13	13
sanr400-0.7	21	21	21	-	21	20

carried out 100 trials for each instance whereas SAA was carried out 20 trials for each instance. The running times of PLS is shown on the TABLE V and TABLE VI but the running times of KLS, SAA and Q-MS are not shown on those tables because SAA and Q-MS user times were not reported in [17, 30] and published results of PLS claimed that it outperformed the KLS both in solution quality and running times. Computer used in the ELS is 5 times faster than the one used in the PLS (2.2 GHz Pentium IV), the average running times of PLS shown in the table are all adjusted with the scaling factor 0.92. From these comparative results shown in the TABLE III, IV, V and VI, ELS achieves excellent performance on the 80 DIMACS benchmark instances.

ELS seems to be competitive with PLS in terms of the running times of average solution results for the 80 DIMACS instances and no information of previous solution obtained by the local search has been preserved and used in the local search steps of ELS. From this point of view, ELS is simplest method.

V. CONCLUSION

Local search algorithms are known to be highly effective for several combinatorial optimization problems. It is worthwhile considering a new local search based on a new parameter 'support' of vertex and edges of a graph for solving the hard problems like maximum clique problem. Based on this motivation a new heuristic algorithm ELS is developed and

TABLE V
SIMULATION RESULTS FOR DIMACS BENCHMARK GRAPHS

G	ELS				PLS
	avg	worst	CPU(s)	S	SCP(s)
brock200_1	21	21	$< \epsilon$	100	0.08
brock200_2	12	12	$< \epsilon$	100	0.05
brock200_3	15	15	$< \epsilon$	100	0.06
brock200_4	17	17	$< \epsilon$	100	0.07
brock400_1	27	27	$< \epsilon$	100	2.46
brock400_2	29	29	$< \epsilon$	100	0.74
brock400_3	31	31	$< \epsilon$	100	1.54
brock400_4	33	33	$< \epsilon$	100	3.67
brock800_1	23	23	1.29	100	56.49
brock800_2	24	24	1.53	100	15.73
brock800_3	25	25	2.59	100	21.91
brock800_4	25.75	25	2.63	100	8.88
C125.9	34	34	0.08	98	0.08
C250.9	44	44	0.09	100	0.06
C500.9	57	57	0.06	100	0.23
C1000.9	67.56	66	1.02	100	0.90
C2000.5	16	16	1.83	97	4.38
C2000.9	76.64	75	26.22	96	97.99
C4000.5	17.52	17	29.82	94	189.00
c-fat200-1	12	12	$< \epsilon$	100	0.08
c-fat200-2	24	24	$< \epsilon$	100	0.12
c-fat200-5	58	58	$< \epsilon$	100	0.19
c-fat500-1	14	14	$< \epsilon$	100	0.25
c-fat500-2	26	26	$< \epsilon$	100	0.37
c-fat500-5	54	54	$< \epsilon$	100	0.86
c-fat500-10	125.89	123	1.83	98	0.47
DSJC500.5	13	13	$< \epsilon$	100	2.01
DSJC1000.5	14.84	14	0.12	100	5.21
gen200_p0.9_44	44	44	$< \epsilon$	100	1.21
gen200_p0.9_55	55	55	$< \epsilon$	100	0.08
gen400_p0.9_55	55	55	1.01	100	4.03
gen400_p0.9_65	65	65	1.05	100	5.01
gen400_p0.9_75	75	75	$< \epsilon$	100	7.03
Hamming6-2	32	32	$< \epsilon$	100	0.06
Hamming6-4	4	4	1.12	100	0.04
Hamming8-2	128	128	$< \epsilon$	100	1.51
Hamming8-4	16	16	$< \epsilon$	100	2.61
Hamming10-2	511.02	507	0.01	100	100
Hamming10-4	39.37	38	$< \epsilon$	100	100

