# Competitive Simulated Annealing and Tabu Search Algorithms for the Max-Cut Problem

Emely Arráiz
Departamento de Computación y Tecnología de la Información
Universidad Simón Bolívar
Caracas, Venezuela
arraiz@ldc.usb.ve

Oswaldo Olivo
Departamento de Computación y Tecnología de la Información
Universidad Simón Bolívar
Caracas, Venezuela
olivo@ldc.usb.ve

## ABSTRACT

The Max-Cut problem consists of partitioning the nodes of an undirected weighted graph into two subsets, such that the sum of the weights of the edges that connect two vertices in different partitions is maximized. It has applications in several fields like statistical physics, VLSI design, among others, and is known to be NP-Hard. We propose a Neighborhood generation method that balances diversity and quality of the obtained solutions. Consequently, the inclusion within Simulated Annealing and Tabu Search frameworks produces similar results to those reported by state-of-the-art methods such as VNSPR and Scatter Search, and in some cases improves them.

## Categories and 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 strategies*

## General Terms

Algorithms, Experimentation

## Keywords

Max-Cut, Metaheuristic, Simulated Annealing, Tabu Search, Algorithms

## 1. INTRODUCTION

Let $G = (V, E)$ be an undirected graph where $V = \{1, .., n\}$ is the set of vertices and $E = \{(i, j) : i, j \in V\}$ is the set of edges. For an edge $(i, j) \in E$, we denote $w_{ij}$ the associated cost. Additionally, let $x_i$ equal 0 or 1 whether the $i^{th}$ node is in the first or second partition, respectively. The Max-Cut problem consists in maximizing the following expression:

$$cut(S, \overline{S}) = 1/2 \sum_{x_i \neq x_j, (i,j) \in E}^{n} w_{ij}$$

The problem has several applications in VLSI circuit design and statistical physics , as well as combinatorial optimization. The Max-Cut stated as a search problem is NP-Hard in the general case. Consult [2] for a more extended insight of the problem. The most recent significant work on Max-Cut (to our knowledge) was the Advanced Scatter Search proposed by Martí et. al.[7]. Our intention in this work was to compare our Tabu Search (TS) and Simulated Annealing (SA), coupled with a complex Neighbor generation method, with Martí's algorithm. Due to space limitations, pseudocodes, tables of results among other relevant issues should consulted in [1].


*GECCO'09,* July 8–12, 2009, Montréal, Québec, Canada.
ACM 978-1-60558-325-9/09/07.

## 2. PROPOSED HEURISTICS

In this section we present the Simulated Annealing and Tabu Search algorithms that we propose to employ in the Max-Cut problem.

### 2.1 Constructive Heuristic

A constructive heuristic is used to generate a starting solution for the main algorithm. We implemented a slight variation of Kahruman's[6] SG3 procedure, which we call *greedySG3*. Consider $\sigma_S(v) = \sum_{u \in S} w_{uv}$ and $\sigma_{\bar{S}}(v) = \sum_{u \in \bar{S}} w_{uv}$ for $v \in V$ . We describe greedySG3 as follows: Two random vertices are chosen to be on different partitions. The current cut value is the weight of the the arc that connects these two nodes (zero if not linked). Then, at each iteration of the greedy loop, a node z with the greatest difference from putting it on one partition or the other is selected ($z \longleftarrow \max\{v \in T :| \sigma_S(v) - \sigma_{\bar{S}}(v) |\}$). It is put on the partition which maximizes the cut. This procedure is repeated until all the vertices have been assigned to a partition.

### 2.2 Improvement Heuristic

After having constructed the initial solution, an improvement procedure is generally invoked in order to produce an even better starting point for the main heuristic. We choose Ejection Chain [4], as Martí et. al. did in their Scatter Search framework. The Ejection Chain procedure consists in making multiple moves as a consequence of ejecting nodes in a recently exchanged vertex's partition. The first procedure consists of sorting the vertices in increasing order, according the difference obtained when a change of partition is performed. The iterative procedure consists in going over this ordered set and changing the vertices with the greatest increment first. If an exchange increases the sum value, the process returns the resulting cut. Otherwise, the vertex is moved to the other partition and the recursive procedure continues by attempting to change a neighbor on the same partition.

## 2.3 Neighborhood Generation Method

For the Simulated Annealing and Tabu Search methodologies a neighborhood generation method is needed. The algorithm consists in moving a fixed vertex from its partition accompanied by another random exchange. We call the former a *linearly selected vertex* and the latter a *randomly selected vertex*. Changing the linearly selected vertex, the randomly selected vertex and both are denominated *linear move* , *random move* and *compound move* respectively. The method terminates whenever it finds an improving neighbor or generates all the neighbors dictated by the parameter.

