An Efficient approach for the Optimization version of Maximum Weighted Clique Problem

S. BALAJI
Department of Mathematics
SASTRA University
Thanjavur - 613 401
INDIA
balaji_maths@yahoo.com

N. REVATHI
Department of Mathematics
RVS Institutions
Coimbatore - 641 402
INDIA
reva17787@yahoo.co.in

Abstract: Given a graph and a weight function defined on the vertex set of a graph, the maximum weighted clique (MWC) problem calls for finding the number of vertices with maximum total weight and also any two of vertices are pairwise adjacent. In this paper, an edge based local search algorithm, called ELS, is proposed for the MWC, a well-known combinatorial optimization problem. ELS is a two phased local search method effectively finds the near optimal solutions for the MWC. A parameter '$support$' of vertices defined 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 DIMACS benchmark graphs indicate that ELS is capable of achieving state-of-the-art-performance for the maximum weighted clique with reasonable average running times.

Key–Words: Maximum weighted clique, local search, heuristic, NP-complete.

1 Introduction

Local search (or local improvement) is a practical tool and a common technique for finding 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 defined as the set of neighbor solutions that can be reached by making slight modifications to x. If no better solutions can be found in its neighborhood, local search immediately stops and returns as final the best solution found during search.

The concept of local search was first applied by Lin and Kernighan to the traveling salesman problem (TSP) in 1973 [20] and graph partitioning problem (GPP) in 1970 [19]. 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 [16, 1] and evolutionary algorithm [24, 23]. 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 [32] suggested an algorithm. For the unconstrained binary quadratic programming problem (UBQP), Merz and Katayama [25] 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 [17] proposed a local search algorithm inspired by VDS for the maximum clique problem (MCP) and they claimed that their algorithm capable of finding better average solutions than compared metaheurstics. Pullan [28, 29] 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 MWC. 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 efficiently iterates searches for the best neighbor solu-

tion with the help of these edge based condition until better one is found.

To show the effectiveness of ELS for the MWC, ELS is repeatedly applied for each of several well known DIMACS benchmark graphs [15]. Based on extensive computational experiments, it is worthwhile to note that ELS is simple, capable of finding better average solutions than those of state-of-the-art metaheuristics, on a broad range of widely studied benchmark instances and hence represent an improvement in heuristic MWC solving algorithms. For most graphs, this approach is comparable to the best available metaheuristic PLS [28] that is based on vertex penalties.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph, where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices and $E \subseteq V \times V$ (not in ordered pairs) is the set of edges with cardinality of $|V| = n$ and $|E| = m$ and the complement graph of $G(V, E)$ is the graph $\overline{G}(V, \overline{E})$, where $\overline{E} = \{v_i, v_j \in V, v_i \neq v_j \; and \; (v_i, v_j) \notin E\}$. Then we have the following basic definitions relative to this work:

Neighborhood of a vertex: For each $v \in V$, the neighborhood of v is defined by $N(v) = \{u \in V | u \; is \; adjacent \; to \; v\}$ and the closed neighbourhood of v is defined by $N[v] = \{v\} \cup N(v)$.

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 i.e., $d(v) = |N(v)|$.

Minimum Vertex Cover: A set $S \subseteq V$ is a minimum vertex cover of G if (i) for every edge $(u, v) \in E$, either $u \in S$ or $v \in S$ or both $u, v \in S$ and (ii) among all covers of E, S has the minimum cardinality. The cardinality of a minimum vertex cover of G is called vertex cover number of G and is denoted by $\nu(G)$.

Minimum weighted vertex cover (MWVC) problem is a generalization of minimum vertex cover (MVC) problem, in which a weight function $\omega : V \rightarrow R$ is associated with the vertex set V and one has to find a vertex cover with minimum weight.

Maximum Independent Set: 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 maximum independent set problem seeks to find an independent set with maximum cardinality. The size of the maximum independent set of G is called the stability number of G and is denoted by α.

Maximum Clique: A clique of G is a subset $C \subseteq V$ in which every pair of vertices are adjacent. A clique is called maximal if it is not a subset of another clique and a maximal clique with the highest cardinality is called maximum. The cardinality of maximum clique is denoted by ω. The objective of maximum clique problem is to find a clique of maximum possible order.

Maximum weighted clique (MWC) problem and *maximum weighted independent set* (MWIS) problem is a generalization of maximum clique problem (MCP) and maximum independent set (MIS) problem respectively, in which vertices have positive weight i.e., a weight function $\omega : V \rightarrow R$ is associated with the vertex set V and one has to find a clique with maximum weight and an independent set with maximum weight. There are some relations between the maximum clique maximum independent set and minimum vertex cover:

Property: Let $G = (V, E)$ be a graph and $\overline{G}$ is the complemented graph of G, then

- $\omega(G) = \alpha(\overline{G})$
- $\alpha(G) = n - \nu(G)$
- $\omega(G) = n - \nu(\overline{G})$

Remark: The relations identified above also hold for the maximum weighted clique, maximum weighted independent set and minimum weighted vertex cover. In addition, if W_T is the total weight of the vertices in G or $\overline{G}$ and W_C, W_S denotes the total weights in the maximum weighted clique in G, minimum weighted vertex cover in $\overline{G}$ then $W_T = W_C + W_S$.

