\\ 132 \\

Almost-optimal solution of large weighted equicut problems

by

Mauro Dell'Amico
Marco Trubian

April 1996

Università di Modena
Dipartimento di Economia Politica
Viale Berengario, 51
41100 Modena (Italia)
e-mail: dellamic@elet.polimi.it

Almost-optimal solution of large weighted equicut problems

Mauro Dell'Amico[a], Marco Trubian[b]

[a]Dipartimento di Economia Politica, Università di Modena,
via Berengario, 51, 41100 Modena, Italy; Email: dellamic@elet.polimi.it

[b]DEIS, Università di Bologna, viale Risorgimento, 2, 40131, Bologna, Italy
Email: trubian@elet.polimi.it

Abstract

Given a weighted undirected graph, the equicut problem consists of finding a partition of the vertex set into two subsets of equal cardinality such that the sum of the weights of the edges belonging to the cut defined by the vertex partition is minimized. The problem is NP-hard and has several practical applications. In the last years a number of algorithms based on metaheuristic techniques have been proposed. In this work we first present a survey of the algorithms from the literature, then we propose a new tabu search algorithm and experimentally compare it with the other heuristics on several classes of graphs, with up to 4000 nodes and 320000 edges. The computational results show that our approach easily determines the optimal solution for small graphs and its average performances are largely superior to those of the other approximating algorithms.

Keywords: Heuristics; Graph partitioning; Tabu search; Metaheuristic

1 Introduction

In the *equicut* problem we are given a graph $G = (V, E)$ with vertex set $V = \{1, \ldots, n\}$, edge set E and with weights associated to the edges. A cut, i.e. a proper partition $(S, V \backslash S)$ of the vertex set (with $S \subset V, S \neq \emptyset$), is an equicut if the number of vertices in the two *shores* S and $V \backslash S$ differs of at most one. We want to find the equicut which minimizes the sum of the weights of the edges involved in the cut. The problem is also known as the *2-way uniform partitioning* problem (see [13]).

More formally let $e = [i, j] \in E$, with $i \in V, j \in V$ and $i \neq j$, be any edge of the graph and w_e (or $w(i, j)$) be the weight of the edge. For any proper partition $s = (S, V \backslash S)$ let $\delta(S) = \{[i, j] \in E : i \in S, j \in V \backslash S\}$ be the set of edges in the cut defined by partition s. We want to find the partition s such that $|S| = \lfloor \frac{n}{2} \rfloor$ and such that $z(s) = \sum_{e \in \delta(S)} w_e$ is a minimum.

The equicut problem is known to be $\mathcal{NP}$-hard (see e.g. Garey and Johnson, [11]), even if $w_e \in \{0, 1\}, \forall e \in E$. In this case it is called the *0–1 equicut problem.*

The equicut has several applications. In VLSI design it models the problem of optimally placing "standard cells" in order to minimize the routing area required to connect

the cells (see Dunlop and Kernighan [8]). It also models the problem of minimizing the number of holes on a circuit board, subject to pin preassignment and layer preferences and has applications in Physics, where it models the problem of finding the ground state magnetization of spin glasses having zero magnetic field (see Barahona et al.[1]). Other applications arise in layout and floor planning (see Dai and Kuh [5]), in computer memory paging and in Group Technology (see Feo and Khellaf [9]).

With no loss of generality we can always assume that the number of vertices in the given graph is even. Moreover, to simplify the presentation, we assume that a generic equipartition is given by (A, B) with $A \subset V, B \subset V, |A| = |B|, A \cup B = V, A \cap B = \emptyset$.

Finally we assume that the reader is familiar with the basic concepts of the following metaheuristic techniques: *local search, multistart, genetic algorithms, simulated annealing* and *tabu search.*

In the following Section we present an updated survey of the literature in which we describe the existing heuristic and exact algorithms, we discuss the relevant ideas of each approach and we compare their relative effectiveness. In Section 3 we introduce our new tabu search algorithm, which is experimentally compared with the other approaches, in the last Section 4.

