

# A Hybrid Evolutionary Approach to Maximum Weight Clique Problem

Alok Singh and Ashok Kumar Gupta

J. K. Institute of Applied Physics and Technology,  
Faculty of Science, University of Allahabad,  
Allahabad – 211002, India  
{alok, akjkiapt}@jk institute.org

**Abstract:** In this paper we propose a hybrid evolutionary approach combining steady-state genetic algorithm and a greedy heuristic for the maximum weight clique problem. The genetic algorithm generates cliques that are then extended into maximum weight clique by the heuristic. Tests on a variety of benchmark problem instances demonstrate the effectiveness of our approach.

**Keywords:** Combinatorial optimization, greedy heuristic, maximum weight clique, steady-state genetic algorithm.

## I. Introduction

A clique of an undirected graph is a subgraph in which each pair of distinct vertices is connected by an edge. A clique is called maximal if it is not contained in any other clique. A maximum clique of a graph is a maximal clique having the maximum number of vertices. Maximum weight clique problem is a generalization of maximum clique problem in which vertices have positive weights and one has to find the clique with maximum weight. Both maximum clique and maximum weight clique problems are NP-Hard [1]. Due to this reason exact algorithm for these problems are guaranteed to return a solution only in time that increases exponentially with number of vertices in the graph and this makes exact algorithms for these problems infeasible even in case of moderately large problem instances. Moreover, these problems are also hard to approximate. It was shown in [2] that unless  $NP = ZPP$  no polynomial algorithm can approximate the clique within a factor of  $n/2^{(\log n)^{(1-\epsilon)}}$  for any  $\epsilon > 0$  where  $n$  is the number of nodes in the graph.

Since the early 1970s many heuristic and exact algorithms have been proposed to solve the maximum weight clique problem. Nemhauser and Trauter [3] formulated the problem as an integer linear program and used an implicit enumeration algorithm to solve it. Extending their work on maximum clique problem, Loukakis and Tsouros [4] developed a recursive backtracking algorithm for the maximum weight clique problem. Carraghan and Pardalos

[5] proposed a partial enumerative algorithm. Pardalos and Desai [6] represented the problem as an unconstrained quadratic 0-1 program and proposed a branch and bound technique to solve it but it was slower than the algorithm of Carraghan and Pardalos [5]. Building on the previous work on unweighted case, Balas and Xue [7] developed an efficient branch and bound procedure using minimum weighted coloring of triangulated graph. Later Balas and Xue [8] developed another branch and bound method that used as an upper bounding procedure, the heuristic that they developed for the weighted fractional coloring problem. Balas and Niehaus [9] developed an optimized crossover based steady-state genetic algorithm. The optimized crossover takes two cliques and produces a single child. The child is produced by finding the maximum weight clique in the subgraph induced by the union of vertex sets of two cliques through solving the maximum flow problem in the complement of the subgraph. Babel [10] 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. Östergård [11] also developed a fast branch and bound based exact method. Bomze et al. [12] proposed a method based on replicator dynamics. It uses the continuous formulation of maximum weight clique problem using Motzkin-Strauss theorem [13]. Massaro et al. [14] proposed PBH, a complementary pivoting based heuristic that also uses the continuous formulation. Busygin [15] developed QUALEX-MS, a new trust region based heuristic, using the continuous formulation. QUALEX-MS and PBH are currently the best heuristics known to solve the maximum weight clique problem.

Besides being theoretically interesting maximum weight clique problem has many practical applications in various fields such as computer vision, pattern recognition, robotics where weighted graphs are used to represent high level pictorial information (see [16] and [17]). A good survey of algorithms, applications and complexity issues of this

