A Low-Level Hybridization between Memetic Algorithm and VNS for the Max-Cut Problem

Abraham Duarte
ESCET-URJC
Campus de Móstoles
28933 Madrid, Spain
+34 914888116
abraham.duarte@urjc.es

Ángel Sánchez
ESCET-URJC
Campus de Móstoles
28933 Madrid, Spain
+34 91 6647452
angel.sanchez@urjc.es

Felipe Fernández
FI-UPM
Campus de Montegancedo
28660 Madrid Spain
+34 91 3367371
Felipe.Fernandez@es.bosch.com

Raúl Cabido
ESCET-URJC
Campus de Móstoles
28933 Madrid, Spain
+34 91 4887190
rcabido@gmail.com

ABSTRACT

The Max-Cut problem consists of finding a partition of the graph nodes into two subsets, such that the sum of the edge weights having endpoints in different subsets is maximized. This NP-hard problem for non planar graphs has different applications in areas such as VLSI and ASIC design. This paper proposes an evolutionary hybrid algorithm based on low-level hybridization between Memetic Algorithms and Variable Neighborhood Search. This algorithm is tested and compared with the results, found in the bibliography, obtained by other hybrid metaheuristics for the same problem. Achieved experimental results show the suitability of the approach, and that the proposed hybrid evolutionary algorithm finds near-optimal solutions. Moreover, on a set of standard test problems, new best known solutions were produced for several instances.

Categories & Subject Descriptors: G.2.1 [**Discrete Mathematics**]: Combinatorics – *combinatorial algorithms*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *heuristic methods, graph and tree search algorithms*

General Terms: Algorithms, Experimentation.

Keywords: Max-Cut, Metaheuristic, Evolutionary Algorithms, Memetic Algorithms, VNS.

1. INTRODUCTION

An important graph bipartition problem is the Max-Cut problem defined for a undirected weighted graph $S = (V, E, W)$, where V is the set of vertices or nodes ($|V| = n$), E is the set of undirected arcs or edges ($|E| = m$), and W is the set of edge weights.

The Max-Cut optimization problem consists in finding a partition of the set V into two disjoint subsets (C, C') such that the sum of the weights of edges with endpoints in different subsets is maximized. Every partition of vertices V into C and C' is usually called a *cut* or *cutset* and the sum of the involved edges weights is called *weight of the cut* or *cut value*.

The considered Max-Cut optimization problem is given by the maximization of the cut value:

$$w(C, C') = \sum_{v \in C, u \in C'} w_{vu}$$

where w_{uv}, is the weight of edge $(u,v) \in V$.

Reference [13] proves that the decision version of Max-Cut problem is NP-Complete. Therefore, it is convenient to devise algorithms for finding an approximate solution to this problem in a reasonable time. Notice that for planar graphs exact algorithms can solve the Max-Cut problem in polynomial time [15].

Figure 1 shows a cut example for an undirected graph. The cut edges are represented with thick lines. Assuming that all edges have the same weight, that is equal to one, the cut value shown in Figure 1 is 6.

Some practical applications of the Max-Cut problem can be found in diverse fields like VLSI design [3], statistical physics [1] and other related to combinatorial optimization [25]

Several continuous, linear programming and semidefinite relaxations for the Max-Cut have been proposed to achieve high quality solutions in a reasonable running time. Among these alternatives, the most suitable is the semidefinite relaxation (SDP) because it is solvable in polynomial time. Moreover, this SDP value establishes an upper bound of optimal cut values [5]. It can be used to test the performance of approximate algorithms for the referred problem.

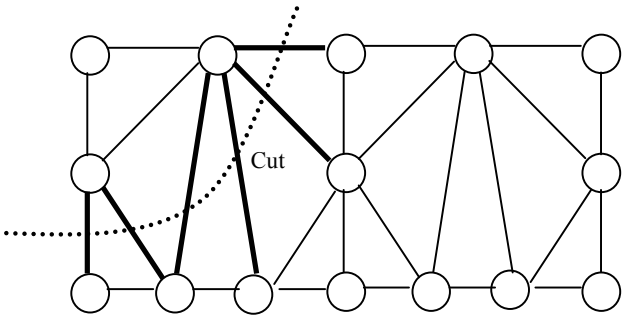

Figure 1. Cutset example on an example graph.

Reference [23] describes a semidefinite relaxation of the Max-Cut problem. Goemans et al. [9] proposed a randomized algorithm that guarantees a 0.878-approximation to the optimum and, in addition, an upper bound on the optimum. A very interesting rank-2 relaxation algorithm is proposed in [2] that gives, in mean, better solutions than other theoretical relaxations [5].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'05, June 25-29, 2005, Washington, DC, USA.