## 2.4 Simulated Annealing

One of the two main frameworks that will be used in our tests is described in this section. The basis of this method comes from metallurgy, which is an activity where a heating process is carried out and then a slow cooling stage to improve the quality of the material.

Our heuristic ressembles common SA implementations. The interesting part is certainly the neighborhood generation, which was described previously.The procedure executes until a number of consecutive iterations (maxIterWithoutImp) without improvement has been reached. Acceptance of non-improving solutions is given by the comparison between delta (a random number in [0..1]) and the exponential of the increment in the cut divided by the temperature.

## 2.5 Tabu Search Heuristic

The Tabu Search generates neighbors and stores employed moves in order to prevent visiting previous states. Again, our approach is quite common, with the distinction of the Neighborhood search presented previously and some other features. The number of iterations that a move is tabu is not fixed. A random number is generated in the interval [minTabuIter,maxTabuIter][3]. A neighbor is created by linearly changing one node and selecting randomly another node for exchange of partition. When no improving move is found, the least worse movement is performed, instead of stopping. We employ the best first approach in the neighbor selection procedure.

# 3. EXPERIMENTAL RESULTS

For the tests we employed the widely used rudy graphs. Helmberg and Rendel [5] describe how to use rudy to generate the graph instances. The machine used in the experiment has an AMD CPU 1.66 GHz processor and 512 MB RAM. All tests were carried out on the same processor, on the Ubuntu 8.10 operation system. The algorithms were coded in g++ (GNU C++, version 4.1.2.) and compiled with the flag -O3. The experiment consisted in running all the graphs proposed by Helmberg (g01.rudy through g52.rudy) on our two proposed heuristics (SA and TS). We ran 10 times each graph on every heuristic, in order to report not only the best value obtained, but also other important metrics (i.e. average value, time).

Note that our algorithms were run on a 1.66 GHZ CPU with 512 MB RAM , while the others were obtained on a machine with a 3.2 GHz processor and 2.0 GB of RAM. The results reported on table 1 were taken from [7].

TS reports 3 of the best results known in the literature. On the other instances, it performs relatively well, but not enough to overcome other methods. However, with an average time of 30 seconds and less than 5 % of deviation from the SDP it is certainly useful when the goal is to obtain high quality solutions with minimal computational burden. The Simulated Annealing heuristic performs better, achieving 12 best results over the 24 instances. SS also reports 12 of the best known results , while CirCut achieves 13 and VNSPR obtains 9 . Additionally, SA comes very close to the results obtained by SS, Circut with a 95.44 against 95.48 performance with respect to the SDP bound, and overcomes VNSPR (95.37). This heuristic also requires considerable less computational times, with a 130.0 average per instance, compared to the 549.0, 251.3 and 77603.1 seconds taken on average by SS, Circut and VNSPR respectively. Therefore, SA reports very high quality results taking at most half of the time on average compared to widely known heuristics.

# 4. CONCLUSIONS AND FUTURE WORK

We have presented Simulated Annealing and Tabu Search procedures for the Max-Cut problem. Crucial was the definition of the neighbor selection method, which consisted of looking for the first linearly selected and random vertex exchange compound move that resulted in an improvement of the cut (or the least worse when none achieves a better cut). We consider that the TS is very useful when minimal computational burden is desired. SA is adequate when top quality is required within medium CPU usage. If larger computational times are acceptable, Scatter Search or CirCut should be employed.

Embedding TS in more complex frameworks such as SS or GAs, and enhancing SA with PR should be attempted the in future work.

# 5. REFERENCES


[1] Arráiz, E., Olivo, O. Competitive Simulated Annealing and Tabu Search Algorithms for the Max-Cut Problem. `www.ldc.usb.ve/~olivo/publications/CompetitiveSAandTSforMaxCut.pdf`

[2] Deza, M. and Laurent, M. Applications of Cut Polyhedra I. *Journal of Computational Applied Math*, 55(2):191–216, November 1994.

[3] Gaspero, L.D. and Schaerf, A. Tabu Search Techniques for Examination Timetabling. *Lecture Notes in Computer Science*, 2079:104–117, 2001.

[4] Glover, F. Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems. *Discrete Applied Mathematics*, 65(1-3):223–253, 1996.

[5] Helmberg, C. and Rendl, F. A Spectral Bundle Method for Semidefinite Programming. *siopt*, 10(3):673–696, 2000.

[6] Kahruman, S., Kolotoglu, E., Butenko, S. and Hicks, I.V. On Greedy Construction Heuristics for the MAX-CUT Problem. *International Journal of Computational Science and Engineering*, 3(3):211–218, 2007.

[7] Martí, R., Duarte, A. and Laguna, M. Advanced Scatter Search for the Max-Cut Problem. *Informs Journal on Computing*, page ijoc.1080.0275, 2008.