Genetic, Iterated and Multistart Local Search for the Maximum Clique Problem

Elena Marchiori

Free University Amsterdam, Department of Computer Science
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
elena@cs.vu.nl

Abstract. This paper compares experimentally three heuristic algorithms for the maximum clique problem obtained as instances of an evolutionary algorithm scheme. The algorithms use three popular heuristic methods for combinatorial optimization problems, known as genetic, iterated and multistart local search, respectively.

1 Introduction

A clique of an undirected graph is a subgraph in which all pairs of distinct nodes are connected by an edge. A maximum clique of a graph is a clique with maximum number of nodes. Computing the maximum clique of a graph is a paradigmatic combinatorial optimization problem which is encountered in many different real life applications, such as cluster analysis, information retrieval, mobile networks, and computer vision (see, e.g., the survey in [2]). MC is highly intractable: it is one of the first problems which has been proven to be NP-hard [13]. Moreover, even its approximations within a constant factor are NP-hard [6, 9]. Due to these strong negative results on the computational complexity of MC, many researchers have concentrated their effort on designing efficient heuristics yielding sub-optimal solutions of satisfactory quality (e.g., [1, 2, 11]).

Genetic algorithms have been applied with success to various hard combinatorial optimization problems. On the MC problem pure genetic algorithms have poor performance when compared with other local search techniques [4, 18]. It is not yet clear which graph properties can be used to measure the hardness of this problem for GAs. Experimental investigations [10, 20] indicate that neither graph density, size, relative maximum clique size, relative number of cliques, nor epistasis variance [5] are appropriate hardness measures.

The situation improves when problem knowledge is incorporated in GAs in the form of ad-hoc genetic operators and/or local optimization techniques [3, 7, 8, 17, 19]. In particular, in [14] we have introduced a hybrid GA for the maximum clique yielding results competitive with those obtained by state-of-the-art heuristic algorithms. The hybrid GA combines a simple GA with a local search procedure which generates maximal cliques.

In this paper we compare experimentally (a slightly different version of) this hybrid GA with an iterated local search algorithm and a multistart local

search algorithm, which are obtained by choosing ad-hoc parameter settings of the hybrid GA (population size, termination condition, mutation and crossover rate). In this way we obtain three heuristic algorithms based on genetic local search, iterated local search, and multistart local search, respectively, which use an equal local search procedure as core of the optimization process, and explore a similar number of search points. These similarities allows one to compare fairly the three approaches the algorithms are based on.

The genetic local search algorithm (called `GENE`) consists of the application of genetic operators to a population of local optima produced by a local search procedure. The process is iterated until either a solution is found or a maximal number of generations is reached. The final output is the best solution found in all iterations. The iterated local search algorithm (`ITER`) acts repeatedly on just one candidate solution and outputs the best solution found in all iterations. The multistart local search algorithm (`MULT`) applies the local search procedure to each element of a large set of candidate solutions and outputs the best solution contained in the set. This has not to be confused with the restart method used in local search algorithms where execution can be periodically restarted in case past events indicate the search could be stuck in an attraction basin of some local optima.

The effectiveness of these three algorithms is tested on standard benchmark instances for MC collected at the DIMACS Center. The genetic and iterated local search algorithms exhibit similar performance, with results comparable to those of the best heuristic algorithms tested at the DIMACS Implementation Challenge for Maximum Clique, Graph Coloring, and Satisfiability [11]. Instead, the results of the multistart algorithm are of inferior quality.

The following notation and terminology is used throughout the paper. A graph is denoted by G, its nodes by $m, n, \ldots$. Nodes of a graph of size N are supposed to be indexed with integers from 1 to N. A subgraph C_G of G is a *clique* if every two distinct nodes n, n' in C_G are connected with an edge. A *maximal clique* C_G of G is a clique which is not properly contained in any other clique of G. A *maximum clique* C_G of G is a clique of maximum size (i.e./ number of maximum number of nodes). Note that a maximum clique is maximal but not vice versa.