Copyright 2005 ACM 1-59593-010-8/05/0006…$5.00.

In this paper we propose a new evolutionary algorithm based on low-level hybridization [24] between a memetic algorithm [10][19][20] and Variable Neighborhood Search (VNS) [16][18] metaheuristic for finding an approximate solution to the Max-Cut problem. In order to evaluate the performance of our approach, we compare results produce by own algorithm with other reported by different hybrid metaheuristics for the same problem and with the same benchmarks. The proposed metaheuristic is based on a genetic algorithm with an additional Variable Neighborhood Search (VNS) based on the problem domain knowledge. This hybrid algorithm achieves a remarkable improvement of the obtained solution.

The rest of the paper is organized as follows. Section 2 revises other algorithms and metaheuristics applied to the Max-Cut problem. An overview of evolutionary algorithms is shown in Section 3. In Section 4 the general optimization strategy is described. Section 5 introduces the main characteristics of Variable Neighborhood Search. A detailed description of the proposed algorithm is presented in Section 6. Experimental results are shown in Section 7. Finally, Section 8 concludes the paper.

2. MAX-CUT ALGORITHMS REVIEW

Several approaches have been proposed to the Max-Cut problem. Probably the most famous algorithm (in graph partition context) is the Lin-Kernigham algorithm [14], based on the improvement of a solution by exchanging nodes between different subsets defined by the cut. This algorithm ensures that both subsets have the same size. In Fiduccia and Mattheyses [7] work this restriction is relaxed. Note that almost all local search strategies are based on these two algorithms [17].

Taking into account SDP relaxation, the work of Shani and Gonzalez [22] can be considered the first algorithm which achieves an acceptable result (0.5-approximation to the optimum). Based on this work, Goemans et al. [9] proposed an SDP relaxation which obtains a 0.878-approximation to the optimum. This algorithm achieves good results for small graphs but usually its performance is affected for medium or large size graphs [4][11][17].

Gosti et al. [11] incorporates a local search strategy in Goemans algorithm, getting higher quality solutions. Based on this idea U. Feige et al. have developed an algorithm that ensures a 0.921-approximation to the optimum for graph whose nodes have limited its degree.

An experimental study of six Max-Cut algorithms is presented in [4]. These are respectively based on an integer relaxation, a random algorithm, two heuristic algorithms, a genetic algorithm and a Divide-and-Conquer algorithm. From this experimental study is concluded that to achieve high quality solutions, the use of local search strategies is required.

Kim el al. [17] proposes a hybrid genetic algorithm (memetic algorithm) for the Max-Cut problem. This work is based on a standard genetic algorithm with several local search strategies (inspired in Fiduccia-Matheysses algorithm [7]).

Festa et al. [5] present an experimental study of three metaheuristics (GRASP, VNS and PR) for the considered problem. These authors propose several hybridizations among these metaheuristics, which usually achieves very good results.

Duarte et al. [6] describe a hierarchical social which is compared with other GRASP and memetic implementations.

Among all revised algorithms, the rank-2 relaxation algorithm [2] is in mean the method which obtains the best solution and, surprisingly, spending lower time than other SDP relaxations [2][5].

3. EVOLUTIONARY ALGORITHMS

Genetic Algorithms (GA) [10][19] are random search algorithms inspired by the Darwinian model of natural evolution. Potential solutions are coded by a chromosome structure, called individual. The set of individuals is called population. In order to solve an optimization problem, GA successively transform the population by means of random operators (selection, crossover and mutation) that generally increases the quality of the corresponding solutions (coded by individuals).

Unlike traditional GAs, Memetic Algorithms (MA) [20] are intrinsically concerned with exploiting available knowledge about the problem under study. This approach is not an optimal mechanism but, in general, yields to an algorithm enhancement. Optimization is accomplished in MA framework by incorporating problem dependent heuristics: approximation algorithms, local search techniques, specialized recombination operators, truncated exact methods, etc. Moreover, MAs can be additionally improved by means of a low-level or high-level hybridization [20] with other metaheuristics.

Evolutionary algorithms (EA) are a broad class of metaheuristics characterized algorithmically by:

- **Population**: of individuals which represent partial or complete solutions.
- **Selection method**: that selects individuals in a slant fashion. The best individuals have the higher selection probability.
- **Modification method**: that generates new individuals by means of a stochastic operator application. It can be:
  - *Unary*: that create a new individual slightly modified from an old individual (mutation)
  - *m-ary*: that create new individuals by means of the combination of *m*-individuals

In this sense, EA includes as particular cases MA and GA and other metaheuristics such that Cultural Algorithms or Swarm Intelligence.

