Solving the Maximum Clique Problem by k-opt Local Search

Kengo Katayama
Department of Information and Computer Engineering,
Okayama University of Science.
1 - 1 Ridai-cho, Okayama,
700-0005 Japan
katayama@ice.ous.ac.jp

Akihiro Hamamoto
Department of Information and Computer Engineering,
Okayama University of Science.
1 - 1 Ridai-cho, Okayama,
700-0005 Japan
a-hamamoto@ice.ous.ac.jp

Hiroyuki Narihisa
Department of Information and Computer Engineering,
Okayama University of Science.
1 - 1 Ridai-cho, Okayama,
700-0005 Japan
narihisa@ice.ous.ac.jp

ABSTRACT

This paper presents a local search algorithm based on variable depth search, called the *k-opt local search*, for the maximum clique problem. The k-opt local search performs add and drop moves, each of which can be interpreted as 1-opt move, to search a k-opt neighborhood solution at each iteration until no better k-opt neighborhood solution can be found. To evaluate our k-opt local search algorithm, we repeatedly apply the local search for each of DIMACS benchmark graphs and compare with the state-of-the-art metaheuristics such as the genetic local search and the iterated local search reported previously. The computational results show that in spite of the absence of major metaheuristic components, the k-opt local search is capable of finding better (at least the same) solutions on average than those obtained by these metaheuristics for all the graphs.

Keywords

Local Search, Variable Depth Search, Neighborhood, Maximum Clique Problem, Combinatorial Optimization

1. INTRODUCTION

It is considered that a design of Local Search is quite important in that many metaheuristic algorithms such as evolutionary algorithms are based on it in order to obtain high quality solutions for combinatorial optimization problems. In fact, if a high performance local search is incorporated into a metaheuristic algorithm, the metaheuristic based on the high performance local search often obtains better solutions than those obtained by a metaheuristic based on a poor performance local search. For example, in the traveling salesman problem [6] that is one of the representative combinatorial optimization problems, it is known that the performance of an iterated local search metaheuristic based on Lin-Kernighan local search heuristic [11] generally outperforms one based on 2-opt or 3-opt local search.

The *variable depth search* (VDS) is known to be a generalization of the local search methods. The VDS was applied by Lin and Kernighan to the traveling salesman problem (TSP) [11] and graph partitioning problem (GPP) [10]. Their algorithms are often called *k-opt local search*. The basic concept is to search a portion of large (k-opt) neighborhood while keeping a reasonable amount of computation time. Recently, for the both problems TSP and GPP the VDS based heuristics have been incorporated into several metaheuristics such as evolutionary algorithm [14] and iterated local search algorithm [1, 7, 9]. Generally, the performance of the metaheuristics embedded with the VDS is remarkably effective for the problems. From this point of view, the metaheuristics with the high performance local search based on VDS are expected to be fairly promising even for other combinatorial optimization problems.

The maximum clique problem (MCP) is also one of the representative combinatorial optimization problems. Solving the MCP is encountered in the various real applications, such as mobile networks, computer vision, cluster analysis, etc. However, the problem is known to be NP-hard [4], and strong negative results such as [5] have been shown. Thus, many researchers have focused on developing efficient metaheuristic algorithms based on local search to find near optimal solutions to the MCP.

Recently, Marchiori [12, 13] showed effective metaheuristic algorithms based on the genetic local search and the iterated local search for the MCP. She demonstrated that her metaheuristics outperformed the previously reported heuristics based on evolutionary algorithms for several DIMACS benchmark graphs [3]. Currently, her metaheuristic algorithms are known to be one of the most effective algorithms based on evolutionary algorithms for the MCP.

In this paper, we present a local search algorithm inspired by VDS, called k-opt local search, for solving the MCP. The k-opt local search performs simple *add* and *drop* moves which can be interpreted as *1-opt* move to search a k-opt neighborhood solution at each iteration. The k-opt neighborhood is defined by the set of solutions obtained by a sequence of several 1-opt moves that add and drop moves are adaptively changed in the feasible search space.