TABLE VI
SIMULATION RESULTS FOR DIMACS BENCHMARK GRAPHS

G	ELS				PLS
	avg	worst	CPU(s)	S	SCP(s)
Johnson8-2-4	4	4	$< \epsilon$	100	5.89
Johnson8-4-4	14	14	$< \epsilon$	100	8.32
Johnson16-2-4	8	8	$< \epsilon$	100	0.08
Johnson32-2-4	16	16	$< \epsilon$	100	0.12
keller4	11	11	$< \epsilon$	100	0.32
keller5	27	27	$< \epsilon$	100	0.45
keller6	59	59	$< \epsilon$	100	170.69
MANN_a9	16	16	$< \epsilon$	100	2.61
MANN_a27	125.69	123	3.86	65	3.43
MANN_a45	344.45	339	32.97	43	276.43
MANN_a81	1098.67	1091	45.62	3	264.32
p_hat300-1	8	8	$< \epsilon$	100	0.34
p_hat300-2	25	25	$< \epsilon$	100	0.24
p_hat300-3	36	36	$< \epsilon$	100	0.62
p_hat500-1	9	9	$< \epsilon$	100	0.76
p_hat500-2	36	36	$< \epsilon$	100	0.23
p_hat500-3	50	50	$< \epsilon$	100	0.55
p_hat700-1	11	11	$< \epsilon$	100	0.82
p_hat700-2	44	44	1.05	100	3.21
p_hat700-3	62	62	2.32	100	7.61
p_hat1000-1	9.56	9	$< \epsilon$	95	2.54
p_hat1000-2	45.87	45	$< \epsilon$	93	3.62
p_hat1000-3	67.98	66	$< \epsilon$	88	2.82
p_hat1500-1	11.85	11	$< \epsilon$	92	7.81
p_hat1500-2	64.74	63	$< \epsilon$	82	9.48
p_hat1500-3	93.02	92	$< \epsilon$	76	1.01
san200-0.7.1	30	30	$< \epsilon$	100	1.23
san200-0.7.2	18	18	$< \epsilon$	100	4.02
san200-0.9.1	69.46	67	1.15	100	2.56
san200-0.9.2	59.56	58	0.09	100	1.12
san200-0.9.3	43.98	43	1.01	100	2.23
san400-0.5.1	13	13	$< \epsilon$	100	5.13
san400-0.7.1	40	40	$< \epsilon$	100	2.07
san400-0.7.2	30	30	$< \epsilon$	100	3.09
san400-0.7.3	22	22	$< \epsilon$	100	5.10
san400-0.9.1	99. 17	97	0.09	100	2.67
san1000	10	10	$< \epsilon$	100	16.55
sanr200-0.7	18	18	$< \epsilon$	100	0.15
sanr200-0.9	42	42	$< \epsilon$	100	0.23
sanr400-0.5	13	13	$< \epsilon$	100	4.12
sanr400-0.7	21	21	$< \epsilon$	100	3.01

implemented. It is worthy to note that the parameter de£ned in the heuristic not only greatly reduce the random selections among vertices in the feasible solution but also the number of iterations and running times in getting the near optimal solution. Although that alone gives ELS a remarkable advantage, an even more signi£cant advantage is that its two phased procedure. The feasible solution obtained by the greedy approach in the £rst phase re£ned in the second phase for getting near optimal solution with the help of an edge based procedure. This search technique yields highly effective cliques even in a short running time.

The computational results on the DIMACS benchmark graphs demonstrated that ELS was outperformed recent meta-heuristic components such as k-opt local search, simulated annealing and trust region technique algorithm in terms of its obtainable solutions for all the graphs. Also, it was competitive with the best suitable heuristic based on stochastic reactive dynamic local search for many graphs.

The excellent performance of ELS, reported that the underlying edge based local search has substantial potential for solving the maximum clique problem and the relevant hard problems.

REFERENCES

[1] S. Lin, B. W. Kernighan, An effective heuristic algorithm for traveling salesman problem, Oper. Res. 21 (1973) 498-516.