2 The Algorithms

We use an approach called genetic local search (see e.g., [15]), which amounts to the repeated application of genetic operators to selected individuals of a population of local optima. The scheme algorithm we use is called GLS and is summarized in pseudo code below, where $P(t)$ denotes the population P at iteration t, $|P(t)|$ its size, and `LMC` is a local search procedure for finding maximal cliques that will be described later on. At each iteration, a new population is generated from the actual one as follows. Two fit individuals called parents are selected, crossover and mutation are applied to produce two offsprings which are then optimized by applying `LMC` to each of them. The best two individuals

amongst parents and offsprings are selected and added to the new population (*keep-two-best* replacement mechanism, see [22]). This process is repeated until the new population reaches the size of the actual one.

In our implementation a chromosome represents a subgraph S_G by means of a bit string x of length N, where $x_i = 1$ (resp. $x_i = 0$) means that node i is (resp. is not) in S_G. Set-theoretic operators on graphs are translated to operators on binary strings in the expected way.

We employ a generational model with fitness proportional selection rule (roulette-wheel, cf. [16]) and elitism [12], where the two best individuals of a population are copied to the population of the next generation.

The fitness function $f : Chrm \rightarrow [0, N]$ is defined by $f(x) = |Nodes(x)|$ (the number of nodes of x) if x is a clique; $f(x) = 0$ otherwise.

Finally, we use classical blind genetic operators: uniform crossover [21], and swap mutation which swaps the values of two randomly selected genes. Note these operators describe meaningful operations on graphs, where crossover corresponds to a merge of graphs and mutation corresponds to exchange of nodes between graphs.

```
PROCEDURE GLS
   BEGIN
    t := 0;
    initialize P(t);
   apply LMC to each element of P(t);
    evaluate P(t);
    WHILE (NOT termination-condition) DO
     BEGIN
      t := t+1;
      WHILE (|P(t)| < |P(t-1)|) DO
       BEGIN
        select parents from P(t-1);
        recombine parents;
        mutate children;
        apply LMC to each of the children;
        evaluate children;
        insert in P(t) best two of parents and children;
       END
     END
   END
```

The local search algorithm `LMC` used in GLS is described below.

```
PROCEDURE LMC
BEGIN
  S_G := Perturb(S_G);
  S_G := Repair(S_G);
  S_G := Extend(S_G);
END
```

This procedure transforms a subgraph into a maximal clique: first, the subgraph is perturbed by deleting and adding some nodes randomly selected (Perturb); next, it is reduced to a clique (Repair); finally it is extended to a maximal clique (Extend) using a sequential greedy heuristic. The three steps are described in detail below.

`Perturb(`S_G`)`:

1. For every node n, $1 \leq n \leq N/2$, if n is in S_G then with small probability (typical value 0.1) remove n from S_G. (In this step nodes are supposed to be sorted in increasing order with respect to their degree. The upper bound $N/2$ has been chosen empirically).
2. Add the sequence $s, s+1, \ldots, s+e$ of nodes to S_G, with s and e randomly chosen, where $3 \leq e \leq BK/2$ and s in $[1, N-e]$ (BK is the size of the largest known clique G. The upper bound $BK/2$ has been chosen empirically).

`Repair(`S_G`)`: Let V be the set of nodes of S_G. Repeat the following steps until V becomes empty.

1. Choose randomly a node n in V.
2. With low probability (typical value 0.01) remove n from S_G; otherwise delete from S_G and from V each node of S_G that is not connected with n, except n itself.
3. Remove n from V.

`Extend(`S_G`)`: Let V be the set of nodes of $G \setminus S_G$. Repeat the following steps until V becomes empty.