To show the effectiveness of the k-opt local search for the MCP, we repeatedly apply the local search for each of well-

SAC '04, March 14–17, 2004, Nicosia, Cyprus.

```
    procedure 1-opt-Local-Search-Add-Move(CC, PA, deg_{G(PA)})
  begin
1     repeat
2         find a vertex v with max_{v∈PA} { deg_{G(PA)}(v) };
3         if multiple vertices with the same maximum degree are found then select one vertex v among them randomly;
4         CC := CC ∪ {v};
5         update PA and deg_{G(PA)}(i), ∀i ∈ PA;
6     until PA = ∅;
7     return CC;
  end;
```

Figure 1: The pseudo code of the 1-opt local search with add move for the MCP

known DIMACS benchmark graphs. Computational results show that although the k-opt local search itself doesn't have major metaheuristic components, the k-opt local search is capable of finding better (at least the same) solutions on average than the metaheuristics based on evolutionary algorithm for all the graphs.

2. MAXIMUM CLIQUE PROBLEM

Let $G = (V, E)$ be an arbitrary undirected graph where $V = \{1, 2, \ldots, n\}$ is the set of vertices in G and $E \subseteq V \times V$ is the set of edges in G. For a subset $S \subseteq V$, let $G(S) = \{S, E \cap S \times S\}$ be *the subgraph* induced by S.

A graph $G = (V, E)$ is *complete* if all its vertices are pairwise adjacent, i.e., $\forall\ i,\ j \in V$ with $i \neq j$, $(i, j) \in E$. A *clique* C is a subset of V such that the induced graph $G(C)$ is complete. The *cardinality* of C is the number of vertices contained in C, denoted by $|C|$. The objective of the maximum clique problem (MCP) is to find a clique of maximum cardinality in G.

Here we give several notation used in this paper.

CC : the current clique.

PA : the vertex set of possible additions, i.e., the vertices that are connected to all vertices of CC.

OM : the vertex set of one edge missing, i.e., the vertices that are connected to $|CC|-1$ vertices of CC, provided that $CC \subseteq OM$.

$deg_{G(S)}(v)$: the degree of a vertex $v \in S$ in the subgraph $G(S)$, where $S \subseteq V$.

3. 1-OPT LOCAL SEARCH WITH ADD MOVE

One of the simplest moves used in the previously reported local search heuristics for the MCP is that a vertex $v \in PA$ (if PA is not empty) is added to the current clique or a vertex $v \in CC$ (if CC is not empty) is dropped from the current clique. Each of the add and drop moves can be interpreted as the 1-opt move because only a single vertex is moved to or from the current clique at each iteration. Although both the add and drop moves are adopted to achieve our k-opt local search described in the next section, we first consider a simple, 1-opt local search with only add move for the MCP.

Given a graph and an initial solution (clique) CC with PA ($\neq \emptyset$), a local search generally tries to enlarge the current clique *one by one* per each iteration. To enlarge the clique CC, one of the simplest operations may be to add a vertex $v \in PA$ by e.g., a random selection without other considerations. However, the resulting clique may be poor.

When a vertex is added to the current clique, we take into account the *degree* of vertex because the vertices with larger degree are usually promising in that it is likely to lead to a larger subset of mutually adjacent nodes in a graph. In the local search, we select a vertex v with the maximum degree in the current subgraph induced by the set PA instead of the original graph G. The vertex v with the maximum degree can be found by scanning each of $deg_{G(PA)}(v)$, $\forall v \in PA$. If multiple vertices with the same maximum degree are found, a random selection is executed among them. Afterward, the selected v is added to CC and the information of PA and $deg_{G(PA)}$ are updated step-wise for the next iteration. Such add moves are repeated until no better clique is found, i.e., until PA is empty. The 1-opt local search is shown in Figure 1 (we suppose that CC, PA and $deg_{G(PA)}$ are given in advance).

4. K-OPT LOCAL SEARCH

The larger sized neighborhoods such as k-opt neighborhood for the MCP may yield better local optima but the effort needed to search the neighborhood is too computationally expensive. The idea of the variable depth search is based on efficiently searching a small fraction of the large neighborhood.

The *variable depth search* (VDS) was successfully applied by Lin and Kernighan to the traveling salesman problem [11] and graph partitioning problem [10]. Their heuristic methods are known to be one of the most powerful local search procedures for the two combinatorial optimization problems. Recently, for the both problems the VDS based heuristics have been incorporated into several metaheuristics such as evolutionary algorithm [14] and iterated local search algorithm [1, 7, 9]. Generally, the performance of the metaheuristics embedded with the VDS is remarkably effective for the problems. For other hard problems, Yagiura et al. [16] proposed an effective VDS algorithm for the generalized assignment problem. For the unconstrained binary quadratic programming problem, an effective local search based on VDS was proposed by Merz and Freisleben [15] and then its variant was shown by Katayama *et al* [8].

