GRASP and VNS for Max-Cut

Paola Festa[*] *P.M. Pardalos*[†] *Mauricio G. C. Resende*[‡] *Celso C. Ribeiro*[§]

[*] Mathematics and Computer Science Department, University of Salerno
84081 Baronissi (SA), Italy
Email: paofes@unisa.it

[†] Industrial and Systems Engineering Department, University of Florida
Gainesville, FL 32611, USA
Email: pardalos@ufl.edu

[‡] Information Sciences Research Center, AT&T Labs Research
Florham Park, NJ 07932, USA
Email: mgcr@research.att.com

[§] Department of Computer Science, Catholic University of Rio de Janeiro
Rua Marquês de São Vicente 225, Rio de Janeiro, RJ 22453-900, Brazil
Email: celso@inf.puc-rio.br

1 Introduction and problem formulation

The Max-Cut problem can be stated as follows: Given an undirected graph $G = (V, E)$ and non-negative weights w_{ij} on the edges $(i, j) \in E$, $i, j \in V$, find a subset of vertices S that maximizes the sum of the edge weights in the *cut* $(S, \bar{S})$, i.e. the sum of the weights of the edges with one endpoint in S and the other in $\bar{S}$. The weight of a cut $(S, \bar{S})$ will be denoted in the following by

$$w(S, \bar{S}) = \sum_{(i,j) \in E \mid i \in S, j \in \bar{S}} w_{ij}.$$

The decision version of the Max-Cut problem was proved to be NP-complete by Karp [8], and remains NP-complete even in the unaligned case, i.e. if $w_{ij} = 1$ for all $(i, j) \in E$. It has found wide application, including VLSI design and statistical physics [1, 2].

In this abstract, we study GRASP and VNS heuristics for Max-Cut, together with a combination of the two in a hybrid procedure. GRASP [3, 4, 5] is characterized by a construction phase and a local search phase. The iterative construction builds a solution, one element at a time, randomly selected from a *restricted candidate list* (RCL), whose elements are well-ranked according to a greedy function. Once a feasible solution is obtained, a local search procedure tries to improve it producing a solution that is locally optimal with respect to the specified neighborhood structure. The construction and the local search phases are repeatedly applied, and the best solution found is kept as an approximation of the optimal. VNS [9] is based on the exploration of a dynamic neighborhood model.

The remainder of this abstract is organized as follows. Section 2 describes the two phases of the proposed GRASP, while Section 3 presents the VNS procedure. Section 4 reports on preliminary computational results. Some concluding remarks are drawn in the final section.

2 GRASP

Each GRASP iteration usually consists of the construction of a feasible trial solution followed by local optimization. In Figure 1 the proposed GRASP pseudo-code schema is shown.

```
procedure GRASP(maxitr,RandomSeed)
1    BestSolutionFound= ∅;
2    do k = 1, 2, ...,maxitr→
3        x ←ConstructGreedyRandomizedSolution();
4        x ← local_search(x);
5       UpdateSolution(BestSolutionFound,x);
6   od;
7   return(BestSolutionFound);
end GRASP;
```

Figure 1: GRASP pseudo-code.

The construction phase, shown in Figure 2, is iterative, greedy, and adaptive, in the sense that the element chosen at each iteration during the construction phase is dependent on those previously chosen. In order to describe the construction phase, an adaptive greedy function, a construction mechanism for the RCL, and a probabilistic selection criterion have to be provided. The greedy function takes into account the contribution to the objective function achieved by selecting a particular element. In the case of the Max-Cut problem, it is intuitive to relate the greedy function to the sum of the weights of its outgoing arcs. More formally, let $(S, \bar{S})$ be a cut, then for each vertex v we define the following quantities:

$$\sigma_S(v) = \sum_{u \in S} w_{vu} \quad \text{and} \quad \sigma_{\bar{S}}(v) = \sum_{u \in \bar{S}} w_{vu}.$$

The greedy choice is to select the vertex v with either highest $\sigma_S(v)$ or highest $\sigma_{\bar{S}}(v)$. If $\sigma_S(v) > \sigma_{\bar{S}}(v)$,

```
procedure ConstructGreedyRandomizedSolution()
1    S = S̄ = ∅;
2    do k = 1, 2, ..., n →
3        MakeRCL();
4        v ← SelectIndex(RandomSeed);
5        S ← S ∪ {v} (or S̄ ← S̄ ∪ {v});
6        Update(σ_S(v), σ_S̄(v));
7    od;
end ConstructGreedyRandomizedSolution;
```

Figure 2: GRASP construction phase pseudo-code.

then v is placed in $\bar{S}$, otherwise it is placed in S. To define the construction mechanism for the RCL, let

$$w_* = \min\{\min_{v \in V'} \sigma_S(v),\ \min_{v \in V'} \sigma_{\bar{S}}(v)\} \quad \text{and} \quad w^* = \max\{\max_{v \in V'} \sigma_S(v),\ \max_{v \in V'} \sigma_{\bar{S}}(v)\},$$

where $V' = V \setminus \{S \cup \bar{S}\}$. Denoting by $\mu = w_* + \alpha\,(w^* - w_*)$ the cut-off value, the restricted candidate list is made up of all vertices that have greedy function values greater than or equal to μ.