3 Related Works

The MC problem is a classic one in computer science and in graph theory. It is related to many real-life problems which includes classification theory, coding theory, project selection, fault tolerance, signal transmission and computer vision [5, 31]. VLSI circuit design and more recently its application in bioinformatics [27, 30] have become important. Both maximum clique and maximum weighted clique problems are NP-hard [13]. Hence simple algorithms which yield acceptable solutions sufficiently fast are quite important for such related practical problems.

The MWC problem is computationally intractable even to approximate with certain absolute performance bounds [10, 12]. Very few numbers of algorithms has been proposed for the MWC. Xiutang Geng [14] proposed a simple simulated annealing algorithm for the MC and he reported with computational experiments that their algorithm outperformed recently proposed efficient simulated annealing algorithms. Balas and Niehaus [4] developed an optimized crossover based steady-state genetic algorithm; it takes two cliques and produces a single child.

The child is produced by finding a maximum weight clique in a subgraph induced by the union of vertex sets of two cliques through solving the maximum flow problems in the complement of a subgraph. Katayama et al. [17, 18] showed an effective heuristic algorithm k-opt local search for the MC and MWC based on variable depth search. They reported with computational experiments over DIMACS [15] benchmark graphs that their algorithm outperformed previously reported metaheuristic based on a simple local search. Busygin [9] proposed a heuristic based on trust region technique. In which he claimed with computational experiments that the proposed algorithm is exact on small graphs and efficient on DIMACS [15] graphs. Babel [2] proposed an efficient branch and bound procedure. This method uses upper and lower bounds, for the maximum weight clique that is computed by coloring the weighted graph. Östergárd [26] also developed a fast branch and bound based exact method. Bomze et al. [8, 7] proposed a method based on replicator dynamics. It uses the continuous formulation of maximum weight clique problem. Recently, Pullan [29]proposed a local search procedure for maximum weighted clique problem, which was supported by computational experiments. Other recent heuristics include complementary pivoting approach [22], branch and bound [33] and augmenting sequences [21]. These are all proposed for the MWC or its related problems like the maximum weighted independent set problem and the minimum weighted vertex cover problem.

4 Driven Parameter-Advantage

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 defined and implemented in our previous research [3]. In which for each $v \in V$, the parameter support of a vertex is defined by $support(v) = s(v) = \Sigma_{u \in N(v)} d_G(u)$ where the quantity $\Sigma_{u \in N(v)} d_G(u)$ is the sum of the degree of vertices which are adjacent to v. Based on this parameter, a greedy heuristic approach applied on the MCP. But that heuristic fails to reach good quality solution for large dense graphs of DIMACS namely $c - fat200 - 5$, $keller6$, $p\_hat700 - 3$, $p\_hat1000 - 3$ and all $MANN$ type of instances except $MANN\_a9$. In order to obtain good quality solution for large dense graphs, based on our previous work, a novel edge based local search algorithm is proposed for MCP. This edge based local search algorithm refines the procedure with the modified definition of the parameter support, in order to increase its strength, produce best solution for large set of instances. The modified definition is given by adding degree of vertex with its $s(v)$ value. i.e. For each $v \in V$, $support$ of a vertex is defined by

$$s(v) = d(v) + \Sigma_{u \in N(v)} d_G(u)$$

and an additional parameter $ratio$ of a vertex $r(v)$ of each $v \in V$ is defined by the relation

$$r(v) = \frac{s(v) \times d(v)}{w(v)}$$

because of a weight function $\omega : V \rightarrow R$ is associated for the vertex set V. It is worthy to note that the selection of vertices with maximum '$support$'value or minimum '$support$'value into the feasible solution decidedly reduce the number of random selection and the number of trials and also the execution time to get a near optimal solution.

4.1 Example

To illustrate how the $degree$ based selection and the new parameter $support$ based selection of vertices differs, Figure 1 gives an example for selection of a vertex based on maximum $support$ value of a vertex. In this figure vertices are numbered 1 to 5. When selecting vertices with maximum $degree$, there is a tie situation with the vertices 1, 2 and 4, each of them shares same $degree$ 2. But selecting vertices based on maximum $support$ value of a vertex selects the only one vertex 2 as maximum $support$ value vertex. i.e. there is no tie situation. In this way the new parameter reduces the number of random selection among vertices as much as possible and also the number of iterations to get a near optimal solution.