In this section, we show an effective local search based on VDS, called the k-opt local search, for the MCP. In the k-opt local search, the k-opt neighborhood is defined by the set of solutions obtained by a sequence of several 1-opt moves that add and drop moves are adaptively changed in the feasible search space. The length t of the sequence is adaptively changed in the algorithm, and all t solutions obtained by the sequence are different because we assure that *cycling*

```
   procedure k-opt-Local-Search(CC, PA, OM, deg_{G(PA)})
     begin
1      repeat
2        CC_prev := CC, D := CC_prev, P := {1, ..., n}, g := 0, g_max := 0;
3        repeat
4          if |PA ∩ P| > 0 then       // Add Phase
5            find a vertex v with max_{v∈{PA∩P}} { deg_{G(PA)}(v) };
6            if multiple vertices with the same maximum degree are found then
               select one vertex v among them randomly;
7            CC := CC ∪ {v}, g := g + 1, P := P\{v};
8            if g > g_max then g_max := g, CC_best := CC;
9          else       // Drop Phase (if {PA ∩ P} = ∅)
10           find a vertex v ∈ {CC ∩ P} such that the resulting |PA| is maximized;
11           if multiple vertices with the same size of the resulting |PA| are found then
               select one vertex v among them randomly;
12           CC := CC\{v}, g := g − 1, P := P\{v};
13           if v is contained in CC_prev then D := D\{v};
14         endif
15         update PA, OM, and deg_{G(PA)}(i), ∀i ∈ PA;
16       until D = ∅;
17       if g_max > 0 then CC := CC_best else CC := CC_prev;
18     until g_max ≤ 0;
19     return CC;
     end;
```

Figure 2: The pseudo code of the k-opt local search for the MCP

among the solutions in the sequence is avoided. The basic procedure of the k-opt local search is as follows.

Given an initial feasible solution (clique) CC with PA ($\neq \emptyset$), in each iteration the local search starts with the solution $CC^{(0)}$ (equal to CC) with $PA^{(0)}$ (we here set $t = 0$). The first move from $CC^{(0)}$ to $CC^{(1)}$ is performed by the add move with the same manner of the 1-opt local search described above. Thus, the solution $CC^{(0)}$ becomes $CC^{(1)} := CC^{(0)} \cup \{v\}$, where v is a vertex with the maximum degree in $G(PA^{(0)})$, and $PA^{(0)}$ is updated to be $PA^{(1)}$ (Note that other information, OM and $deg_{G(PA)}$, is also updated). Such a phase of the add moves is repeated until $PA = \emptyset$. Now we assume that $PA = \emptyset$ after some add moves. It is obvious that the add move is impossible for the current clique CC. However, the k-opt neighborhood search tries to find better cliques after dropping a vertex or few vertices from the current clique.

The drop phase starts from $CC^{(l)}$ with no possible additions. This phase aims at dropping a vertex $v \in CC^{(l)}$ where as many as possible of the vertex set of possible additions (i.e., $|PA^{(l+1)}|$) that are connected to the member of $CC^{(l)} \backslash \{v\}$ occur. The vertex v can be found by checking the number of all nodes that are lacking the edge to v in $OM^{(l)}$.

Let us assume that a vertex v to be dropped from the current clique is found. The vertex v is dropped from CC. Then we update the information of PA, etc. If a vertex addition to CC is possible after the drop phase, then the add phase immediately starts again, otherwise the drop phase is continued. The k-opt neighborhood search which consists of the two phases is repeated until a termination condition is satisfied.

The information of PA, etc. is updated whenever a single vertex is moved. The updating technique and data structure used in the local search are derived from the literature [2].

At the final stage in each iteration, t solutions, i.e., $CC^{(1)}$, $CC^{(2)}, \ldots, CC^{(t)}$, obtained by the sequence of the 1-opt moves are obtained. From the t solutions, we choose the best one $CC^{(k)}$ ($1 \leq k \leq t$) as a new initial feasible solution setting $CC^{(0)} := CC^{(k)}$ for the next k-opt neighborhood search. The k-opt local search is repeated until no better solution is found.

Figure 2 shows the pseudo code of the k-opt local search for the MCP (here we suppose that CC, PA, OM, and $deg_{G(PA)}$ are given in advance). In this figure, we use a *gain* denoted by g (see line 2), where g is defined by the difference between the clique size of the given solution and that of the current one during the search. In the local search, a candidate set P (see line 2) is used to assure that the cycling among the solutions in the sequence is avoided.