[2] B. W. Kernighan, S. Lin, An ef£cient heuristic procedure for partitioning graphs, Bell System Techn. J. 49 (1970) 291-307.
[3] K. Katayama, H. Narihisa, Iterated local search approach using genetic transformation to the traveling salesman problem, Proceedings of the Genetic and Evolutionary Computation Conference, (1999) 321-328
[4] D. Applegate, W. Cook, A. Rohe, Chained Lin-Kernighan for large traveling salesman problems, Informs Journal of Computing, 15(1) (2003) 82-92.
[5] P. Merz, B. Freisleben, Memetic algorithms for the traveling salesman problem, Complex Systems, 13(4) (2001) 297-345.
[6] P. Merz, B. Freisleben, Fitness landscapes, memetic algorithms and greedy operators for graph partitioning, Evolutionary Computing, 8(1) (2000) 61-91.
[7] M. Yaguira, T. Yamaguchi, T. Ibaraki, A variable depth search algorithm for the generalized assignment problem, in S. Voss, S. Martello, I. H. Osman, C. Roucairol(Eds.), Meta-Heuristics: Advance and Trends in Local Search Paradigms for Optimization, Kluwer, Dordrecht, (1999) 459-471.
[8] P. Merz, K. Katayama, Memetic algorithms for the unconstrained binary quadratic programming problem, Biosystems, 78(1-3) (2004) 99-118.
[9] K. Katayama, A. Hamamoto, H. Nirihisa, An effective local search for the maximum clique problem, Information Processing Letters, 95 (2005) 503-511.
[10] W. Pullan, Phased local search for the maximum clique problem, J. Comb. Optim. 12 (2006) 303-323.
[11] D. S. Johnson, M. A. Trick (Eds.): Cliques, Coloring and sats£ability, Second DIMACS Implementation Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 26, American Mathematical Society, Providence, RI, 1996.
[12] P. A. Pevzner, S.-H. Sze, Combinatorial approaches to £nding subtle signals in DNA sequences, in proceedings of Eighth International Conference on intelligent systems for Molecular Biology, AIAA Press, New York, (2000) 269-278.
[13] Y. Ji, X. Xu, G. D. Stormo, A graph theoretical approach for predicting common RNA secondary structure motifs including pseudoknots in unaligned sequences. Bioinformatics 20(10) (2004) 1591-1602.
[14] M. R. Garey, D. S. Johnson: Computers and Intractability: A Guide to the theory NP - completeness. San Francisco: Freeman (1979).
[15] J. Hastad: Clique is hard to approximate within n1- , Acta Mathematica, 182 (1999) 105-142.
[16] R. Boppana, M. Halldorsson, Approximating maximum independent sets by excluding subgraphs. Bit 32 (1992) 180-196.
[17] X. Geng, J. Xu, J. Xiao and L. Pan: A simple simulated annealing algorithm for the Maximum Clique problem", 177 (2007) 5064-5071.
[18] E. Tomita, T. Kameda, An ef£cient branch-and-bound algorithm for £nding a maximum clique with computational experiments, J. Glob. Optim. 37 (2007) 95-111.
[19] P. R. J. Ostergard: A fast algorithm for the maximum clique problem, Discrete Applied Mathematics, 120 (2002) 197-207.
[20] E. Balas, W. Niehaus, Optimized crossover-Based Genetic Algorithm for the Maximum Cardinality and Maximum Weight Clique Problems, Journal of Heuristics, 4 (1998) 107-122.
[21] S. Busygin, A new trust region technique for the maximum weight clique problem, Discrete Applied Mathematics, 154 (2006) 2080-2096.
[22] G. -Marin, E. -Casermeiro, J. -Perez, Modelling competitive Hop£eld networks for the maximum clique problem, Computers & Operations Research, 30 (2003) 603-624.
[23] W. Pullan, H. H. Hoos, Dynamic local search for the maximum clique problem, Journal of Arti£cial Intelligence Research, 25 (2006) 159-185.
[24] W. Pullan, Approximating the maximum vertex/edge weighted clique using local search, J. Heuristics, 14 (2008) 117-134.
[25] P. J. Taillon, A new approach for solving the maximum clique problem, LNCS, Springer-Verlag Berlin Heidelberg, 4041 (2006) 279-290.
[26] V. C. Barbosa, C. D. Campos, A novel evolutionary formulation of the maximum independent set problem, J. Comb. Optim. 8 (2004) 419-437.
[27] B. Alidaee, G. Kochenberger, H. Wang, Simple and fast surrogate constraint heuristics for the maximum independent set problem, J. Heuristics, 14 (2008) 571-585.
[28] T. S. Feo, M. G. C. Resende, S. H. Smith, A greedy randomized adaptive search procedure for maximum independent set, Operations Research, 42(5) (1994) 860-978.
[29] S. Balaji, V. Swaminathan, K. Kannan: Approximating maximum weighted independent set using vertex Support, International Journal of Computational and Mathemtical Sciences1, 3(8) (2009) 406-411.
[30] S. Balaji, N. Revathi: An Ef£cient approach for the Optimization version of Maximum Weighted Clique Problem, WSEAS Transactions on Mathematics, 11(9) (2012) 773-783.

S. Balaji received his B.Sc and M.Sc degree in Mathematics from Manonmaniam Sundaranar University in 1999 and 2001. In 2004 he received his M.Phil degree in Mathematics from Madurai Kamaraj University, India. He received his Ph.D in Mathematics from SASTRA University, India in 2012. He is currently an Assistant Professor in the Department of Mathematics, SASTRA University, India. His research interest include elegant design of graph theoretical based approximation algorithms for graph and combinatorial optimization probelms and designing routing protocols for the adhoc wireless networks.