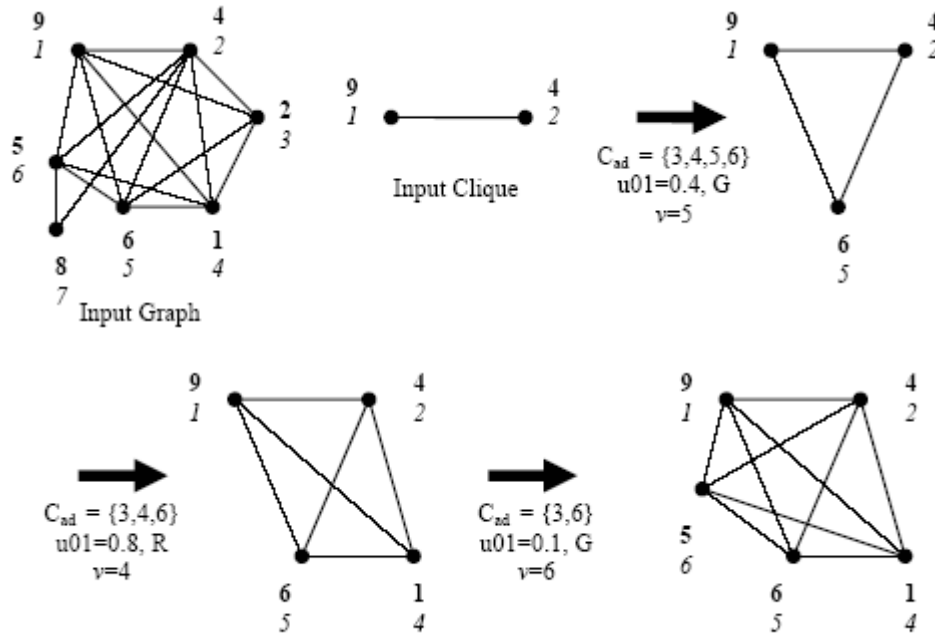


Figure 1. Illustrating the heuristic by a simple example

problem can be found in Bomze et al. [18].

In this paper we present a heuristic based steady-state genetic algorithm (WT-HSS) for the maximum weight clique problem. The steady-state genetic algorithm generates cliques which are then extended into maximal weight clique by the heuristic. The present work extends our approach [19] for the maximum clique problem to maximum weight clique problem.

We have compared our algorithm with QUALEX-MS [15] and PBH [14]. We have also tested our algorithm on benchmark instances of Östergard [18].

In this paper we assume that the input graph in which we have to find the maximum weight clique is  $G = (V, E, w)$ , where  $V$  denotes the set of vertices,  $E$  denotes the set of edges and  $w: V \rightarrow \mathbb{R}^+$  is the weight function associated with vertices of  $V$ .

This paper is organized as follows: Section 2 describes our heuristic. Section 3 discusses the steady-state genetic algorithm as used in our approach. In section 4, we compare WT-HSS with QUALEX-MS and PBH. Section 5 outlines some conclusions

## II. The Heuristic

The heuristic proposed here extends a clique to maximal weight clique. It begins with finding the set of vertices adjacent to current clique, then it semi randomly (with probability  $p_{ad}$ ) selects the vertex, maximizing the product of its weight and local degree (degree in the subgraph induced by the set of adjacent vertices) to be included in the current clique. Ties are broken arbitrarily. The whole procedure is repeated again and again until the set of adjacent vertices becomes empty. The pseudo-code of our heuristic is given

below where  $u01$  is a uniform variate,  $C_c$  is the current clique,  $C_{ad}$  is the set of vertices adjacent to  $C_c$  and  $random(C_{ad})$  returns a vertex in  $C_{ad}$  randomly:

```

 $C_{ad} \leftarrow \{ v : v \notin C_c \wedge (u,v) \in E \forall u \in C_c \}$ 
while (  $|C_{ad}| > 0$  ) {
  if (  $u01 \leq p_{ad}$  )
     $v \leftarrow arg \max_{u \in C_{ad}} ( w(u) \times | \{ t : t \in C_{ad} \wedge (u,t) \in E \} | )$ 
  else
     $v \leftarrow random(C_{ad})$ 
   $C_c \leftarrow C_c \cup \{ v \}$ 
  recalculate  $C_{ad}$ 
}
return  $C_c$ 

```

In all our computational experiments we have used  $p_{ad} = 0.6$ . This value is chosen empirically. Figure 1 explains the heuristic with the help of an example. In this figure the number in italics indicates vertex index whereas number in bold above vertex index indicates weight of corresponding vertex. Suppose the input clique  $C_c = \{1, 2\}$  so set of vertices adjacent to  $C_c$ ,  $C_{ad} = \{3,4,5,6\}$ . Suppose in the first iteration we get the value of  $u01$  to be 0.4, therefore the next vertex to be included in the clique is selected from  $C_{ad}$  according to greedy criterion (As indicated by G just below the arrow in Figure 1). The vertex 5 have the maximum value (=18) of the product of weight and local degree therefore it is included in the clique. After this we compute the  $C_{ad}$  again. Hence after the first iteration  $C_c = \{1,2,5\}$  and  $C_{ad} = \{3,4,6\}$ . Now the second iteration of the heuristic begins. Suppose in this iteration value of  $u01$  turns out to be