The local search has inner and outer loops. In the inner loop (lines 3–16), a sequence of the add and drop moves for the given initial solution is obtained with the restriction due to the set P, and the best solution found by the sequence is selected. In the outer loop (lines 1–18), the best solution selected in the inner loop is evaluated with the maximum gain g_{max} whether it is better than the previous one ($|CC_{prev}|$).

The termination condition that we set for the inner loop (line 16) is when the set D is empty. Initially D is set to the vertex set of the initial clique CC (i.e., CC_{prev}) given at line 2 before the search in the inner loop starts. A vertex v, which is dropped in the drop phase, is removed from D if v is in CC_{prev} at line 13.

Note that no parameter setting is required in the k-opt local search shown in this paper for the MCP.

5. EXPERIMENTAL RESULTS

Computational experiments were conducted to show the

effectiveness of the k-opt local search for the MCP. In the experiments, we repeatedly applied the local search with different starting solutions for each of the well-known 37 DIMACS benchmark graphs [3]. We carried out such a multi-start method 10 runs for each graph on a Pentium IV PC (2.0GHz). The algorithm code was compiled with **gcc** compiler using optimization flag **-O2**.

The multi-start method with the k-opt local search is executed as follows. In each of the multi-start method runs (with different random number generator seeds), the local search is repeatedly executed n times, where n is the number of nodes of a given graph G, and the best clique found among n local searches is output as a result. Each of n initial solutions for each multi-start run is given simply: a feasible (poor) clique with a single vertex randomly chosen from n nodes without duplicate in a graph G. Therefore, every n initial solution is the same with every one used in each run. However, since the k-opt local search performs a random selection when tie-breaking choice, it is expected that the final solution or search is different even if the same initial solution is given for the local search.

Table 1 summarizes the results of the multi-start k-opt local search for the 37 benchmark instances. In the table, the best clique size found in the 10 runs of the multi-start method, the average clique size with standard deviation (s.dev.), the worst clique size, and the average running time with (s.dev.) in seconds consumed until the best solution is found in each run are shown for each instance. BR denotes the best result of all DIMACS workshop participants (* if optimality is proved). For a comparison, we show the published average results of the state-of-the-art metaheuristic algorithms: genetic local search `GENE` [12] and iterated local search `ITER` [12]. In addition, we give the average results of the reactive local search `RLS` proposed by Battiti and Protasi [2] that is known to be another effective metaheuristic that employs a sophisticated search strategy based on the idea of tabu search. It is known that the RLS finds the best known solution in almost all the DIMACS graphs in reasonable running time.

We observed that the k-opt local search was capable of finding the satisfactory clique that was better than or corresponded to BR for 28 out of 37 cases even under the simple framework of the multi-start method described above. In particular, among the 28 graphs, the local search obtained better solutions (in bold style) of the clique size **77** for `C2000.9` and of **1099** for `MANN_a81` than BR.

It seems to be competitive with `RLS` in terms of the average solution results for 25 graphs (indicated with "•" at the column of `RLS`), i.e., 2 graphs of `DSJC`, 3 graphs of `MANN`, 2 graphs of `hamming`, 9 graphs of `p_hat`, etc. However, for the remaining 12 graphs `RLS` obtained better average results than our local search slightly.

`GENE` and `ITER` are known to be one of the most effective metaheuristics based on evolutionary algorithm incorporating local search for the MCP. However, in spite of the absence of major metaheuristic components, the k-opt local search is capable of finding better solutions (or at least the same solutions) on average than the metaheuristics of `GENE` and `ITER` for all the graphs. It is observed that the graphs for which the performance of `GENE` or `ITER` seems to be competitive with that of ours are relatively small, e.g., `C125.9`, `MANN_a27`, `hamming8-4`, `keller4`, `p_hat300-1`, and `p_hat300-2`.

It is difficult to strictly compare the running time of our multi-start k-opt local search with those of the other algorithms due to different computer, OS, programming language, etc. However, it is possible to roughly compare their running times in order to show that our local search may be relatively efficient. For example, for `p_hat300-1` that is a graph for which each of the four algorithms of k-opt local search, `GENE`, `ITER`, and `RLS` is capable of reaching the optimum solution in all runs of their tests, average running times to the best were as follows: our algorithm took 0.007 (s) on Pentium IV machine (2.0GHz), `GENE` and `ITER` needed 0.9 (s) and 0.4 (s) on Sun Ultra 250 (UltraSPARC-II 400MHz), respectively, and `RLS` spent 0.018 (s) on Pentium II workstation (450 MHz). As one of the largest graphs in DIMACS, for `MANN_a81`, our algorithm took 93.3 (s), `GENE` and `ITER` needed 2773.8 (s) and 693.9 (s), respectively, and `RLS` required about 2830.8 (s) on their computers described above.