Let u, v be two vertices such that $u \in \bar{S}$ and $v \in S$. Two types of neighbors are defined as follows:

1. $S \leftarrow S \setminus \{v\}$, $\bar{S} \leftarrow \bar{S} \cup \{v\}$;
2. $\bar{S} \leftarrow \bar{S} \setminus \{u\}$, $S \leftarrow S \cup \{u\}$.

The GRASP local search procedure works as follows. Starting from the first elements of the sets S and $\bar{S}$, it checks in turn whether moving the elements from one set to the other leads to an improvement of the objective function. If an improving move is found, it is made. The local search ends when no further such improvement is possible.

3 VNS

Let the binary $|V|$-vector x represent a Max-Cut solution. If node $i \in S$, then $x_i = 0$, otherwise $x_i = 1$. The k-th order neighborhood $N^{(k)}(x)$ of a solution x is formed by all solutions x' whose Hamming distance from x is exactly k, i.e. $\sum_{i=1}^{i=|V|} |x_i - x'_i| = k$. Let $k_{\max}$ be the order of the highest neighborhood explored.

```
procedure VNS(x)
1     k ← 1;
2     do k ≤ k_max  →
3           Randomly generate x' ∈ N^(k)(x);
4           x̄ ← local_search(x');
5           k ← k + 1;
6           if f(x̄) > f(x)  →
7                 x ← x̄;
8                 k ← 1;
9           fi;
10    od;
end VNS;
```

Figure 3: VNS pseudo-code.

Figure 3 describes the VNS. The neighborhood size parameter k is initialized to 1. The search explores a sequence of increasing order neighborhoods. Starting from the initial solution x, the loop is repeated until the neighborhood size parameter exceeds the maximum size k_{max}, in which case neighborhoods $N^{(1)}(x), \ldots, N^{(k_{\max})}(x)$ will have been explored without finding any improving solution. To do this, a solution x' belonging to the k-th order neighborhood of x is randomly generated. The local search procedure used in the pure GRASP, and described in the previous section, is applied to x'. If the locally optimal solution $\bar{x}$ improves upon the current solution x, the latter is updated and the algorithm returns to explore again the first order neighborhood. Otherwise, the search continues using the next order neighborhood.

4 Experimental results

In this section, we report some results obtained with preliminary experiments using three heuristics:

- `grasp`: A pure GRASP in which `maxitr` independent iterations are executed, each of them using the RCL parameter α selected from a uniform distribution interval $[0, 1]$;
- `g-vns`: `grasp` using VNS as the local search; and
- `vns`: `g-vns` with randomly constructed initial solutions.

Some preliminary experiments were performed on an SGI Challenge computer (28 196-MHz MIPS R10000 processors) with 7.6 Gb of memory. The codes were written in Fortran 77 and compiled with

the SGI MIPSpro F77 compiler using flags `-O3 -r4 -64`. CPU times were measured with the system function `etime`.

We report in Table 1 the results obtained with the three algorithms on four Max-Cut instances. Problem G11, with 800 nodes and 1600 edges, was created by Helmberg and Rendl [7]. Problem FFKN500 is a sparse, randomly generated instance proposed by Fujisawa et al. [6] to test their semidefinite programming code SDPA. Instance pm3-8 comes from the benchmark problem set of the 7th DIMACS Implementation Challenge, and was generated by M. Jünger and F. Liers using the Ising model of spin glasses. Finally, problem B4 is a large toroidal graph having 8000 nodes and 16000 edges.

Table 1: Comparison of three heuristics on ten instances.

Instance	$\|V\|$	$\|E\|$	UB	Best	`grasp`	`g-vns`	`vns`
G11	800	1600	629	554	550	560	560
FFKN500	500	625	598	574	568	574	574
pm3-8	512	1536	461	452	442	456	458
B4	8000	16000	16000	16000	16000	16000	16000

For each instance, we ran each heuristic a total of 100 iterations. Parameter $k_{\max} = 100$ was used on both `vns` and `g-vns`. Note that the running times for the 100 iterations vary considerably for the three heuristics. The pure GRASP is the fastest, while the hybrid is the slowest. For each instance the table lists its name, its dimensions, its semidefinite programming upper bound, and its best known lower bound, followed by the solution values obtained by each heuristic.