2 A survey of the approaches from the literature

The equicut problem has been investigated for more than twenty five years. With the exception of a classical deterministic heuristic (the first approach presented) and an exact algorithm (one of the last approaches), all the other algorithms from the literature use metaheuristic techniques. This is probably due to the simple structure of the problem which allows to represent a solution with compact data structure, to have simple and natural neighborhoods of a given solution and to have immediate feasibility tests.

The simplest data structure to store a solution $s = (A, B)$ is a boolean array with n elements: vertex $i \in V$ is assigned to shore A or B if the i-th element of the array is given value 0 or 1. This representation immediately suggests a *genetic* approach where each boolean array, defining a solution or *chromosome*, is partitioned into two or more substrings so giving a number of *alleles* which are combined to obtain a new *offspring.* The feasibility of the new solutions can be simply checked by counting the number of $'0'$ and $'1'$ in the new arrays.

Given a feasible solution s a set of *neighboring* solutions $N(s)$ can be immediately obtained by selecting $2k$ $(1 \leq k < n/2)$ vertices, k from shore A and k from shore B, and moving each vertex to the opposite shore. Alternatively one can move h $(1 \leq h < n)$ vertices from one shore to the other, allowing infeasible solutions that can be transformed into feasible ones again, in successive iterations.

In the following we describe the results from the literature, then we compare the performance of the various algorithms and finally we make some considerations on the implementations of basic operations common to many approaches.

2.1 Algorithms from the literature

A first important difference between the existing approaches for the solution of the equicut problem is the following: (a) algorithms for the equicut with generic weights on the edges; (b) algorithms for the 0–1 equicut. Although the algorithms in (a) can certainly solve the 0–1 equicut, the simple structure of the objective function with only 0–1 costs allows to use sophisticated data structure and to obtain fast specialized heuristics. Thus algorithms written for the weighted case are not effective for the 0–1 case. On the contrary in order to modify an algorithm for the unweighted case to solve the general problem, one has to reconsider the implementation, the data structure to be used and the general strategy (see Section 2.3).

The first (classical) heuristic for the general equicut problem is due to Kernighan and Lin [13] (KL in the following). This algorithm starts from a given feasible partition (A, B) of the vertices (randomly generated) and iteratively improves it. At each iteration, for all vertices $a \in A$ let $E_a = \sum_{j \in B} w(a, j)$ be the *external cost* and $I_a = \sum_{j \in A} w(a, j)$ be the *internal cost* (similarly, for a vertex $b \in B$, define $E_b = \sum_{j \in A} w(b, j)$ and $I_b = \sum_{j \in B} w(b, j)$). For any vertex j the *difference* D_j $(= E_j - I_j)$ of the external and internal costs is the variation of the cut weight, or gain, if j is moved from one shore to the other. If a vertex $a \in A$ and a vertex $b \in B$ are *exchanged* (i.e. a is moved to set B and b to set A), the value of the new solution is $D_a + D_b - 2w(a, b)$ units less than the value of the original solution. Each iteration of the algorithm determines a subset of $k < n/2$ vertices from set A to be exchanged with k vertices from B as follows. First determine $a(1) \in A$ and $b(1) \in B$ such that the overall gain $g(1) = D_{a(1)} + D_{b(1)} - 2w(a(1), b(1))$ is a maximum among all possible pairs a, b. Then vertices $a(1)$ and $b(1)$ are temporarily exchanged, the values D_i $(i \in V)$ are updated and $a(1)$, $b(1)$ are removed from sets A and B, respectively. A second pair of vertices $a(2) \in A$, $b(2) \in B$ for which $g(2) = D_{a(2)} + D_{b(2)} - 2w(a(2), b(2))$ is a maximum, is selected. This procedure is repeated $n/2 - 1$ times to compute $g(1), \ldots, g(n/2 - 1)$ and finally the value k which maximize $G = \sum_{i=1}^{k} g(i)$ is determined. If $G > 0$, then a new (improved) solution is obtained by moving vertices $a(1), \ldots, a(k)$ from the original set A to set B and moving vertices $b(1), \ldots, b(k)$ from the original set B to A, and a new iteration is started. If otherwise $G \leq 0$ the procedure terminates.