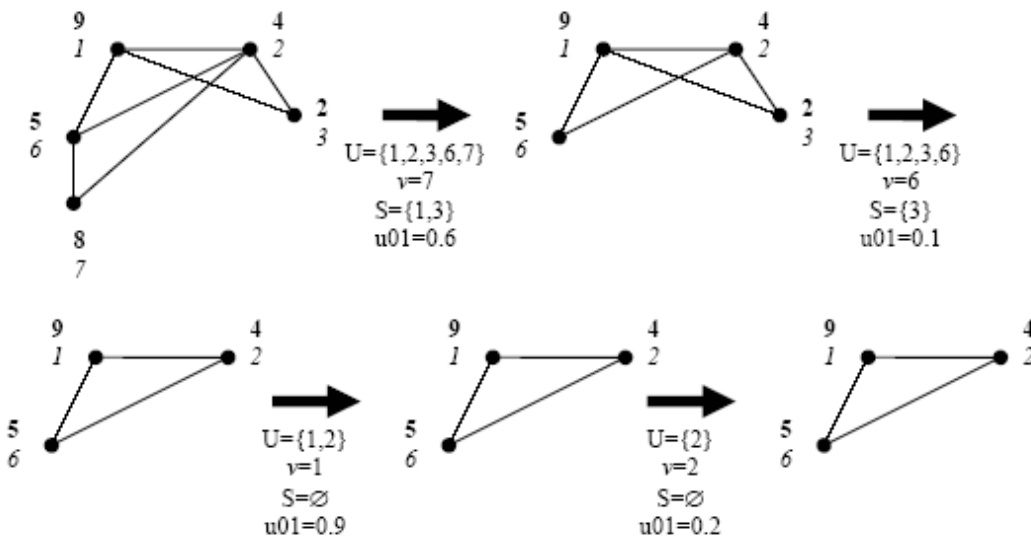


Figure 2. Illustrating the repair procedure by a simple example

0.8 so next vertex is selected randomly. Suppose vertex 4 is selected at random and included in the clique. After this we again compute  $C_{ad}$ . Therefore after the second iteration  $C_c = \{1,2,4,5\}$  and  $C_{ad} = \{3,6\}$ . In the third iteration suppose the value of  $u01$  comes out to be 0.1 so next vertex have to be selected by greedy rule. Now the product of weight and local degree of vertex 3 as well as vertex 6 is zero as these vertices are not connected to each other. Therefore tie is broken by selecting the vertex 6 at random. After including vertex 6 in  $C_c$ , the  $C_{ad}$  becomes empty and the heuristic stops.  $C_c = \{1,2,4,5,6\}$  is returned as the maximal weight clique. Its weight is 25. Here it is to be noted that maximal weight clique  $C_c$  returned by the heuristic depends on the values of  $u01$  obtained during iterations of the heuristic.

### III. The Genetic Algorithm

We have used a steady-state genetic algorithm [20]. The main features of our genetic algorithm are described below:

**Chromosome Representation:** We have used a bit vector of length  $n$  to represent the chromosome, where  $n$  is the number of vertices in the graph. A value of 1 at the  $i^{th}$  position indicates that vertex  $i$  is in the solution, whereas a 0 indicates that it is not. However, only those chromosomes that correspond to cliques are considered here.

**Fitness:** Fitness of a chromosome is equal to the sum of the weights of the vertices present in the clique it represents.

**Crossover:** Fitness based crossover as proposed by Beasley and Chu [21] is used in our genetic algorithm which produces a single child in the following way: If  $p_1$  and  $p_2$  are the two parents then the child receives bits from the parent

$p_1$  with probability  $f(p_1)/(f(p_1)+f(p_2))$  and from the parent  $p_2$  with probability  $f(p_2)/(f(p_1)+f(p_2))$  where  $f(p_1)$  and  $f(p_2)$  respectively are the fitness of  $p_1$  and  $p_2$ . Clearly in this scheme the child probabilistically receives more bits from best of the two parents.

Crossover is applied with probability  $p_c$  otherwise a random child is generated in which each bit is set to 1 with probability  $\min(1.0, 2 \times \text{maxsize}/n)$ , where  $\text{maxsize}$  is the size of the largest clique (in terms of number of vertices, not weight) so far generated by the algorithm and  $n$  is the number of vertices in the graph. This is the same probability as used in [19].