4. OPTIMIZATION STRATEGY

EA are metaheuristics where almost all implementation effort comes from the search diversification. On the other hand, strategies such that VNS and their variants [16][18] focus almost entirely on the search intensification. With respect to this fact, metaheuristic such that EA and VNS can be considered as complementary algorithms. Moreover, the hybridization of these techniques can yield to very effective and robust methods.

This paper proposes a new evolutionary algorithm based on a low-level hybridization [24] between a specific MA, developed for the Max-Cut Problem and Variable Neighborhood Search (VNS) metaheuristic. The developed algorithm is a good trade-off

between intensification and diversification. The intensification phase is mainly carried out by the VNS procedure. This metaheuristic intensively looks for quality solutions in a predefined set of neighborhood structures. If the search procedure is stuck, VNS changes the neighborhood structure in order to get away from local optimum. Notice that although the main task of VNS is the search intensification, this metaheuristic also diversifies the search procedure by means of neighborhood changing.

On the other hand, EA objective is mainly related with the diversification stage. This task is accomplished with traditional operators (selection, mutation and crossover) enriched with some knowledge about the problem. Notice that although the main task of MA is the search diversification, this metaheuristic also intensifies the search procedure by means of population evolution and the inclusion of problem-dependent operators.

In the following two sections, we respectively describe the VNS and the hybrid MA implementations, developed for the Max-Cut problem.

5. VARIABLE NEIGHBORHOOD SEARCH

This section resumes the main features of Variable Neighborhood Search (VNS) metaheuristic. This metaheuristic, which was originally proposed by Hansen and Mladenovíc [16][18], is based on the exploration of a dynamic neighborhood model. Each step has three major phases: neighbor generation, local search and jump.

Unlike to other metaheuristics based on local search methods, VNS allows changes of the neighborhood structure during the search. VNS explores increasingly neighborhoods of the current best found solution *x*. The basic idea of VNS is to change the neighbourhood structure when the local search is trapped on a local minimum.

Let N_k, $k = 1,\ldots, k_{max}$ be a set of predefined neighborhood structures and let $N_k(x)$ be the set of solutions in the k^{th}-order neighborhood of a solution *x*. In the first phase, a neighbor $x' \in N_k(x)$ of the current solution is applied. Next, a solution *x''* is obtained by applying local search to *x'*. Finally, the current solution jumps from *x* to *x''* if it improves the former one. Otherwise, the order *k* of the neighborhood is increased by one and the above steps are repeated until some stopping condition is met. The pseudo-code of a typical VNS procedure is illustrated in Figure 2.

In the case of the Max-Cut problem, the k^{th}-order neighborhood is defined by all solutions that can be derived from the current one by selecting *k* vertices and transferring each vertex from one subset of the vertex bipartition to the other subset.

The local search phase is based on the following neighborhood structure. Let (*Ca, Ca'*) be the current cutset solution. For each vertex $v \in V$ we associate a new neighbor cutset (*Cb, Cb'*):

$$(Cb, Cb') = N_1(Ca, Ca') = \begin{cases} Ca - \{v\}; Ca' + \{v\} & if \quad v \in Ca \\ Ca + \{v\}; Ca' - \{v\} & if \quad v \in Ca' \end{cases}$$

We define for each node $v \in V$ the functions σ and σ' as:

$$\sigma(v) = \sum_{u \in C} w_{vu} \qquad \sigma'(v) = \sum_{u \in C'} w_{vu}$$

These two functions are characterized by the change in the objective function value associated with moving vertex *v* from one subset of the cut to the other. This way, a vertex makes a movement in order to improve the cut value in the two following situations:

$$if \quad v \in C \quad \wedge \quad \sigma(v) > \sigma'(v) \quad then \quad C \xrightarrow{v} C'$$
$$if \quad v \in C' \quad \wedge \quad \sigma(v)' > \sigma(v) \quad then \quad C' \xrightarrow{v} C$$

where $C \xrightarrow{v} C'$ (equivalently $C' \xrightarrow{v} C$) represents the movement of vertex *v* from subset *C* to *C'* ($C \cup C' = V$)