5 Edge based local search algorithm - ELS

In this section we will look at some different approach to solve the maximum clique problem based on the relation between maximum weighted clique and minimum weighted vertex cover problem. i.e., MWCP

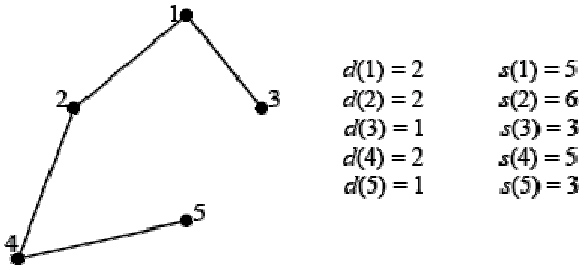

Figure 1: Selection of a vertex based on $support$ value

is approximated by the edge based local search approach proposed for the MWVC. 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 efficiently 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 the proposed method for the MWCP, 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 ELS is simple, capable of finding better average solutions than those of state-of-the-art metaheuristics, in particular KLS [17], on a broad range of widely studied benchmark instances and hence represent an improvement in heuristic MWCP solving algorithms. For most graphs, this approach is comparable to the best available metaheuristic PLS [29] that is based on vertex penalties.

The ELS algorithm is now described using following notations.

5.0.1 Notations:

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

5.1 The ELS Algorithm

Input: $G(V, E)$ with adjacency matrix $A = (a_{ij})$ and a weight function $\omega : V \rightarrow R$
Output: $|V_{wc}|$

1. $V_{wc} \leftarrow \phi; D \leftarrow \phi$
2. $\forall v \in V$ and $\forall e \in E$
3. ***while*** $E \neq \phi$ ***do***
4. $d(v) = |N(v)|$, $s(v) = d(v) + \Sigma_{u \in N(v)} d_G(u)$ and $r(v) = \frac{d(v) \times s(v)}{w(v)}$
5. $u \leftarrow max_{v \in V} r(v)$ if multiple vertices with same maximum $r(v)$ is found then select one vertex randomly among them.
6. $D \leftarrow D \cup \{u\}$
7. $E \leftarrow E - E^{'}(u)$
8. ***end while***
9. (V, E, D) /*Local search procedure*/
10. ***repeat*** $\forall v \in G[D]$, update $d(v)$, $s(v)$ and $r(v)$
11. $w \leftarrow max_{v \in D} r(v)$, apply the condition as in step 5, for selecting a vertex having minimum $r(v)$
12. ***if*** $E_D(w) \subseteq \cup_{v \in D - \{w\}} E(v)$
13. ***if*** $\forall v \in D - \{w\}, E \subseteq E^{'}(v)$
14. ***then*** $V_{wc} \leftarrow D - \{w\}; D \leftarrow D - \{w\}$
15. ***else if*** $\forall v \in D - \{N(w) \cap D\}, E \subseteq E^{'}(v)$
16. ***then*** $V_{wc} \leftarrow D - \{w\}; D \leftarrow D - \{w\}$
17. ***else*** $V_{wc} \leftarrow V_{wc} \cup \{w\}; D \leftarrow D - \{w\}$
18. ***else***
19. $V_{wc} \leftarrow V_{wc} \cup \{w\}; D \leftarrow D - \{w\}$
20. ***until*** $D = \phi$
21. ***end***
22. ***return*** updated V_{wc}

The ELS operates as follows: The algorithm is working in two phases. First phase of the algorithm greedily select vertices in to a weighted 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 refines the solution space, obtained in the first phase of the algorithm, in order to make it as a near optimal solution for the minimum weighted vertex cover problem. The description of the algorithm as follows:

For a given graph G, an adjacency has been generated and this graph is being input to the ELS. After calculating $degree, support$ and $ratio$ of each vertices (line 4), to select a vertex in to a temporary weighted vertex cover set D search starting in the line 5. This search space terminates when an edge set E becomes empty. In this search a vertex with maximum $ratio$ value added into the set D. Once a vertex is selected into D, the corresponding vertex and its incident edges removed from 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 $ratio$ value, if multiple vertices have equivalent maximum $ratio$ 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 weighted vertex cover.

In order to implement the above idea, $degree, support$ and $ratio$ values are updated for all the vertices in the subgraph induced by D. Among these vertices, a vertex with minimum $ratio$ 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 $ratio$ value if its edges in the induced subgraph $G[D]$ is a subset of union of all edges of the induced subgraph $G[D]$ (D is a vertex set updated by excluding the minimum $ratio$ 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 $ratio$ value vertex will be added in to the final MVC set V_{wc}.

Case 1(a): If the edge set E of G is a subset of set of all edges of $G[D]$ where D is a vertex set updated by excluding the minimum $ratio$ value vertex then the corresponding minimum $ratio$ value vertex is dropped from the final MWVC set V_{wc}. If this condition fails, search move follows *case 1(b)*.