**Mutation:** A variation of simple bit flip mutation is used. In this mutation a bit which is 1 is set to 0 with probability  $p_m$  and a bit which is 0 is set to 1 with probability  $p_m$  only when the product of its weight and degree is greater than the average product of weight and degree of all the nodes of the graph.  $p_m$  is changed dynamically depending on the weight of the current best clique. It is an average of two terms one of which is constant and is set to 0.01 while the other terms is equal to  $\min(0.1, f(\text{current\_best})/(avwt \times n))$  where  $\text{current\_best}$  is the best clique found so far by the algorithm and  $avwt$  is the average weight of the vertices of the graph.

**Selection:** Binary tournament selection, where the candidate with better fitness is selected with probability  $p_{\text{betters}}$  is used to select the two parents.

**Repair:** As the child obtained after crossover and mutation may not be a clique, we have to use some sort of repair procedure to transform the child into a clique. The repair procedure randomly selects a vertex of the child and

Table 1. Performance of WT-HSS, QUALEX-MS and PBH on normal random weighted graphs

N	density	WT-HSS		QUALEX-MS		PBH	
		Avg. R.	St. Dev.	Avg. R.	St. Dev.	Avg. R.	St. Dev.
100	0.10	100.00%	0.00	100.00%	0.00	97.95%	0.15
100	0.20	100.00%	0.00	100.00%	0.00	97.73%	0.16
100	0.30	100.00%	0.00	99.87%	0.05	97.25%	0.17
100	0.40	100.00%	0.00	99.48%	0.18	95.04%	0.23
100	0.50	100.00%	0.00	99.45%	0.19	94.61%	0.24
100	0.60	100.00%	0.00	99.18%	0.21	94.71%	0.23
100	0.70	100.00%	0.00	98.02%	0.32	96.10%	0.20
100	0.80	100.00%	0.00	98.54%	0.29	93.13%	0.26
100	0.90	99.94%	0.00	98.43%	0.27	94.29%	0.24
100	0.95	99.98%	0.00	98.72%	0.20	96.49%	0.19
200	0.10	100.00%	0.00	100.00%	0.00		
200	0.20	100.00%	0.00	99.55%	0.19		
200	0.30	100.00%	0.00	99.33%	0.29		
200	0.40	100.00%	0.00	99.08%	0.45		
200	0.50	100.00%	0.00	98.34%	0.46		
200	0.60	100.00%	0.00	98.00%	0.35		
200	0.70	99.92%	0.00	96.99%	0.64		
200	0.80	99.91%	0.00	96.21%	0.55		
300	0.10	100.00%	0.00				
300	0.20	100.00%	0.00				
300	0.30	100.00%	0.00				
300	0.40	100.00%	0.00				
300	0.50	100.00%	0.00				
300	0.60	99.86%	0.00				
300	0.70	99.15%	0.01				
300	0.80	99.39%	0.01				

deletes either the vertex itself or all those vertices of the child which are not connected to the vertex selected. The whole procedure is repeated until the child becomes a clique. The repair procedure is described in detail by the pseudo-code given below where  $p_{delall}$  is the probability of deleting all those vertices of the child  $C$  that are not connected with the selected vertex,  $C_f$  determines the relative weight given to the average product of weight and degree of vertices not connected to the selected vertex in comparison to the product of weight and degree of the selected vertex:

```

U ← C
while (U ≠ ∅) {
    v ← random (U)
    S ← { u : u ∈ C ∧ (u, v) ∉ E }
    if (u0I ≤ pdelall) {
        C ← C - S
        U ← U - S
    }
else if ( deg(v) × w(v) ≤ Cf × ∑s ∈ S deg(s) × w(s) / |S| )
    C ← C - {v}
else {
    C ← C - S
}

```

```

U ← U - S
}
U ← U - {v}
}
return C

```

In all our computational experiments we have used  $p_{delall} = 0.5$  and  $C_f = 1.1$ . These values are chosen empirically. Figure 2 explains the repair procedure with the help of an example. This example uses the input graph given in Figure 1.

**Replacement Policy:** The child is first tested for uniqueness among the existing population members. If it is unique then it always replaces the worst member of the population irrespective of its own fitness otherwise it is discarded.