6. CONCLUSION

The variable depth search (VDS) based algorithms have been proposed and known to be considerably effective for several combinatorial optimization problems. Thus, it is worth while considering such a local search algorithm for solving the other problems. With this motivation, this paper presented an effective heuristic algorithm inspired by VDS, the k-opt local search, for the maximum clique problem.

Computational experiments were conducted on the DIMACS benchmark instances. The results showed that the k-opt local search was capable of reaching state-of-the-art results even in the absence of major metaheuristic components. Particularly, the local search outperformed the metaheuristics of genetic local search and iterated local search reported recently in terms of obtainable solutions on average for all the instances studied.

The k-opt local search has a more *significant* advantage in that it is suited for using as a local search engine without modification for metaheuristics such as memetic algorithm, iterated local search, GRASP, etc. Since better results than the effective metaheuristics are obtained even if the k-opt local search starts with a poor initial clique, a metaheuristic approach based on the k-opt local search is much more promising for solving the maximum clique problem.

7. REFERENCES

[1] D. Applegate, W. Cook, and A. Rohe, "Chained Lin-Kernighan for large traveling salesman problems," INFORMS Journal on Computing, vol. 15, no. 1, pp.82–92, 2003.

[2] R. Battiti and M. Protasi, "Reactive local search for the maximum clique problem," Algorithmica, vol. 29, no. 4, pp. 610–637, 2001.

[3] DIMACS URL: `http://dimacs.rutgers.edu/Challenges/`.

[4] M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, New York, 1979.

[5] J. Hastad, "Clique is hard to approximate within $n^{1-\epsilon}$," Acta Mathematica, vol. 182, pp. 105–142, 1999.

[6] D.S. Johnson and L.A. McGeoch, "The traveling salesman problem: A case study," Local Search in Combinatorial Optimization, John Wiley & Sons, pp. 215–310, 1997.

Table 1: Results of the k-opt local search and the state-of-the-art metaheuristic algorithms (RLS by Battiti and Protasi, GENE and ITER by Marchiori) for the DIMACS benchmark instances