Case 1(b): If the set of all edges E of G is a subset of set of all edges of $G[D]$, 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 final MWVC set V_{wc}.

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 final MWVC set V_{wc}.

In all the above main and sub cases after adding a vertex in the final MWVC set V_{wc} or dropping a vertex from the final MWVC set V_{wc}, 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 final step (line 23) of second phase of the algorithm returns a final MVC set V_{wc}.

5.2 Algorithm for maximum weighted clique Problem

1. **Input:** $G(V, E)$ with adjacency matrix $A = (a_{ij})$ and a weight function $\omega : V \rightarrow R$
2. Generate: Complemented graph $\overline{G}(V, \overline{E})$ with an adjacency matrix B = (b_{ij}) $(i = 1, 2, ..., n; j = 1, 2, ..., n)$
3. Apply steps 1 to 22 of ELS for $\overline{G}(V, \overline{E})$
4. **Output:** Maximum weighted Clique $C(G)$ = $V - V_{wc}$ where V_{wc} is the minimum weighted vertex cover of $\overline{G}$

A vertex with maximum $ratio$ value in a complemented graph (less $ratio$ in the original graph) is selected into a weighted vertex cover of a complemented graph and exceedingly which is not included in the weighted clique of the original graph. As a consequence a vertex with maximum $ratio$ 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 finds the maximum weighted clique of a graph.

6 Computational Complexity

To determine the time complexity of ELS, note that in the first phase, while loop executed atmost mn times. In the second phase of the algorithm, the process is repeated for all the vertices in the temporary vertex cover D. Since $D \subseteq V$, the number of vertices in D is $\leq n(= |V|)$. Moreover for each and every vertex in D, local search searching all its edges. At the most level the search space may move up to $m(= |E|)$ times. Therefore the second phase of the algorithm runs at most mn times. Hence the time complexity of the algorithm is $O(mn)$.

7 Experimental Results and Analysis

In order to evaluate the performance of ELS, for MWC, extensive computational experiments were carried out on random and benchmark instances identified below. All the procedures of ELS 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.

7.1 Random Graphs-$G(n, p)$ Model

This section outlines the $G(n, p)$ graph model, random graphs, used to assess the effectiveness of the proposed algorithm in the previous section.

The $G(n, p)$ model is also called Erdös Renyi random graph model [6], consists of graphs of n vertices

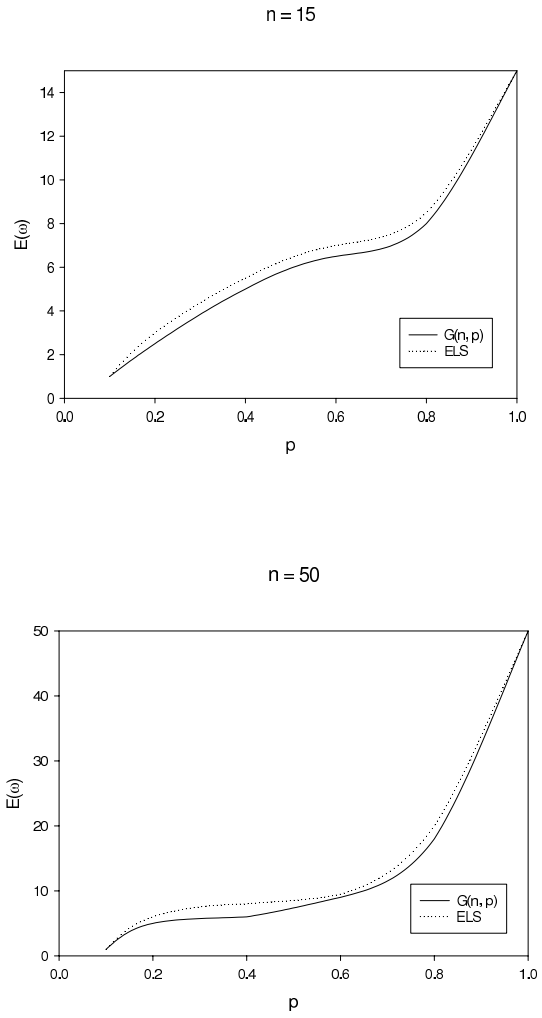

Figure 2: E(ω) for $n = 15, 50$ and for different p values

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.

7.1.1 Algorithm to generate $G(n,p)$ graphs

The pseudo code for generating $G(n,p)$ graphs as follows

1. initialize graph $G(V,E)$
2. for $i \leftarrow 1$ *to* n
3. for $j \leftarrow i+1$ *to* n
4. add edge (i,j) to E with probability p
5. return (G).

The expected number of edges of $G(n,p)$ graph is $pn(n-1)/2$ and expected degree is np. Graphs are generated for different p and n values.

7.2 DIMACS benchmarks

These benchmarks were constructed using the maximum clique instances from the second DIMACS Implementation Challenge [15] 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 tailings 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 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 first phase of the ELS.