**Initial Population Generation:** To generate each member of the initial population we first generate a subgraph, where each vertex of the graph can be included in the subgraph with probability 0.2 ([19], [22]). Then the subgraph is transformed into a clique by the repair procedure and extended into maximal weight clique by the heuristic. The maximal weight clique is checked against already generated members for uniqueness and it is included in the initial population only if it is unique.

Table 2. Performance of WT-HSS, QUALEX-MS and PBH on irregular random weighted graphs

N	density	WT-HSS		QUALEX-MS		PBH	
		Avg. R.	St. Dev.	Avg. R.	St. Dev.	Avg. R.	St. Dev.
100	0.10	100.00%	0.00	100.00%	0.00	98.44%	0.13
100	0.20	100.00%	0.00	99.88%	0.05	98.64%	0.12
100	0.30	100.00%	0.00	99.89%	0.04	98.84%	0.11
100	0.40	100.00%	0.00	99.75%	0.05	98.53%	0.12
100	0.50	100.00%	0.00	99.81%	0.04	98.74%	0.12
100	0.60	100.00%	0.00	99.93%	0.02	99.64%	0.06
100	0.70	100.00%	0.00	99.84%	0.03	98.94%	0.11
100	0.80	100.00%	0.00	99.99%	0.00	98.56%	0.12
100	0.90	100.00%	0.00	99.99%	0.00	99.56%	0.07
100	0.95	100.00%	0.00	100.00%	0.00	99.75%	0.05
200	0.10	100.00%	0.00	99.97%	0.04		
200	0.20	100.00%	0.00	99.86%	0.04		
200	0.30	100.00%	0.00	99.45%	0.16		
200	0.40	100.00%	0.00	99.36%	0.35		
200	0.50	100.00%	0.00	99.32%	0.14		
200	0.60	100.00%	0.00	99.61%	0.10		
200	0.70	100.00%	0.00	99.54%	0.12		
200	0.80	100.00%	0.00	99.71%	0.10		
300	0.10	100.00%	0.00				
300	0.20	100.00%	0.00				
300	0.30	100.00%	0.00				
300	0.40	100.00%	0.00				
300	0.50	100.00%	0.00				
300	0.60	100.00%	0.00				
300	0.70	100.00%	0.00				
300	0.80	100.00%	0.00				

Table 3. Performance of WT-HSS on benchmark instances of Östergard.

Instance	n	Opt.	Clique Size			% Opt.	Time(sec.)	
			Best	Avg	St. Dev.		Avg.	St. Dev.
11-4-4	150	34	34	34.00	0.00	100%	0.03	0.02
12-4-6	230	110	110	109.30	3.05	95%	0.26	0.30
14-4-7	223	282	282	277.70	6.36	65%	0.41	0.28
14-6-6	807	42	42	42.00	0.00	100%	0.28	0.16
16-4-5	156	322	322	321.65	1.53	95%	0.19	0.19
16-8-8	2246	30	30	30.00	0.00	100%	0.06	0.04
17-4-4	132	156	156	156.00	0.00	100%	0.02	0.02
17-6-6	558	70	70	67.50	2.50	50%	0.71	1.19
19-4-6	263	1448	1448	1437.20	17.21	70%	0.67	0.54
19-8-8	2124	62	62	62.00	0.00	100%	2.13	1.69
20-6-5	1302	84	84	84.00	0.00	100%	1.70	1.70
20-6-6	1498	190	190	187.50	4.33	75%	11.60	7.46
20-8-10	2510	83	83	83.00	0.00	100%	0.18	0.15
21-10-9	5098	26	26	26.00	0.00	100%	6.58	4.65
22-10-10	8914	46	46	46.00	0.00	100%	6.92	6.86

## IV. Computational Results

The WT-HSS has been coded in C and executed on a Pentium IV 2.6 GHz Linux based system with 512 MB RAM. In all our experiments we have used a population of 50 individuals,  $p_{ad} = 0.6$ ,  $p_c = 0.8$ ,  $p_{better} = 0.8$ ,  $p_{delall} = 0.5$  and  $C_f = 1.1$ . All six parameters were set to their respective values after a large number of trials. These parameter values provided good results although they are in no way optimal for all problem instances. We ran WT-HSS until either the optimum solution (if known) is found or for a maximum of 20000 generations.