1. Choose a random node n in V:
2. if $\{n\} \cup S_G$ is a clique then add n to S_G.
3. remove n from V.

We consider the following three instances of GLS, obtained by setting the parameters crossover rate, mutation rate, population size, and termination criterion to specific values. The parameter settings are such that all algorithms process approximately the same number (equal to 20,000) of individuals.

- Genetic algorithm (`GENE`). It is obtained from GLS by setting population size to 10, mutation rate to 0.1, crossover rate to 0.9 and termination-condition to $2,000$ generations. This algorithm performs genetic local search.
- Iterated local search (`ITER`). It is obtained from GLS by setting population size to 1, mutation rate to 0 (hence no genetic operators are used), and termination-condition to $20,000$ generations. This algorithm iterates local search (using `LMC`) starting from one random point in the search space.
- Multistart local search (`MULT`). This is obtained from GLS by setting population size to $20,000$ and termination-condition to 0 generations (hence no genetic operators are used). This algorithm performs a local search (using `LMC`) from several points randomly distributed over the entire search space.

3 Experimental Comparison

In order to test and compare the three algorithms we consider the benchmark instances employed in the International Implementation Challenge on Maximum

Clique, Graph Coloring, and Satisfiability organized by the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) [11]. The algorithms described in [11] are based on various approaches, like tabu search, simulated annealing, and neural networks. These instances are available from the DIMACS archive using ftp to `dimacs.rutgers.edu` directory `pub/challenge` or at URL `http://dimacs.rutgers.edu/ Challenge/`. The 37 instances here considered include random graphs (`Cx.y` and `DSJCx.y` of size x and density 0.y), Steiner Triple Graphs (`MANNx` with up to 3321 nodes and 5506380 edges), Brockington Graphs (`brockx_2` and `brockx_4` of size x), Sanchis graphs (`genx_p0.9_z` of size x), Hamming graphs (`hamming8-4` and `hamming10-4`), Keller graphs (`keller4, keller5, keller6` with up to 3361 nodes and 4619898 edges), and P-hat graphs (`p_hatx-z` of size x).

GLS has been implemented in C++ and run on a Sun Ultra 250, UltraSPARC-II 400MHz. The initial population is generated randomly, where each gene of a chromosome is set to 1 with probability 0.2, otherwise it is set to 0. These and the other GA parameters have been set to values experimentally determined after a small number of trials. `MULT`, `ITER` and `GENE` are run 10 times on each graph instance using different random seeds. The results of the experiments are summarized in Table 1 which reports the best, average and standard deviation of clique size, and the best result obtained by the fifteen heuristic algorithms for MC presented at the DIMACS Challenge (column labeled DIMACS best). Table 2 which contains average and standard deviation of time to find the best solution (in seconds).

The multistart variant `MULT` has worst performance: `MULT` outperforms `ITER` and `GENE` only on one instance (`brock200_4`), while on all other instances it yields in general results of poor quality. `ITER` and `GENE` find maximal cliques of comparable quality: `ITER` outperforms `GENE` on five instances while `GENE` outperforms `ITER` on four instances (entries in bold style). The variance of the results obtained by `ITER` and `GENE` is also comparable. On 23 of the 37 instances `ITER` is able to find the best value found by the DIMACS algorithms.

The experiments on the DIMACS snapshot indicate that `ITER` and `GENE` are the most effective of the three variants. It seems that most of the work in guiding the search towards promising regions is performed by the local search procedure `LMC`, while the genetic operators are useful for escaping from local optima, due to their disruptive effect (e.g. on random graphs `C500.9`, `C1000.9` and `C2000.9`).

The best heuristic algorithm for the MC problem we are aware of is based on reactive local search (RLS) [1]. This algorithm is roughly ten times faster than `ITER`, and yields the best known results in almost all the DIMACS benchmark instances, with very small standard deviations. RLS employs a sophisticated search strategy that exploits information about past events in two ways: memorization of past events as done in tabu search, and a reactive strategy which regulates the search diversification. Moreover an explicit memory-influenced restart is activated periodically. In contrast, GLS is a Markov (i.e., memory-less) process, where the next generation depends only on the current one.