7.3 Results on Random Graphs

For finding clique number, the proposed ELS tested on $G(n,p)$ random graphs [6] for various values of n and p. Since finding clique number is a computationally difficult problem, we had to limit ourselves to moderate values of n and/or p. It is well known that [6] the clique number of the $G(n,p)$ graph is, with probability converging to one, within a constant of $2log_{1/p}n - 2log_{1/p}log_{1/p}n$ when p is held fixed as n grows, called asymptotic formula for clique number. The obtained results on random graphs shown in the Figure 2 and Figure 3. In these figures $G(n,p)$ denotes the expected clique number of the corresponding $G(n,p)$ graph, which was obtained by the above asymptotic formula and ELS denotes the expected clique number $E(\omega)$ obtained by the proposed approach ELS and is approximated by averaged over 100 independent trials of each instance of $G(n,p)$ for different values of n and p. For small values of p, the ELS finds best solution than the result obtained by the asymptotic formula and ELS get coincides with the asymptotic formula values when the values of p are

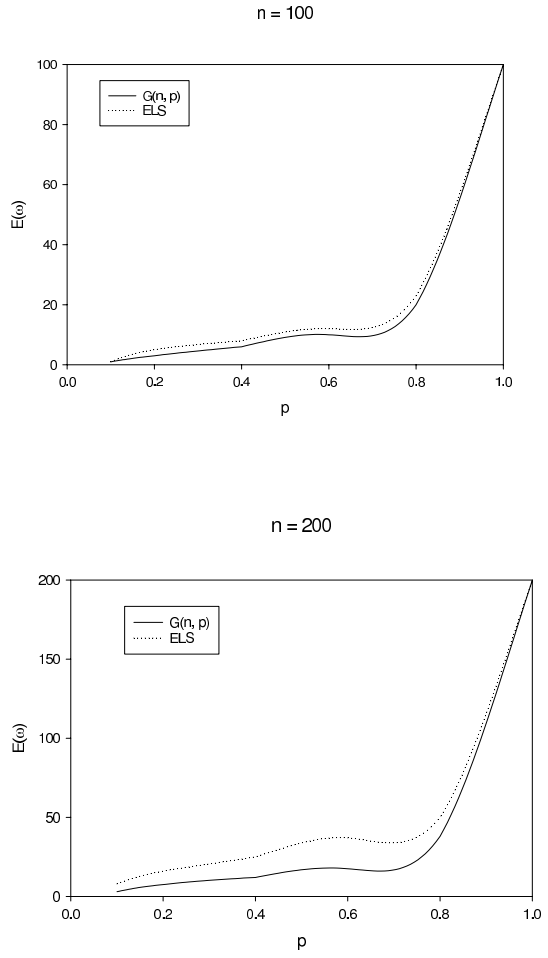

Figure 3: E(ω) for $n = 100, 200$ and for different p values

large. From these Figure 1 and Figure 2, we see that average clique number obtained by ELS matches with the analytical average of the clique number of the corresponding random graphs and also the proposed approach finds best solution for maximum clique even in the increasing values of n of $G(n, p)$ random graphs.

7.4 Results on Benchmark graphs

As there are no resulted benchmark set for the maximum weighted clique, to test the performance of ELS approach and also design easily reproducible experiments, the DIMACS benchmarks instances of maximum clique problem were used. Those DIMACS instances were converted into weighted instances by allocating w_i, for vertex v_i ($1 \leq i \leq n$), uniformly random in the range [1, 100]. i.e., the nodes of $G(V, E)$ allocated by the weights $(v_i \ mod \ 100) + 1$. The constant 100 in the weight calculation was determined after a number of experiments showed in the previous chapters that the generated problems appeared to be reasonably difficult for ELS. To evaluate the performance of ELS on the converted weighted DIMACS instances, 100 independent trials were performed for each instance using different feasible solution. The results from these experiments are displayed in Tables 1, 2, 3 and 4. In Table 1 and Table 2, the first three columns reports the name, size and density of the graphs and the fourth column $\omega(G)$ reports the cardinality of maximum clique without weight; For the maximum weighted clique of G, the fifth column W_C is the total weight in the maximum weighted clique of G. The remaining obtained results are displayed in Table 3 and Table 4. In which first column reports the name of the instances. For the maximum weighted clique of G, the second and third columns $\omega_v(G)$ reports the cardinality obtained by ELS and $A(C)$ is the average vertex weight; Time(s) in the fourth column is the run-time in CPU seconds of ELS, averaged over all successful runs, for each instance; and in the fifth column Δ reports the difference between the cardinality of maximum clique and maximum weighted clique instances obtains by ELS i.e. $\Delta = \omega(G) - \omega_v(G)$. When compared to MCP, adding vertex weights can increase the difficulty of the problem to ELS. i.e. as can be seen from Tables 3 and 4 and from the results reported in [3], the running time to find the maximum weighted clique by ELS is higher than to find the maximum clique in the unweighted instances and also from the Figure 4, the number of iterations required to find the maximum weighted clique for the $p.hat$ type of instances are larger than the unweighted case of $p.hat$ instances.