All possible moves are examined. The current solution is replaced by its best improving neighbor solution. The search stops after all possible moves have been evaluated and no improving neighbor is found. The used local search strategy is summarized by the pseudo-code of Figure 2.

```
procedure VNS(x)
  var
    x: Initial solution
    x',x'': Intermediate solutions
    k: Neighbourhood order
  begin
  /*First Neighbourhood Structure*/
  k = 1;
  while k < kmax do
    /*Select an random solution in k-
    neighbourhood structure*/
    x'  = Random(x,Nk(x))
    /*Use the local search procedure shown
    in Figure 3*/
    x'' = LocalSearch(x');
    /*Replace the actual solution by the new
    one when an improvement is obtained */
    if w(x'') > w(x) then
      x = x'';
      k = 1;
    else
      k = k + 1;
    end if
  end while
  end
end VNS
```

Figure 2. VNS high level pseudo-code

This local search procedure tests all possible movements for each node between *C* and *C'* and vice versa. Therefore, the current solution is replaced by the best solution found in the neighborhood structure defined above. The procedure ends when none possible neighbor movement improves the current solution.

6. HYBRID METAHEURISTIC

This section describes a new evolutionary low-level hybridization for the Max-Cut problem. In order to use a memetic algorithm for solving the Max-Cut problem, we need to code each feasible solution. Let *V= {1,...,n}* the nodes set of a given graph. The possible cuts on this graph can be coded by a Boolean *n*-vector *I* = $(i_1,\ldots,i_n)$ such that the value of each component $i_u \in \{0,1\}$ is given by the characteristic function:

$$i_u = \begin{cases} 1, & if \quad u \in C \\ 0, & if \quad u \in C' \end{cases}$$

```
procedure Local_Search(g)
  var
    g=(C,C´): Cutset structure
  begin
  for v = 1 to Nodes_in_considered_graph do
    if v∈C and σ(v)>σ′(v) then
      /* v: C→C´ */
       C = C \ {v};
       C´= C′ ∪ {v};
    end if
    if v∈C´and σ(v)<σ′(v) then
      /* v: C′→C */
      C′= C′\{v};
      C = C ∪ {v};
    end if
  end for
  end
end Local_Search
```

Figure 3. Local search high level pseudo-code

Figures 4.a and 4.b show two examples of cuts and their respective encoding.

7. HYBRID METAHEURISTIC

This section describes a new evolutionary low-level hybridization for the Max-Cut problem. In order to use a memetic algorithm for solving the Max-Cut problem, we need to code each feasible solution. Let $V= \{1,...,n\}$ the nodes set of a given graph. The possible cuts on this graph can be coded by a Boolean n-vector $I = (i_1,...,i_n)$ such that the value of each component $i_u \in \{0,1\}$ is given by the characteristic function:

$$i_u = \begin{cases} 1, & if \quad u \in C \\ 0, & if \quad u \in C' \end{cases}$$

Figures 4.a and 4.b show two examples of cuts and their respective encoding.

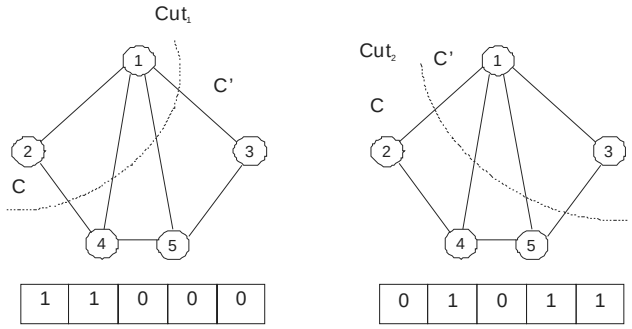

Figure 4. Cutsets representation

In the evaluation step, we used as selection method the fitness roulette-wheel selection [10][19], which favors individuals with high fitness value without suppressing the chance of selection of individuals with low fitness, thus avoiding premature convergence of the population.

The proposed algorithm starts with a random initial population of cuts, generated by *Initial_Population* procedure. Then, these cuts are improved (with probability p_i) by means of a local search procedure described in Figure 3.

The selection of a subset of individuals in the implemented genetic algorithm is carried by means of a standard roulette wheel procedure. Some selected individuals are crossovered, with a probability p_r. In the proposed implementation, we have not use standard crossover because this method can destroy the high quality structures obtained by means of evolution. We have considered fixed crossover [4][19], which takes into account the structural information of each individual and provides more quality descendants [19]. Graphically, the crossover strategy is presented in the Figure 5.