3.1 Comparison with Hybrid Genetic Algorithms

We are aware of two hybrid genetic algorithms for MC which have being tested on some benchmarks from the DIMACS repository.

The algorithm by Bui and Eppley [3], called GMCA, is a hybrid GA with local optimization for improving the chromosomes. Moreover pre-processing is used for reordering the nodes of the graph in such a way that nodes which are likely to belong to a clique (e.g. with high degree) occur near each other. The fitness function is the weighted sum of density and size of the subgraph represented by the chromosome.

In [19] Sakamoto et al. introduce a GA for the maximum independent set problem, here called SLS. An independent set of a graph G is a clique of the complement of G (it has the nodes of G but only those edges which are not in G). The maximum clique and the maximum independent problem are equivalent since a maximum clique of G is a maximum independent set of the complement of G. In SLS a chromosome is a permutation of the nodes of the graph. A greedy decoding method is used which constructs a independent set using the nodes in the order in which they appear in the chromosome. The chromosome is then replaced with the sequence of nodes of the independent set followed by the remaining nodes of the graph. The fitness function is the size of the independent set minus the minimum of the sizes of the independent sets in the actual population.

In Table 3 GMCA, SLS, and `GENE` are compared by reporting the results contained in [3] and [19] concerning experiments on a set of DIMACS instances. Both GMCA and SLS use a population of size 50 and are run for 50 generations. In order to compare running times - GMCA and SLS are run on a Sun SPARC LX - we converted the running time of `GENE` to Sun SPARC LX time using the Dhrystone score. It turns out that our computer is about 17 times faster than a Sun SPARC LX.

On 12 instances (in bold style) `GENE` outperforms GMCA and SLS; on the other instances SLS and `GENE` yield equal best results, and they outperform GMCA on 10 instances. Concerning the running time, `GENE` is faster than GMCA and SLS, except on the `MANN*` instances: in particular on `MANN_a45` and `MANN_a81` `GENE` is about ten times slower than SLS, but is able to find the best known result.

The results also indicate that instances from the `Cfat`, `Johnson` and `Hamming` classes can be regarded as GA easy.

References

1. R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. *Algorithmica*, 29(4):610–637, 2001.
2. I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo. The maximum clique problem. *Handbook of Combinatorial Optimization*, 4, 1999.
3. T.N. Bui and P.H. Eppley. A hybrid genetic algorithm for the maximum clique problem. In L.J. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA)*, pages 478–484. Morgan Kaufmann, 1995.