It is difficult to compare the experimental results of weighted cliques of ELS with other heuristics approaches not only because of non availability of results on a standard benchmark set (such as DIMACS for maximum cliques) in the literature but also differences in the respective experimental protocols, random number generators and run-time environments. Other than that of these conditions to check whether the ELS reaches the best solution for these maximum weighted clique instances, the experimental results reported in Bomze et al., [7], Babel [2] and Pullan [29] are taken for the comparative study. We have obtained some statistical quantities from the percentage of deviation of these heuristics with the proposed algorithm. i.e., for some of these instances of same condition and we compared the performance of ELS with other heuristics Babel [2], RD [7] and PLS [29]. The corresponding statistical values are shown in the Table 5. In the obtained results positive values tells us that ELS reaches the best solution, negative values represents the ELS fails to reach the optimum solution than the other heuristics compared and if the values are exactly equal to zero then ELS and compared heuristics

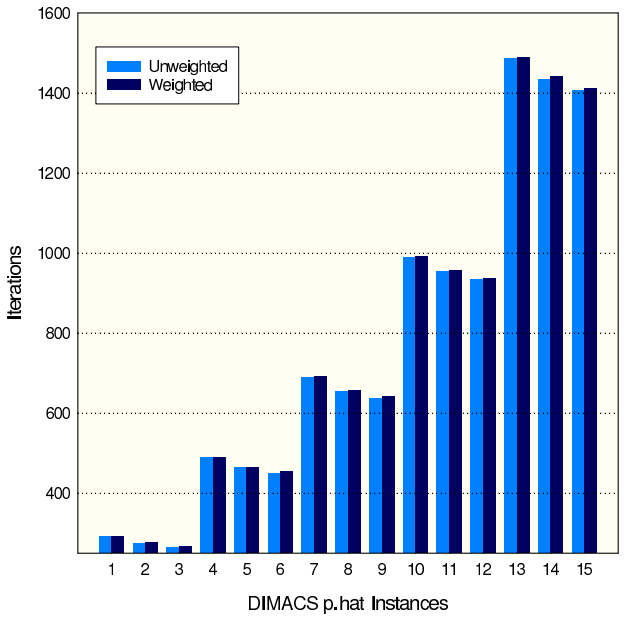

Figure 4: Number of iterations taken to find MC and MWC of DIMACS $p.hat$ instances by ELS

reaches the same solution. Hence from Table 5, we identified that Babel heuristic deviated highly and PLS gets low deviation from ELS. From these results we see that the quality of the solution delivered by ELS is much better than the other heuristics involved in this experiment and ELS obtained best known solution for the converted weighted DIMACS instances. It will be useful for future researchers to benchmark their results.

8 Conclusions

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 weight clique problem. Based on this motivation a new heuristic algorithm ELS is developed and implemented. It is worthy to note that the parameter defined 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 significant advantage is that its two phased procedure. The feasible solution obtained by the greedy approach in the first phase refined 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 excellent performance of ELS, reported that the underlying edge based local search has substantial potential for solving the maximum weight clique problem and the relevant hard problems.

Table 1: Simulation results for the weighted DIMACS instances

$G(V,E)$	V	Density	$\omega(G)$	W_C
brock400_1	400	0.748	27	211
brock400_2	400	0.749	29	210
brock400_3	400	0.748	31	237
brock400_4	400	0.749	33	237
brock800_1	800	0.649	23	163
brock800_2	800	0.651	24	163
brock800_3	800	0.649	25	165
brock800_4	800	0.65	26	172
c-fat200-1	200	0.077	12	86
c-fat200-2	200	0.163	24	156
c-fat200-5	200	0.426	58	384
c-fat500-1	500	0.036	14	96
c-fat500-2	500	0.073	26	186
c-fat500-5	500	0.186	54	423
Hamming6-2	64	0.905	32	211
Hamming6-4	64	0.349	4	29
Hamming8-2	256	0.969	128	718
Hamming8-4	256	0.639	16	102
Hamming10-2	1024	0.99	512	2906
Hamming10-4	1024	0.829	40	316
Johnson8-2-4	28	0.556	4	29
Johnson8-4-4	70	0.768	14	86
Johnson16-2-4	120	0.765	8	51
Johnson32-2-4	496	0.879	16	96

References:

[1] D. Applegate, W. Cook, A. Rohe, Chained Lin-Kernighan for large traveling salesman problems, *Informs Journal of Computing*, 15(1), 2003, 82–92.