As there are no widely accepted benchmark graphs for the maximum weight clique problem we followed the approach of [14] to test our algorithm. We have tested WT-HSS against family of normal and irregular graphs of various node sizes and edge densities. To generate irregular graphs algorithm 4.1 of [12] was used. Vertex weights were uniformly distributed random integers in the range [1, 10]. A family of 20 random graphs was created, for a particular vertex size and edge density, for each of the two graph types. For each graph of the family the ratio of largest weight clique found by the algorithm to the actual maximum weight clique expressed in percentage was calculated. The performance of the algorithm was measured in terms of average ratio over the whole family. To find the actual maximum weight clique an exact algorithm as described in [11] was used. This C code for this algorithm is available at <http://www.tcs.hut.fi/~pat/wclique.html>. We are able to test WT-HSS not only on 100 and 200 vertex graphs as done in [15] for QUALEX-MS but also on 300 vertex graph with edge density up to 0.80. PBH [14] was tested only on graphs of size 100. However, due to the significant slowing down of exact solver on larger graphs, such type of testing was not possible.

Table 1 compares the performance of WT-HSS with QUALEX-MS [15] and PBH [14] on normal graphs while table 2 compares the performance of our algorithm on irregular graphs. Only a single run of WT-HSS was performed on a particular graph instance. As the obtained results depend on the seed used for random number generator, same seed value of 2 was used in all our experiments. Data for QUALEX-MS and PBH were taken respectively from [15] and [14]. Computational results clearly demonstrate the superiority of WT-HSS over QUALEX-MS and PBH. For irregular graphs WT-HSS was able to find the actual maximum weight clique in every member of every family. In case of normal graphs also the average ratio never falls below 99.15% that too in case of 300 node graph. Computational time of WT-HSS was also quite small. Average time of WT-HSS on any family of graphs was never more than 0.4 seconds.

Östergard [11] proposed 15 benchmark instances for the maximum weight clique problem based on real life coding theory problems. Number of nodes varied from 132 to 8914

in these instances. Table 3 shows the performance of our algorithm on these benchmark instances. Here WT-HSS was executed 20 times on each benchmark instance each time with a different random seed. Table 3 reports the best, average and standard deviation of maximal weight cliques found by WT-HSS on each instance. It also reports the average and standard deviation of time to find the best solution. % Opt is the percentage number of runs for which the optimal clique value was found by our algorithm in all runs of the algorithm. WT-HSS was able to find the optimal value for all instances. There are nine instances for which %Opt is 100% indicating that our algorithm was able to find the optimal value in all 20 runs. Computation times were also quite small.

However, the comparison of WT-HSS with genetic algorithm proposed in [9] is not possible. Balas and Niehaus [9] did not use the approach of [14], [15] for testing their algorithm. In [9] some random graph instances were used for testing, which are not available now. The source code of the genetic algorithm proposed in [9] is also not available and the algorithm itself is very difficult to program. Due to these reasons the two approaches could not be compared.

## V. Conclusions

We have developed a heuristic based steady-state genetic algorithm for the maximum weight clique problem which outperforms QUALEX-MS and PBH, the best heuristics known for the maximum weight clique problem. Our algorithm performs well, not only in terms of solution quality but also in terms of running time.

## Acknowledgement

We thank two anonymous referees for their helpful comments, and Anurag Singh Baghel for many fruitful discussions.

## References

- [1] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H Freeman, 1979.
- [2] S. Khot. "Improved Inapproximability Results for Max Clique, Chromatic Number and Approximate Graph Coloring". In *Proc. of the 42<sup>nd</sup> Annual IEEE Symposium on the Foundations of Computer Science*, pp. 600-609, 2001.
- [3] G. L. Nemhauser and L. E. Trotter. "Vertex Packings: Structural Properties and Algorithms", *Math. Programming*, VIII, pp. 232-248, 1975.
- [4] E. Loukakis and C. Tsouros. "An Algorithm for the Maximally Internally Stable Set in a Weighted Graph", *Int J. Computer Math.*, XIII, pp. 117-129, 1983.
- [5] R. Carraghan and P. M. Pardalos. *A Parallel Algorithm for the Maximum Weight Clique Problem*, Tech. Rep.