DIMACS benchmarks		k-opt Local Search						GENE [12]	ITER [12]	RLS [2]
Instance	BR	Best	Avg	(s.dev.)	Worst	Time(s)	(s.dev.)	Avg	Avg	Avg
C125.9	34*	34	34.0	(0)	34	0.001	(0.003)	33.8	34.0	• 34.0
C250.9	44*	44	44.0	(0)	44	0.062	(0.070)	42.8	43.0	• 44.0
C500.9	57	57	56.1	(0.300)	56	0.904	(0.526)	52.2	52.7	57.0
C1000.9	67	67	66.0	(0.447)	65	6.930	(6.413)	61.6	61.6	68.0
C2000.9	75	**77**	75.1	(0.943)	74	72.688	(48.807)	68.2	68.7	77.6
DSJC500.5	14*	13	13.0	(0)	13	0.133	(0.081)	12.2	12.1	• 13.0
DSJC1000.5	15*	15	15.0	(0)	15	6.351	(3.0730)	13.3	13.5	• 15.0
C2000.5	16	16	16.0	(0)	16	14.306	(13.618)	14.2	14.2	• 16.0
C4000.5	18	17	17.0	(0)	17	71.735	(58.982)	15.4	15.6	18.0
MANN_a27	126*	126	126.0	(0)	126	0.036	(0.033)	125.6	126.0	• 126.0
MANN_a45	345*	344	343.6	(0.490)	343	5.876	(6.877)	342.4	343.1	• 343.6
MANN_a81	1098	**1099**	1098.1	(0.300)	1098	93.300	(189.052)	1096.3	1097.0	• 1098.0
brock200_2	12*	11	11.0	(0)	11	0.024	(0.033)	10.5	10.5	12.0
brock200_4	17*	16	16.0	(0)	16	0.021	(0.022)	15.4	15.5	17.0
brock400_2	29*	25	24.6	(0.490)	24	0.308	(0.289)	22.5	23.2	26.1
brock400_4	33*	25	25.0	(0)	25	0.201	(0.168)	23.6	23.1	32.4
brock800_2	21	21	20.8	(0.400)	20	2.374	(2.938)	19.3	19.1	21.0
brock800_4	21	21	20.5	(0.500)	20	2.756	(2.684)	18.9	19.0	21.0
gen200_P0.9_44	44*	44	44.0	(0)	44	0.073	(0.080)	39.7	39.5	• 44.0
gen200_P0.9_55	55*	55	55.0	(0)	55	0.003	(0.006)	50.8	48.8	• 55.0
gen400_P0.9_55	55	53	52.3	(0.640)	51	0.618	(0.470)	49.7	49.1	55.0
gen400_P0.9_65	65	65	65.0	(0)	65	0.274	(0.203)	53.7	51.2	• 65.0
gen400_P0.9_75	75	75	75.0	(0)	75	0.186	(0.160)	60.2	62.7	• 75.0
hamming8-4	16*	16	16.0	(0)	16	0.002	(0.004)	16.0	16.0	• 16.0
hamming10-4	40	40	40.0	(0)	40	0.642	(0.507)	37.7	38.8	• 40.0
keller4	11*	11	11.0	(0)	11	0.001	(0.003)	11.0	11.0	• 11.0
keller5	27	27	27.0	(0)	27	0.081	(0.053)	26.0	26.3	• 27.0
keller6	59	57	55.5	(0.671)	55	137.398	(153.039)	51.8	52.7	59.0
p_hat300-1	8*	8	8.0	(0)	8	0.007	(0.013)	8.0	8.0	• 8.0
p_hat300-2	25*	25	25.0	(0)	25	0.006	(0.009)	25.0	25.0	• 25.0
p_hat300-3	36*	36	36.0	(0)	36	0.030	(0.018)	34.6	35.1	• 36.0
p_hat700-1	11*	11	11.0	(0)	11	0.636	(0.378)	9.8	9.9	• 11.0
p_hat700-2	44*	44	44.0	(0)	44	0.054	(0.051)	43.5	43.6	• 44.0
p_hat700-3	62	62	62.0	(0)	62	0.088	(0.083)	60.4	61.8	• 62.0
p_hat1500-1	12*	12	12.0	(0)	12	16.967	(12.500)	10.8	10.4	• 12.0
p_hat1500-2	65	65	65.0	(0)	65	0.464	(0.465)	63.8	63.9	• 65.0
p_hat1500-3	94	94	94.0	(0)	94	2.300	(2.402)	92.4	93.0	• 94.0

[7] D.S. Johnson, "Local optimization and the traveling salesman problem," Proc. 17th. International Colloquium on Automata, Languages and Programming, pp. 446–461, 1990.

[8] K. Katayama, M. Tani, and H. Narihisa, "Solving large binary quadratic programming problems by effective genetic local search algorithm," Proc. of the 2000 Genetic and Evolutionary Computation Conference, pp. 643–650, 2000.

[9] K. Katayama and H. Narihisa, "Iterated local search approach using genetic transformation to the traveling salesman problem," Proc. of the Genetic and Evolutionary Computation Conference, vol. 1, pp. 321–328, 1999.

[10] B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," Bell System Technical Journal, vol. 49, pp. 291–307, 1970.

[11] S. Lin and B.W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," Operations Research, vol. 21, pp. 498–516, 1973.

[12] E. Marchiori, "Genetic, iterated and multistart local search for the maximum clique problem," Applications of Evolutionary Computing, Springer, LNCS 2279, pp. 112–121, 2002.

[13] E. Marchiori, "A simple heuristic based genetic algorithm for the maximum clique problem," Proc. of ACM Symposium on Applied Computing (SAC-98), pp. 366–373, 1998.

[14] P. Merz and B. Freisleben, "Memetic algorithms for the traveling salesman problem," Complex Systems, vol. 13, no. 4, pp. 297–345, 2001.

[15] P. Merz and B. Freisleben, "Greedy and local search heuristics for unconstrained binary quadratic programming," Journal of Heuristics, vol. 8, no. 2, pp. 197–213, 2002.

[16] M. Yagiura, T. Yamaguchi and T. Ibaraki, "A variable depth search algorithm for the generalized assignment problem," Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, (S. Voss, S. Martello, I.H. Osman and C. Roucairol, eds.), Kluwer Academic Publishers, pp. 459–471, 1999.