Table 2: Simulation results for the weighted DIMACS instances

$G(V,E)$	V	Density	$\omega(G)$	W_C
MANN_a9	45	0.927	16	100
MANN_a27	378	0.99	126	786
MANN_a45	1035	0.996	345	2658
p_hat300-1	300	0.244	8	61
p_hat300-2	300	0.489	25	175
p_hat300-3	300	0.744	36	218
p_hat500-1	500	0.253	9	78
p_hat500-2	500	0.505	36	247
p_hat500-3	500	0.752	50	342
p_hat700-1	700	0.249	11	80
p_hat700-2	700	0.498	44	288
p_hat700-3	700	0.748	62	405
p_hat1000-1	1000	0.245	10	87
p_hat1000-2	1000	0.49	46	297
p_hat1000-3	1000	0.744	66	436
p_hat1500-1	1500	0.253	12	91
p_hat1500-2	1500	0.506	65	431
p_hat1500-3	1500	0.754	94	583
san200-0.7.1	200	0.7	30	219
san200-0.7.2	200	0.7	18	125
san200-0.9.1	200	0.9	70	483
san400-0.7.1	400	0.7	40	275
san400-0.7.2	400	0.7	30	207
san400-0.9.1	400	0.9	100	643
san1000	1000	0.502	10	95
sanr200-0.7	200	0.697	18	101
sanr200-0.9	200	0.898	42	257
sanr400-0.5	400	0.501	13	75
sanr400-0.7	400	0.7	21	143

Table 3: Simulation results for the weighted DIMACS instances

$G(V,E)$	$\omega_v(G)$	$A(C)$	Time(s)	Δ
brock400_1	23	9.17	<1	4
brock400_2	23	9.13	<1	6
brock400_3	27	8.77	<1	4
brock400_4	28	8.46	<1	5
brock800_1	18	9.05	5	5
brock800_2	19	8.57	3	5
brock800_3	19	8.68	7	6
brock800_4	20	8.60	6	6
c-fat200-1	12	7.16	<1	0
c-fat200-2	22	7.09	<1	2
c-fat200-5	56	6.86	<1	2
c-fat500-1	14	6.86	<1	0
c-fat500-2	24	7.75	<1	2
c-fat500-5	64	6.60	<1	10
Hamming6-2	31	6.80	<1	1
Hamming6-4	4	7.25	<1	0
Hamming8-2	126	5.69	4	2
Hamming8-4	16	6.37	6	0
Hamming10-2	512	5.67	13	0
Hamming10-4	40	7.90	28	0
Johnson8-2-4	4	7.25	<1	0
Johnson8-4-4	12	7.16	<1	2
Johnson16-2-4	8	6.37	<1	0
Johnson32-2-4	15	6.40	6	1

[2] L. Babel, A fast algorithm for the maximum weight clique problem, *Computing*, 52, 1994, pp. 31–38.

[3] S. Balaji, V. Swaminathan, K. Kannan, A Simple Algorithm for Maximum Clique and Matching Protein Structures, *International Journal of Combinatorial Optimization Problems and Informatics*, 1(2), 2010, pp. 2–11.

[4] E. Balas, W. Niehaus, Optimized crossover-Based Genetic Algorithm for the Maximum Cardinality and Maximum Weight Clique Problems, *Journal of Heuristics*, 4, 1998, pp. 107–122.

[5] E. Balus, C. Yu, Finding maximum clique in an arbitrary graph, *SIAM journal of computing*, 15(4), 1986, pp. 1054–1068.

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

[7] I. Bomze, M. Pelillo, V. Stix, Approximating the maximum weight clique using replicator dy-

Table 4: Simulation results for the weighted DIMACS instances

$G(V,E)$	$\omega_v(G)$	$A(C)$	Time(s)	Δ
MANN_a9	14	7.14	<1	2
MANN_a27	121	6.50	18	5
MANN_a45	338	7.86	40	7
p_hat300-1	8	7.63	<1	0
p_hat300-2	23	7.60	<1	2
p_hat300-3	33	6.60	<1	3
p_hat500-1	9	8.66	4	0
p_hat500-2	35	7.05	6	1
p_hat500-3	46	7.43	7	4
p_hat700-1	9	8.88	3	2
p_hat700-2	43	6.69	18	1
p_hat700-3	59	6.86	22	3
p_hat1000-1	9	9.66	8	1
p_hat1000-2	43	6.90	26	3
p_hat1000-3	63	6.92	35	3
p_hat1500-1	10	9.10	24	2
p_hat1500-2	58	7.43	30	7
p_hat1500-3	88	6.62	31	6
san200-0.7.1	27	8.11	<1	3
san200-0.7.2	16	7.82	<1	8
san200-0.9.1	67	7.21	5	3
san400-0.7.1	38	7.23	4	2
san400-0.7.2	29	7.14	3	1
san400-0.9.1	96	6.70	8	4
san1000	10	9.50	5	0
sanr200-0.7	18	5.61	3	0
sanr200-0.9	41	6.27	4	1
sanr400-0.5	13	5.77	2	0
sanr400-0.7	20	7.15	<1	1

Table 5: Statistical values of % of deviation

Alg.	Min.	Median	Average	Max.	S.D
RD	0.23	12.88	12.95	32.5	8.58
Babel	0	7.69	10.85	33.83	10.09
PLS	-1.04	1.81	3.37	19.75	4.47

namics, *IEEE Transactions Neural Network*, 11, 2000.