- CS-90-40, Department of Computer Science, Pennsylvania State University, 1990.
- [6] P. M. Pardalos and N. Desai. "An Algorithm for Finding a Maximum Weighted Independent Set in an Arbitrary Graph", *Int. J. Computer Math.*, XXXVIII, pp. 163-175, 1991.
- [7] E. Balas and J. Xue. "Minimum Weighted Coloring of Triangulated Graphs, with Application to Maximum Weight Vertex Packing and Clique Finding in Arbitrary Graphs", *SIAM Journal of Computing*, XX, pp. 209-221, 1991.
- [8] E. Balas and J. Xue. "Weighted and Unweighted Maximum Clique Algorithms with Upper Bounds from Fractional Coloring", *Algorithmica*, XV, pp. 397-412, 1996.
- [9] E. Balas and W. Niehaus. "Optimized Crossover-Based Genetic Algorithms for the Maximum Cardinality and Maximum Weight Clique Problems", *Journal of Heuristics*, IV, pp. 107-122, 1998.
- [10] L. Babel. "A Fast Algorithm for the Maximum Weight Clique Problem", *Computing*, LII, pp.31-38, 1994.
- [11] P. R. J. Östergård. "A New Algorithm for the Maximum Weight Clique Problem," *Nordic Journal of Computing*, VIII, pp. 424-436, 2001.
- [12] I. M. Bomze, M. Pelillo and V. Stix. "Approximating the Maximum Weight Clique using the Replicator Dynamics", *IEEE Transactions on Neural Networks*, XI, pp. 1228-1241, 2000.
- [13] T. S. Motzkin and E. G. Straus. "Maxima for Graphs and a New Proof of Theorem of Turan", *Canadian Journal of Mathematics*, XVII, pp. 533-540, 1965.
- [14] A. Massaro, M. Pelillo and I. M. Bomze. "A Complementary Pivoting Approach to the Maximum Weight Clique Problem", *SIAM Journal of Optimization*, XII, pp. 928-948, 2002.
- [15] S. Busygin. "A New Trust Region Technique for the Maximum Weight Clique Problem", *Discrete and Applied Mathematics (Special issue on combinatorial optimization)*, Submitted for publication.
- [16] R. Horaud and T. Skordas. "Stereo Correspondence through Feature Grouping and Maximal Cliques", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, XI, pp. 1168-1180, 1989.
- [17] M. Pelillo, K. Siddiqi K. and S. W. Zucker. "Matching Hierarchical Structures using Associated Graphs", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, XXI, pp. 1105-1120, 1999.
- [18] I. M. Bomze, M. Budinich, P. M. Pardalos and M. Pelillo. "The Maximum Clique Problem", in *Handbook of Combinatorial Optimization*, IV, Boston, MA: Kluwer Academic Publishers, 1999.
- [19] A. Singh and A. K. Gupta. "A Hybrid Heuristic for the Maximum Clique Problem", *Journal of Heuristics*, XII, pp. 5-22, 2006.
- [20] L. Davis. *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.
- [21] J. E. Beasley and P. C. Chu. "A Genetic Algorithm for the Set Covering Problem", *European Journal of Operation Research*, XCIV, pp. 394-404, 1996.
- [22] E. Marchiori. "Genetic, Iterated and Multistart Local Search for the Maximum Clique Problem". In *Proceedings of Applications of Evolutionary Computing in Combinatorial Optimization (EvoCOP 2002)*, pp. 112-121, LNCS 2279, Springer, 2002.

### Author Biographies

**Alok Singh** received the Bachelors and Masters degrees in Computer Science from Banaras Hindu University, Varanasi, India in 1996 and 1998 respectively. He received the Ph.D. degree in Computer Science from University of Allahabad, Allahabad, India in 2006.

He is currently a Lecturer of Computer Science at the J. K. Institute of Applied Physics and Technology, University of Allahabad, Allahabad, India. His primary research interests lie in the area of combinatorial optimization with problem-specific heuristics and meta-heuristics. He has authored or coauthored 10 publications in referred proceedings and journals.

**Ashok Kumar Gupta** received the B.Sc. and M.Sc.(Physics) degrees from University of Allahabad, Allahabad, India in 1960 and 1962 respectively.

He is currently a Professor of Computer Science at the J. K. Institute of Applied Physics and Technology, University of Allahabad, Allahabad, India. He is also the Director of the Institute of Interdisciplinary Studies at the University of Allahabad. He had been the member of University Grants Commission (2002-2005), the apex body of higher education in India. His current research interests are evolutionary computation and bio-informatics. He has authored or coauthored more than 50 publications in referred proceedings and journals.