4. B. Carter and K. Park. How good are genetic algorithms at finding large cliques: an experimental study. Technical report, Boston University, Computer Science Department, MA, October 1993.
5. Y. Davidor. Epistasis variance: a viewpoint on GA-hardness. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 23–35. Morgan Kaufmann, 1991.
6. U. Feige, S. Goldwasser, S. Safra, L. Lovász, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 2–12, 1991.
7. C. Fleurent and J.A. Ferland. Object-Oriented imlementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In D. Johnson and M. Trick, editors, *Cliques, Coloring and Satisfiability*. AMS, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 26, 1996.
8. J.A. Foster and T. Soule. Using genetic algorithms to find maximum cliques. Technical report, Dept. of Computer Science, Univ. Idaho, 12 1995.
9. J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proc. 37th Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 627–636, 1996.
10. T. Haynes. Clique detection as a royal road function. In *Genetic Programming*, 1998.
11. D. Johnson and M. Trick (Eds.). *Cliques, Coloring, and Satisfiability*. AMS, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 26, 1996.
12. K.A. De Jong. An analysis of the behaviour of a class of genetic adaptive systems. Doctoral Dissertation, University of Michigan, Dissertation Abstract International 36(10), 5140B, 1975.
13. R.M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, NY, 1972.
14. E. Marchiori. A simple heuristic based genetic algorithm for the maximum clique problem. In J. Carroll et al., editor, *ACM Symposium on Applied Computing*, pages 366–373. ACM Press, 1998.
15. P. Merz and B. Freisleben. Genetic local search for the TSP: New results. In *IEEE International Conference on Evolutionary Computation*, pages 159–164. IEEE Press, 1997.
16. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 1994.
17. A.S. Murthy, G. Parthasarathy, and V.U.K. Sastry. Clique finding - a genetic approach. In *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pages 18–21. IEEE Press, 1994.
18. K. Park and B. Carter. On the effectiveness of genetic search in combinatorial optimization. In *Proceedings of the 10th ACM Symposium on Applied Computing*. ACM Press, 1995.
19. A. Sakamoto, X. Liu, and T. Shimamoto. A genetic approach for maximum independent set problems. *IEICE Trans. Fundamentals*, E80-A(3):551–556, 1997.
20. T. Soule and J.A. Foster. Genetic algorithm hardness measures applied to the maximum clique problem. In T. Bäck, editor, *Seventh International Conference on Genetic Algorithms*, pages 81–88. Morgan Kaufmann, 1997.
21. G. Syswerda. Uniform crossover in genetic algorithms. In J. Schaffer, editor, *Third International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1989.
22. D. Thierens and D. Goldberg. Elitist recombination: an integrated selection recombination GA. In *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pages 508–518. IEEE Press, 1994.

Graph	MULT		GENE		ITER		
	Avg(Stdv)	Best	Avg(Stdv)	Best	Avg(Stdv)	Best	DIMACS best
C125.9	32.6(0.5)	33	33.8(0.4)	34	34(0.0)	34	34
C250.9	39.1(0.7)	40	42.8(0.7)	44	43.0(0.6)	44	44
C500.9	46.7(0.6)	48	52.2(1.6)	**56**	52.7(1.4)	55	57
C1000.9	53.5(1.2)	56	61.6(2.1)	**66**	61.6(1.6)	64	67
C2000.9	59.7(0.9)	62	68.2(2.4)	**72**	68.7(1.2)	70	75
DSJC500.5	12.0(0.0)	12	12.2(0.4)	13	12.1(0.3)	13	15
DSJC1000.5	13.1(0.3)	14	13.3(0.5)	14	13.5(0.5)	14	15
C2000.5	14.1(0.3)	15	14.2(0.4)	15	14.2(0.4)	15	16
C4000.5	15.2(0.4)	16	15.4(0.5)	16	15.6(0.5)	16	18
MANN_a27	124.8(0.4)	125	125.6(0.5)	126	126.0(0.0)	126	126
MANN_a45	339.7(0.5)	340	342.4(0.5)	343	343.1(0.8)	**345**	345
MANN_a81	1091.2(0.7)	1092	1096.3(0.6)	1097	1097.0(0.4)	**1098**	1098
brock200_2	12(0.0)	12	10.5(0.7)	12	10.5(0.8)	12	12
brock200_4	15.7(0.9)	**17**	15.4(0.5)	16	15.5(0.5)	16	17
brock400_2	21.7(0.6)	23	22.5(0.7)	24	23.2(0.7)	**25**	25
brock400_4	21.8(0.4)	22	23.6(0.8)	**25**	23.1(0.5)	24	24
brock800_2	18.0(0.6)	19	19.3(0.6)	20	19.1(0.8)	**21**	21
brock800_4	18.0(0.0)	18	18.9(0.5)	20	19.0(0.4)	20	21
gen200_P0.9_44	36.3(0.5)	37	39.7(1.6)	44	39.5(1.6)	44	44
gen200_P0.9_55	43.4(2.0)	46	50.8(6.4)	55	48.8(7.6)	55	55
gen400_P0.9_55	43.7(0.6)	45	49.7(1.2)	55	49.1(1.0)	51	55
gen400_P0.9_65	44.6(0.9)	47	53.7(7.4)	65	51.2(4.7)	65	65
gen400_P0.9_75	47.7(1.7)	52	60.2(12.1)	75	62.7(12.3)	75	75
hamming8-4	15.7(0.9)	16	16.0(0.0)	16	16.0(0.0)	16	16
hamming10-4	32.0(0.4)	33	37.7(1.9)	40	38.8(1.2)	40	40
keller4	11.0(0.0)	11	11.0(0.0)	11	11.0(0.0)	11	11
keller5	23.9(0.9)	25	26.0(0.8)	27	26.3(0.6)	27	27
keller6	45.3(1.1)	48	51.8(1.5)	55	52.7(1.8)	**56**	59
p_hat300-1	8.0(0.0)	8	8.0 (0.0)	8.0	8.0 (0.0)	8.0	8
p_hat300-2	22.9(0.9)	25	25(0.0)	25	25(0.0)	25	25
p_hat300-3	31.0(0.4)	32	34.6(0.9)	36	35.1(0.8)	36	36
p_hat700-1	9.1(0.3)	10	9.8(0.9)	11	9.9(0.7)	11	11
p_hat700-2	35.5(0.9)	37	43.5(0.8)	44	43.6(0.7)	44	44
p_hat700-3	49.5(1.4)	52	60.4(1.0)	62	61.8(0.6)	62	62
p_hat1500-1	10.2(0.4)	11	10.8(0.4)	11	10.4(0.5)	11	12
p_hat1500-2	46.9(1.0)	48	63.8(1.0)	65	63.9(2.0)	65	65
p_hat1500-3	64.3(1.3)	67	92.4(1.3)	94	93.0(0.8)	94	94