The considered fixed crossover $f:\{1,0\}\times\{1,0\}\rightarrow\{1,0\}$ is specified by the random Boolean function:

$$f(\alpha,\beta) = \begin{cases} 0 & if \quad \alpha = 0 \ \wedge \ \beta = 0 \\ 1 & if \quad \alpha = 1 \ \wedge \ \beta = 1 \\ rand01() & if \quad \alpha \neq \beta \end{cases}$$

where *rand01*() is a random Boolean value. In this way, if both parents are in the same subset, the offspring node lies in this subset. Otherwise, the node is randomly assigned to one of the subsets.

With this crossover function, each bit i_u of new offspring is given by:

$$i_u = f(father(i_u), mother(i_u)) \qquad \forall u \in V$$

To end up the evolution cycle, new individuals are subject to mutation (a random change of a node from C to C' or vice versa) with probability $p_m=1/|V|$. By this way, the allele mutation probability (p_m) is problem independent.

Figure 6 shows the high level pseudo-code of the corresponding hybrid evolutionary algorithm.

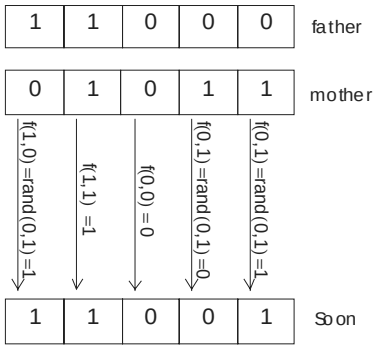

Figure 5. Fixed crossover procedure

8. EXPERIMENTAL RESULTS

This section describes the obtained experimental results using the proposed hybrid metaheuristic. We also show a quantitative comparison with other metaheuristics applied to the same problem.

The computational experiments were performed in an Intel Pentium 4 processor at 1.7 GHz, with 256 MB of RAM. All algorithms were coded in C++, without optimization, and by the same programmer in order to have more comparable results.

```
procedure Hybrid_Evolutionary_Algorithm()
  var

g=(C,C´): individual cutset structure
    gg: population of cutsets
    MaxGen: Number of Generations
    PopSize: Number of individuals
    pc,pm: Cross. and mut. probabilities
    pi: Improvement probability
    i: Generation Counter
    j: Individual Counter
  begin
    /*Generate random cuts individuals*/
    gg=Initial_Population();
    /*Optimize initial population*/
    Apply(Local_Search(),pi);
    Evaluate_Population();
    Best_Solution = Best_Individual();
    for i = 1 to MaxGen
      j = 0;
      while(i < PopulationSize)
        /*Criteria: Random Wheel*/
        Father = Selection();
        r = rand01();/*Random function*/
        if (r < pc)
          /*Criteria: Random Wheel*/
          Mother = Selection();
          Child=FixCross(Father,Mother);
          Apply(VNS(Child),pi);
          InsertInPopulation(Child);
          j = j + 1;
        else
         InsertInPopulation(Father);
      end if
    end while
    Apply(Mutation(),pm);
    Evaluate_Population();
    Best_Solution = Best_Individual();
  end for
end
end  Hybrid_Evolutionary_Algorithm()
```

Figure 6. Hybrid algorithm high level pseudo-code.

The main parameters and corresponding using values of the designed hybrid evolutionary metaheuristic are:

- **Memetic algorithm**:
  - Initial random population of 50 individuals, called *PopSize*.
  - *PopSize* initially is improved by means of the referred local search strategy (in Figure 3) with a probability $p_i = 0.25$.
  - The probability of crossover p_c is 0.6 and it is performed by *FixedCrossover* method.
  - The maximum number of generations *MaxGen* is 50.
  - After the crossover, the new individuals are also improved by the described VNS strategy with a probability $p_i = 0.25$.
  - In each generation, a mutation process is applied with a probability $p_m = 1/|V|$.
  - The procedure ends when none individual improves its fitness or it is reached the *MaxGen* value
- **Variable Neighborhood Search**
  - Each child obtained after a fixed crossover application is the *Initial Solution* for VNS procedure.
  - The maximum neighborhood order k_{max} is set to 1% of the number of nodes in the graph.

The metaheuristic was tested on the benchmark graphs *Gx* shown in Table 1. These test problems were generated by Helmberg and Rendl using the graph generator described in [12]. These graphs are planar, toroidal and randomly generated with varying sparsity and size. The last two graphs types are non-planar. There are graphs with unitary, integer and real weights. Moreover, these weights can be positive or negative. In the experiments, the graph sizes vary from 800 nodes to 3000 and their density from 0.17% to 6.12%.