Fiduccia and Mattheyses [10] proposed a variant of KL in which the single vertex j with maximum gain (D_j) is moved to the opposite shore. In order to have balanced partitions vertex j is chosen alternatively form A and B. This speeds up the algorithm, but produces a slight decrease in the solution quality.

The first tentative to solve the equicut problem with a metaheuristic technique is due to Kirkpatrik, Gelatt and Vecchi [15] and to Kirkpatrik [14], who used a *simulated annealing* (SA) approach. The computational results presented were limited and a not efficient implementation of the Kernighan-Lin algorithm was used as a competitor.

Johnson et al. [12] presented a deep and extensive study of the simulated annealing approach to the 0–1 equicut problem. In order to optimize the parameter settings and to determine the better components of the algorithm the authors perform a wide series of

computational experiments on two classes of unweighted graphs with up to 1000 vertices. Worth is noting that the resulting algorithm is very effective for the 0–1 case, but not necessarily good for the weighted case. Moreover the sophisticated data structure used cannot be utilized for the weighted case (see Section 2.3, below).

The algorithm is as follows. Given a partition $s = (S, V\backslash S)$, not necessary feasible, the neighborhood $N(s)$ of s consists of all the partitions obtained by moving one vertex from one shore to the other (i.e. $N(s) = \{(S\backslash\{v\}, (V\backslash S) \cup \{v\}) : v \in S\} \cup \{(S \cup \{v\}, V\backslash(S \cup \{v\})) : v \in V\backslash S\}$. The value of a partition is computed by adding to the weight of the cut a quadratic penalty function of the lack of balance. More precisely, the cost of the partition s is $c(s) = |\delta(S)| + \alpha(|S| - |V\backslash S|)^2$, where α is an appropriate parameter called the IMBALANCE FACTOR. (Note that this cost function is written for the unweighted case.) The pseudocode of the general SA algorithm, which utilizes four further parameters, follows.

Procedure SA
randomly select a starting solution s;
select an initial temperature $T > 0$ using parameter $INITPROB$;
while the number of accepted solutions is greater than MINPERCENT **do**
  for $n \times SIZEFACTOR$ times **do**
    select a neighbor $s' \in N(s)$;
    if $c(s') - c(s) \leq 0$ **then** set $s := s'$;
    else set $s := s'$ with probability $e^{(c(s)-c(s'))/T}$
  end for;
  $T := TEMPFACTOR \times T$
end while;
return s;

Basing on their experiments the authors suggest values for the five parameters. Moreover they observe that changes in the cooling strategy do not produce significant improvements in the performances. The same happen if a *cutoff* is included to speed up the computation (i.e the **for** loop is terminated either after $n \times SIZEFACTOR$ iterations or when $CUTOFF \times n \times SIZEFACTOR$ solutions have been accepted (with $0 < CUTOFF < 1$)). Another characteristic observed is that if the initial solution is not selected randomly, but with the KL heuristic, the performances do not change, however starting solutions of different nature (e.g. manual solutions of real life problems) can lead to better final values.

Two genetic algorithms have been developed by Rolland and Pirkul [20],[18]. The first algorithm uses a one-point crossover (i.e. each chromosome is divided into two alleles), a bit-by-bit probabilistic *mutation* technique and a *fitness* function which includes a linear penalty function of the imbalance of the two shores. The second algorithm is an improvement of the previous one obtained by removing the imbalance penalty and transforming each chromosome in a feasible solution by means of a greedy balancing procedure.