Table 1. Results for DIMACS 'snapshot': clique size

Graph	MULT	ITER	GENE
	Avg(Stdv)	Avg(Stdv)	Avg(Stdv)
C125.9	2.4(0.1)	0.5(0.6)	0.1(0.1)
C250.9	4.4(0.1)	2.4(2.0)	3.7(3.5)
C500.9	8.8(0.1)	2.7(2.6)	5.7(5.5)
C1000.9	18.2(0.1)	8.6(6.5)	12.4(13.4)
C2000.9	46.3(0.3)	24.8(23.8)	33.8(34.2)
DSJC500.5	6.6(0.1)	0.4(0.5)	2.1(4.9)
DSJC1000.5	13.4(0.1)	2.3(3.8)	2.2(1.9)
C2000.5	29.3(0.1)	2.3(2.0)	7.2(12)
C4000.5	63.2(0.3)	15.7(8.0)	13.5(14.0)
MANN_a27	14.4(0.1)	15.6(10.1)	3.7(4.0)
MANN_a45	92.7(0.4)	54.4(44.3)	135.2(106.4)
MANN_a81	1203.3(39.5)	693.9(922.5)	2773.8(1158.3)
brock200_2	2.7(0.0)	0.2(0.2)	1.3(1.8)
brock200_4	2.8(0.1)	0.5(0.6)	0.9(1.6)
brock400_2	5.7(0.1)	2.0(2.3)	1.9(3.3)
brock400_4	5.7(0.1)	1.3(1.4)	1.3(1.4)
brock800_2	11.0(0.1)	3.9(4.9)	5.2(7.4)
brock800_4	11.0(0.1)	4.1(3.4)	7.8(7.7)
gen200_P0.9_44	3.5(0.0)	1.7(1.5)	1.3(1.3)
gen200_P0.9_55	3.5(0.0)	1.3(1.5)	1.4(2.9)
gen400_P0.9_55	6.8(0.0)	2.8(3.0)	3.8(6.5)
gen400_P0.9_65	6.8(0.0)	2.7(3.3)	4.3(4.3)
gen400_P0.9_75	6.8(0.1)	4.6(3.2)	3.7(6.3)
hamming8-4	3.5(0.0)	0.0(0.0)	0.0(0.0)
hamming10-4	15.5(0.1)	5.3(4.4)	5.8(5.6)
keller4	2.4(0.0)	0.0(0.0)	0.0(0.0)
keller5	10.9(0.1)	4.0(3.9)	9.1(10.2)
keller6	69.2(0.1)	36.2(23.8)	60.2(55.0)
p_hat300-1	3.7(0.0)	0.9(0.8)	0.4(0.5)
p_hat300-2	4.1(0.0)	0.5(0.5)	0.5(0.6)
p_hat300-3	4.4(0.0)	1.5(1.7)	3.6(4.5)
p_hat700-1	8.7(0.1)	2.6(3.8)	5.8(6.8)
p_hat700-2	9.5(0.0)	1.2(1.2)	1.6(1.9)
p_hat700-3	10.5(0.0)	4.5(4.4)	5.6(6.3)
p_hat1500-1	19.5(0.1)	2.1(3.1)	14.2(14.7)
p_hat1500-2	21.6(0.1)	12.2(9.2)	9.1(7.8)
p_hat1500-3	24.5(0.0)	7.1(11.4)	14.6(18.2)