The first three columns of Table 1 respectively show the graph name, and their number of nodes (*n*) and arcs (*m*). The following five columns present respectively, for 50 independent iterations of the proposed algorithm, the following statistical values: mean (*mn*), standard deviation (*sd*), max value (*max*), min value (*min*) and frequency (*fq*). This last value gives the (maximum) number of times that the search procedure has found the same value.

The last two right columns show respectively the SDP value and a ratio between the maximum cutset value (obtained for each graph) and its SPD bound (for the same graph), given by the formula:

$$r = 1 - \frac{SDPValue\,(Gx) - MaxValue\,(Gx)}{SDPValue\,(Gx)}$$

This parameter *r* establishes a measurement of how close is the value obtained by our hybrid metaheuristic and the corresponding upper bound given by SDP value. As shows Table 1, the maximum obtained value achieves, at least, 88 % of the SDP bound.

Therefore, the solutions found with our approach have a high-quality. Moreover, the proposed algorithm is highly robust because the mean value is relatively high. This robustness is also confirmed with the following factors: the closeness between max and min value and the low value of standard deviation. It is important to remark that the obtained results are quite general because the graphs used in the experiments have a high variety.

The proposed algorithm converges to the same solution in very few occasions (see *fq* column). We can conclude that our proposal ensures a good search procedure diversification. This property is mainly relevant in problems with sharp space solutions.

Table 2 presents a quantitative comparison between our proposal and seven Max-Cut state-of-the-art algorithms. The first three columns show the name of the graph. Obviously the number of nodes *n* and the number of arcs *m* are the same that the graphs in Table 1.

The next column shows the achieved results with 0.612 version of circut rank-2 relaxation with default parameters except (*N*,*M*)=(50, 10), for intensifying the search [1].

The following seven columns display the results for the considered metaheuristics: GRASP, GRASP + Path Relinking (PR), GRASP + Variable Neighborhood Search (VNS), GRASP + VNS + PR, VNS and VNS + PR. A detailed description of these metaheuristics can be found in [5]. Note that all these approach are trajectorial metaheuristics, and they only consider one solution in each iteration. The results shown by these columns are the best found cut value in 1000 independent iterations of each metaheuristic. These results have been extracted from the work of Festa et al. [5].The 8th column shows the results obtained by our hybrid evolutionary algorithm. Finally, the right column shows the SDP value [5,9] that can be considered as an upper bound.

The best cutset value for each graph is highlighted in bold in Table 2. As shown by this table, the proposed low-level hybridization between memetic algorithm and VNS, obtains similar computational results compared with the rest of metaheuristics. The main different is that our algorithm is executed only once with a population of 50 individuals and 50 iterations. Remind that the other metaheuristics have been run 1000 times.

Our proposal obtain the best cut value known up to now for 6 graphs only with 50 individuals and 50 generations. And find the best cut value for 12 graphs. Moreover, in our VNS implementation k_{max} is 1% of number of nodes, so this value is bounded by $8 \leq k_{max} \leq 30$). In VNS implementations presented in Table 2, so k_{max} is 100. Notice that this value has a terrible impact in execution time because it increases hugely the procedure execution time.

Problem			**Statistic values in 50 iterations**						
Name	*Nodes (n)*	*Edges (m)*	*Mean (mn)*	*Standard Deviation*	*Max Value*	*Min Value ()*	*Frequency (fq)*	*r*	*SDP*
G1	800	19176	11608.28	14.31	11624	11576	11	0,9624	12078
G2			11528.78	7.87	11620	11581	1	0,9616	12084
G3			11606.30	10.15	11622	11585	6	0,9623	12077
G11	800	1600	555.92	2.86	562	550	1	0,8963	627
G12			547.68	3.15	554	542	1	0,8921	621
G13			527.52	2.70	580	566	1	0,8992	645
G14	800	4694	3052.00	4.19	3061	3043	5	0,9605	3187
G15			3036.26	4.65	3046	3028	3	0,9612	3169
G16			3039.02	3.97	3047	3031	4	0,9606	3172
G22	2000	19990	13295.36	13.13	13318	13278	6	0,9430	14123
G23			13299.30	15.38	13322	13274	5	0,9429	14129
G24			13309.15	12.81	13319	13305	8	0,9425	14131
G32	2000	4000	1381.00	7.17	1392	1368	15	0,8923	1560
G33			1352.29	5.82	1362	1344	18	0,8861	1537
G34			1359.14	6.20	1368	1350	21	0,8877	1541
G35	2000	11778	7647.67	9.85	7665	7631	4	0,9581	8000
G36			7634.67	7.38	7643	7624	5	0,9559	7996
G37			7646.81	5.81	7657	7638	5	0,9560	8009
G43	1000	9990	6646.56	8.89	6655	6657	6	0,9471	7027
G44			6638.28	7.20	6649	6650	6	0,9469	7022
G45			6637.12	6.91	6634	6650	5	0,9450	7020
G48	3000	6000	6000	6000	6000	6000	41	1,0000	6000
G49			6000	6000	6000	6000	39	1,0000	6000
G50			5880	2862	5880	5880	32	0,9820	5988