[8] I. Bomze, M. Budinich, M. Pelillo, Rossi, Annealed Replication: A new heuristic for the maximum clique problem, *Discrete Applied Mathematics*, 121, 2002, pp. 27–49.

[9] S. Busygin, A new trust region technique for the maximum weight clique problem, *Discrete Applied Mathematics*, 154, 2006, pp. 2080–2096.

[10] P. Crescenzi, C. Fiorini, R. Silvestri, A note on the approximation of the max. clique problem, *Information Processing Letters*, 40(1), 1991, pp. 1–5.

[11] 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, pp. 860–978.

[12] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, M. Szegedy, Approximating clique is almost NP-complete, *Proceedings of the 32nd IEEE Annual symposium on Foundations of computer science*, San Juan, Paerto Rico, 1991, pp. 2–12.

[13] M. R. Garey, D. S. Johnson, Computers and Intractability: *A Guide to the theory NP - completeness*, San Francisco: Freeman 1979.

[14] X. Geng, J. Xu, J. Xiao, L. Pan, A simple simulated annealing algorithm for the Maximum Clique problem, *Information Sciences*, 177, 2007, pp. 5064–5071.

[15] D. S. Johnson, M. A. Trick (Eds.), Cliques, Coloring and satsfiability, Second DIMACS Implementation Challenge, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 26, American Mathematical Society, Providence, RI, 1996.

[16] 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, pp. 321–328.

[17] K. Katayama, A. Hamamoto, H. Nirihisa, An effective local search for the maximum clique problem, *Information Processing Letters*, 95, 2005, pp. 503–511.

[18] K. Katayama et., al, Local search for Maximum weight clique problem, *Transactions of the Institute of Electronics*, Information and Communication Engineers, J89-A 8, 2006, pp. 649–661.

[19] B. W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, *Bell System Techn. J.* 49, 1970, pp. 291–307.

[20] S. Lin, B. W. Kernighan, An effective heuristic algorithm for traveling salesman problem, *Oper. Res.* 21, 1973, pp. 498–516.

[21] C. Mannino and E. Stefanutti, An augmentation algorithm for the Maximum weighted stable set problem, *Computational Optimization and Applications*, 14, 1999, pp. 367–381.

[22] A. Massaro, M. Pelillo, I. M. Bomze, A Complementary pivoting approach to the maximum weight clique problem, *SIAM J. Optim.* 12(4), 2002, pp. 928–948.

[23] P. Merz, B. Freisleben, Fitness landscapes, memetic algorithms and greedy operators for graph partitioning, *Evolutionary Computing*, 8(1), 2000, pp. 61–91.

[24] P. Merz, B. Freisleben, Memetic algorithms for the traveling salesman problem, *Complex Systems*, 13(4), 2001, pp. 297–345.

[25] P. Merz, K. Katayama, Memetic algorithms for the unconstrained binary quadratic programming problem, *Biosystems*, 78(1-3), 2004, 99–118.

[26] P. R. J. Östergård, A New Algorithm for the Maximum Weight Clique problem, *Nordic Journal of Computing*, 8, 2001, pp. 424–436.

[27] P. A. Pevzner, S. H. Sze, Combinatorial approaches to finding subtle signals in DNA sequences, *Proceedings of Eighth International Conference on intelligent systems for Molecular Biology*, AAAI Press, 2000, pp. 269–278.

[28] W. Pullan, Phased local search for the maximum clique problem, *J. Comb. Optim.* 12, 2006, pp. 303–323.

[29] W. Pullan, Approximating the maximum vertex/edge weighted clique using local search, *J. Heuristics*, 14, 2008, pp. 117–134.

[30] J. Razmara, S. B. Deris, A Novel Text Modeling Approach for Structural Comparison and Alignment of Biomolecules, *WSEAS Transactions on Computers*, 9, 2010, pp. 675–685.

[31] Y. Sun, X. Gu, J. Qian, Construction of Virtual Backbone on Growth-Bounded Graph with Variable Transmission Range, *WSEAS Transactions on Computers*, 7, 2008, pp. 32–38.

[32] 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, pp. 459–471.

[33] K. Yamaguchi, S. Masuda, A new exact algorithm for the maximum weight clique problem, *The 23rd international conference on circuits/systems, Computers and communications*, 2008, pp. 317–320.