Table 2. Results for DIMACS 'snapshot': time till best (in seconds)

Graph	GMCA		SLS		GENE	
	Avg Time	Best	Avg Time	Best	Avg Time	Best
c-fat200-1	8.2	12	12.3	12	0.0	12
c-fat500-1	33.2	14	60.7	14	0.0	14
johnson16-2-4	6.0	8	4.5	8	0.0	8
johnson32-2-4	187.4	16	63.2	16	1.7	16
keller4	13.3	11	9.1	11	0.0	11
keller5	438.1	18	256.7	27	63.7	27
keller6	-	-	4798.6	50	1023.4	**55**
hamming10-2	886.6	512	351.3	512	37.2	512
hamming8-2	53.0	128	21.2	128	1.7	128
san200_0.7_1	51.7	30	16.6	30	10.2	30
san400_0.5_1	411.2	7	43.1	13	42.5	13
san400_0.9_1	128.6	50	70.5	100	47.6	100
sanr200_0.7	21.5	17	14.6	18	10.5	18
sanr400_0.5	69.6	12	45.8	13	42.5	13
san1000	704.3	8	242.8	10	15.3	10
brock200_1	27.9	20	14.9	20	15.3	20
brock200_2	-	-	12.1	10	22.1	**12**
brock200_4	-	-	14.0	16	15.3	16
brock400_1	118.8	20	50.9	23	44.2	**24**
brock400_2	-	-	52.5	24	32.3	24
brock400_4	-	-	56.8	23	22.1	**25**
brock800_1	460.8	18	172.1	18	27.2	**19**
brock800_2	-	-	188.8	19	66.3	**20**
brock800_4	-	-	171.8	20	69.7	20
p_hat300_1	20.0	8	24.1	8	6.8	8
p_hat300_2	-	-	30.4	25	8.5	25
p_hat300_3	-	-	36.1	35	61.2	**36**
p_hat500_1	49.1	9	62.0	9	3.5	9
p_hat700_1	310.1	8	117.7	11	105.4	11
p_hat700_2	-	-	165.0	44	27.2	44
p_hat700_3	-	-	192.7	61	95.2	**62**
p_hat1000_1	671.0	8	247.6	10	54.4	10
p_hat1500_1	1580.3	10	573.2	11	249.9	11
p_hat1500_2	-	-	952	64	154.7	**65**
p_hat1500_3	-	-	1079.5	92	248.2	**94**
MANN_a27	121.8	125	52.4	126	62.9	126
MANN_a45	916.9	337	339.5	341	2298.4	**345**
MANN_a81	-	-	6249.0	1094	47154.6	**1098**

Table 3. Results for DIMACS benchmark graphs: GMCA, SLS, GENE