Table 1. Relative results for Helmberg´s instances [11] in 50 independent iteratons

Name	*Circut*	*GRASP*	*GRASP + PR*	*GRASP + VNS*	*GRASP+ VNS+PR*	*VNS*	*VNS+ PR*	**MA+ VNS**	*SDP*
G1	**11624**	11540	11563	11589	11589	*11621*	11621	**11624**	12078
G2	11617	11567	11567	11598	11598	11615	11615	**11620**	12084
G3	**11622**	11551	11585	11596	11596	11622	11622	**11622**	12077
G11	560	552	564	560	564	560	560	562	627
G12	552	546	552	550	556	554	**556**	554	621
G13	574	572	**580**	576	578	580	**580**	**580**	645
G14	3058	3027	3041	3044	3044	3055	3055	**3061**	3187
G15	**3049**	3013	3034	3031	3031	3043	3043	3046	3169
G16	3045	3013	3028	3031	3031	3043	3043	**3047**	3172
G22	**13346**	13185	13203	13246	13246	13295	13295	13318	14123
G23	13317	13203	13222	13258	13260	13290	13290	**13322**	14129
G24	13314	13165	13242	13255	13255	13276	13276	**13319**	14131
G32	1390	1370	1392	1382	**1394**	1386	1396	1392	1560
G33	1360	1348	1362	1356	1368	1362	**1376**	1362	1537
G34	1368	1348	1364	1360	1368	1368	**1372**	1368	1541
G35	**7670**	7567	7588	7605	7605	7635	7635	7665	8000
G36	**7660**	7555	7581	7604	7604	7632	7632	7643	7996
G37	**7666**	7676	7602	7601	7608	7643	7643	7657	8009
G43	**6656**	6592	6621	6622	6622	6659	6659	6655	7027
G44	6643	6587	6618	6634	6634	6642	6642	**6649**	7022
G45	**6652**	6598	6620	6629	6629	6646	6646	6634	7020
G48	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	6000
G49	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	6000
G50	**5880**	5862	**5880**	**5880**	**5880**	5868	**5880**	**5880**	5988

Table 2. Comparison between our evolutionary hybrid metaheuristics and seven metaheuristic for the Helmberg's instances.

In order to get more comparable results, an average among all graphs is taken. These results are shown in Table 3. First column shows the metaheuristic name. In the second column appears the sum of the cutsets obtained for the 24 graphs. Finally, the third presents the relative cut value with respect to SDP upper bound. Again, the average value obtained by our proposal is good. Our proposal beats clearly to the rest of metaheuristicse except circut rank-2 relaxation. In this case, although this method obtains slightly better results, it needs a very long execution time. Notice that individuals, generations and $k_{max,}$ values are very low. Increasing or tuning adequately these values, probably our hybrid algorithm even could be reached at circuit.

Metaheuristic	**Sum**	**% of SDP**
SDP	157743	100
Circut	150623	95.49
GRASP	149337	94.67
GRASP + PR	149809	94.97
GRASP + VNS	149981	95.08
GRASP + NNS + PR	150060	95.13
VNS	150395	95.34
VNS + PR	150441	95.37
MA + VNS	150580	95.46

Table 3. Relative results for all metaheuristics

9. CONCLUSIONS

This paper has introduced a hybrid evolutionary algorithm to efficiently solve the Max-Cut problem. This hybridization is based on a low-level hybridization between a Variable Neighborhood Search and a Memetic Algorithm. This hybrid schema exploits the power of memetic algorithms to explore the solution space. On one hand, we have used a Variable Neighborhood Search as an additional intensification procedure to improve the corresponding optimization process. On the other hand, a memetic algorithm is mainly used to diversify the corresponding search process. Notice that this memetic algorithm includes several problem-dependent data and methods.

Taking into account the experimental results shown in thr above section, we can conclude that the hybrid schema proposed exploits the power of hybrid evolutionary algorithms to explore the solution space, enhanced with VNS as an additional intensification procedure. In other words, without adding much additional computational burden, Variable Neighborhood Search is able to improve the basic MA intensification strategy. On the other hand, the proposed memetic algorithm improves also the basic diversification strategy of VNS. The low-level hybridization proposed in this paper get a synergy between both metaheuristics

The experimental results also show that this algorithm has a robust behaviour and gives high quality solutions, independently of the graph characteristics. The hybrid memetic-VNS algorithm is quite efficient, taking into account that the other metaheuristics have been executed 1000 times and the proposed algorithm only once.