For the same number of iterations, the pure VNS and the hybrid heuristic found, in general, solutions that are better than those found by the pure GRASP. However, this is at the cost of longer computation times. Note that both the pure VNS and the hybrid heuristic improved the current best known solution for instance pm3-8,

Since the running times per iteration of the three heuristics vary, we plot in Figure 4 the empirical distribution of the random variable *time-to-target-solution-value* considering instance FFKN500, using a target value of 570. We performed 200 independent runs of the pure VNS and the hybrid, as well as 77 independent runs of the pure GRASP. On these runs, parameter $k_{\max} = 100$ was used on the pure VNS and $k_{\max} = 15$ in the hybrid heuristic, to allow for a greater number of iterations.

This figure clearly shows that the pure VNS was the fastest to find cuts of weight 570 or larger. It was followed by the hybrid heuristic, while the pure GRASP was the slowest. The pure VNS found the target value on all 200 trials in at most 125 seconds, while the hybrid heuristic required as much as 935 seconds and the pure GRASP as much 10641 seconds. Fifty percent of the pure VNS runs found the target value in less than 17 seconds, while half of the hybrid heuristic and pure GRASP runs concluded in less than 99 and 1255 seconds, respectively. In the time it took the pure VNS to conclude its longest run, only 57% of the hybrid heuristic runs were completed and 7% of the pure GRASP runs. In the time it took the hybrid heuristic to conclude its longest run, 43% of the pure GRASP runs were completed.

5 Concluding remarks

Though it is hard to make definitive conclusions from the preliminary computational results, they indicate VNS as a strong or robust heuristic for Max-Cut. One interesting observation is that VNS seems to be not very sensitive to the initial solution, performing well from randomly generated starting solutions.

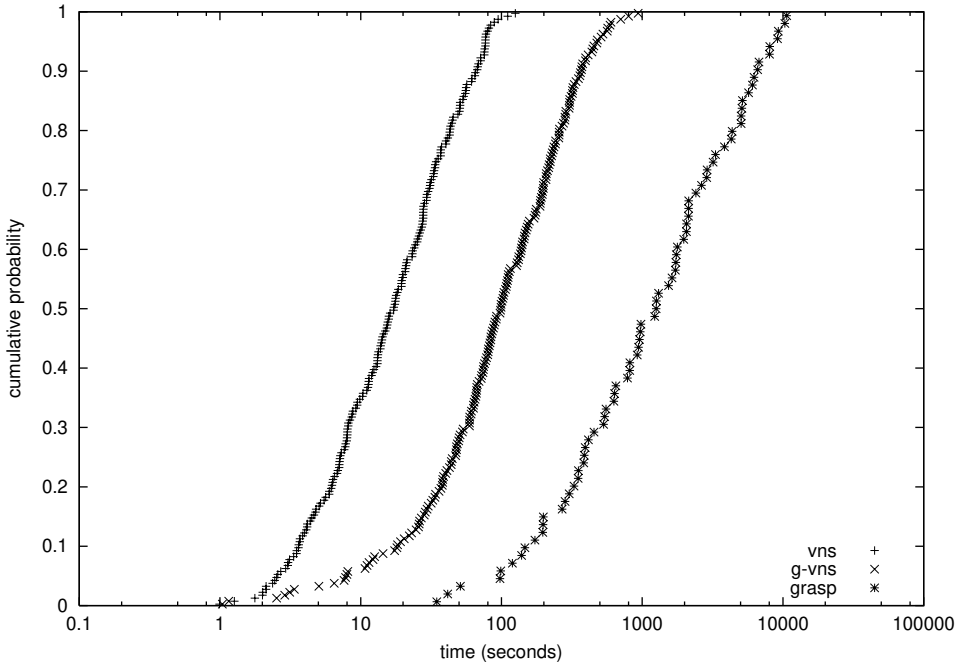

Figure 4: Empirical distribution of time to target solution value.

In this preliminary study, we have limited ourselves to only two values for $k_{\max}$. We will explore other values of this parameter both in VNS and in the hybrid heuristic and observe how it affects their behaviors. We are also currently exploring the use of memory mechanisms, such as path relinking, in the above algorithms.

References

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

[2] K.C. Chang and D.-Z. Du. Efficient algorithms for layer assignment problems. *IEEE Trans. on Computer-Aided Design*, CAD-6:67–78, 1987.

[3] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

[4] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *J. of Global Optimization*, 6:109–133, 1995.

[5] P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In P. Hansen and C.C. Ribeiro, editors, *Essays and Surveys on Metaheuristics*. Kluwer Academic Publishers, 2001.

[6] K. Fujisawa, M. Fukuda, M. Kojima, and K. Nakata. Numerical evaluation of SDPA (Semidefinite Programming Algorithm. Technical report, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguru-ku, Tokyo 152, Japan, 1997.

[7] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. Technical report, Konrad-Zuse-Zentrum fuer Informationstechnik Berlin, Takustrasse 7, D-14195 Berlin, Germany, 1997.

[8] R.M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, NY, 1972.

[9] N. Mladenović and P. Hansen. Variable neighbourhood search. *Computers and Operations Research*, 24:1097–1100, 1997.