10. RERERENCES

[1] Barahona, F., Grötschel, M., Jürgen, M., and Reinelt, G., An application of combinatorial optimization to statistical optimization and circuit layout design. *Operations Research*, 36:493–513, 1988.

[2] Burer, S., Monteiro, R.D.C., Zhang X.: Rank-two Relaxation heuristic for the Max-Cut and other Binary Quadratic Programs. *SIAM Journal of Optimization*, 12:503-521, 2001.

[3] Chang, K.C, and. Du, D.-Z., Efficient Algorithms for Layer Assignament Problems. *IEEE Transaction on Computer-Aided Desgin*, CAD-6:67-78, 1987.

[4] Dolezal O., Hofmeister, T., Lefmann, H: A comparison of approximation algorithms for the MAXCUT-problem. *Reihe CI 57/99, SFB 531*, Universität Dortmund, 1999.

[5] Festa P., P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro: Randomized heuristics for the MAX-CUT problem, Optimization *Methods and Software*, vol. 7, pp. 1033-1058, 2002.

[6] Duarte, A., Fernández, F., Sánchez, A., Sanz, A.: A Hierarchical Social Metaheuristic for the Max-Cut Problem, *4th European Conference on Evolutionary Computation in Combinatorial Optimization* (EvoCOP 2004), LNCS v. 3004, Springer 2004

[7] Fiduccia, C. and R. Matheysses: A Linear-Time Heuristic for Improving Network Partitions. *In Proceedings of 19th Design Automation Conference*, pp. 175-181, 1982.

[8] Glover, F. and G. Kochenberger, editors: *Handbook of Metaheuristic*. Kluwer, Massachusetts, USA, 2003.

[9] Goemans, M. X., Williams, D.P.: Improved Approximation Algorithms for Max-Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM*.42:1115-1142, 1995.Goldberg. D.: Genetic Algorithms in Search, *Optimization and Machine Learning*. Addison-Wesley, 1989.

[10] Gosti, W. et al.: Approximation Algorithms for the Max-Cut Problem. *Technical Report*, Dept. Electrical Engineering and Computer Science, University of California, 1995.

[11] Helmberg, C., Rendl, F.: A Spectral Bundle Method for Semidefinite Programming. *SIAM Journal of Computing*, 10:673:696, 2000.

[12] Karp, R.M.: Reducibility among Combinatorial Problems. In R. Miller J. Thatcher, editors, *Complexity of Computers Computation*, Prenum Press, New York, USA (1972).

[13] Kernigham, B.W. and S. Lin: An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell System Technical Journal*, pp. 291-307, 1970

[14] Hadlock F. O: Finding a Maximum Cut of a Planar Graph in Polynomial Time. SIAM *Journal on Computing* 4 (1975) 221-225.

[15] Hansen, P. and Mladenovíc, N., Developments of variable neighborhood search. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 415–439. Kluwer Academic Publishers, 2001.

[16] Kim, S.-H., Y.-H. Kim and B.-R. Moon: A Hybrid Genetic Algorithm for the MAX CUT Problem. *In Genetic and Evolutionary Computation Conference*, pp. 416-426, 2001

[17] Mladenovíc, N. and Hansen, P.. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

[18] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd edn. Springer-Verlag, Berlin Heidelberg New York, 1996.

[19] Moscato P., Cotta C.: *A Gentle Introduction to Memetic Algorithms*. In Handbook of Metaheuristic. F. Glover and G. A. Kochenberger, editors, Kluwer, Norwell, Massachusetts, USA, 2003.

[20] Resende M.G.: GRASP With Path Re-linking and VNS for MAXCUT*, In Proc. of 4th MIC*, Porto, July 2001.

[21] Shani, S. and T. Gonzales: P-Complete Approximations Problems. *Journal of ACM*, 1976.

[22] Shor, N. Z.: Quadratic Optimization Problems, *Soviet Journal of Computing and System Science*, 25:1-11, 1987.

[23] Talbi, E.–G., A Taxonomy of Hybrid Metaheuristics, *Journal of Heuristics*, 8 (5): 541-564, 2002.

[24] . Wheeler, J.W, An investigation of the Max-Cut Problem, Internal Report, University of Iowa, 2004.