Laguna, Feo and Elrod [16] presented a *Greedy Randomized Adaptive Search Procedure* ($GRASP$) for the 0–1 equicut problem. The main body of $GRASP$ consists of two phases: (i) identify an initial solution through a greedy randomized algorithm; (ii) improve this solution using a local search procedure. Since phase (i) is randomized the two phases can be repeated to look for new better solutions. Phase (i) starts with an empty partition $A = B = \emptyset$, than selects one vertex at a time to be added to set A or B, alternatively. For each unselected vertex j the difference D_j is computed with respect to the current sets A and B, and a vertex is randomly selected among the first k elements with smallest value D_j (were k is a fixed parameter). For the improving phase the authors considered local search procedure based on the exchange of two vertices (one from set A and one from set B). Among different strategies they have chosen the *slightest swap* technique which search, among all possible pairs, the first exchange with minimum positive gain (i.e. a pair a, b, with $a \in A, b \in B$ such that $g = D_a + D_b - 2w(a,b) = 1$.)

Rolland, Pirkul and Glover [21] presented a tabu search approach TS which uses a restricted version of neighborhood $N(s)$ of [12]. More specifically, given a partition $s = (S, V \backslash S)$ (with $1 \leq |S| < n/2$) the restricted neighborhood $RN(s) \subseteq N(s)$ is equal to $N(s)$ if the absolute value of the imbalance $|S| - |V \setminus S|$ is less or equal to a specified value $ImBalanceFactor$; otherwise $RN(s)$ contains only the partitions obtained by moving one vertex from the largest shore to the smallest one. A solution obtained by moving a vertex j from S to $V \backslash S$ is considered tabu if vertex j was moved from $V \backslash S$ do S in the last *tabu tenure* iterations. At each iteration the algorithm selects the solution $s' \in RN(s)$ which is not tabu and which maximize the improvement of the objective function value. A move *aspiration criteria* is applied, which remove the tabu status if the move produces a solution which improves the better solution found. The tabu tenure is fixed to $3 \cdot \sqrt{n/2}$. The value $ImBalanceFactor$, initially zero, is dynamically updated so as to obtain a *strategic oscillation* (see Glover and Laguna [17]). The resulting algorithm is as follows.

Procedure TS
randomly select a starting solution s;
$Best\_Sol = \emptyset$; $NoImprove := ImBalanceFactor := 0$;
for *iterations* $:= 1$ **to** $\max(100, 5n)$ **do**
  let C be the set of solutions $s' \in RN(s)$ obtained from s with a not tabu move or with a tabu, move, under the condition that $z(s') < z(Best\_Sol)$ (aspiration criteria);
  choose $s' \in C$ which has minimum value;
  define tabu the move which leads from s' to s;
  if $(z(s') < z(Best\_Sol)$ **and** s' is feasible) **then** $Best\_Sol := s'$;
  else $NoImprove := NoImprove + 1$;
  if $NoImprove > 20$ **then**
    $ImBalanceFactor := ImbalanceFactor + 1$; $NoImprove := 0$;
  if $ImBalanceFactor > 4$ **then** $ImbalanceFactor := 0$;
end for

A lagrangian based heuristic for the general equicut problem has been developed

by Pirkul and Rolland [19]. They start with a mathematical model, for the problem, which uses two kinds of variables: x_i to assign a vertex $i \in V$ to one of the two shores, and y_{ij} to recognize if an edge $[i,j] \in E$ belongs to the cut determined by the current partition. Embedding in a lagrangian fashion the constraints which relate the variables associated to the vertices with the variables associated to the edges, two simple problems with either x or y variables, are obtained. In particular the problem with only the x variables defines an *equipartition* (disregarding the weights of the edges). Optimizing the lagrangian parameters through a subgradient ascent technique a lower bound for the equicut is obtained. During the ascent phase several feasible solutions are generated: at each one a local search procedure is applied, which, in practice, is a single iteration of KL. This heuristic can be seen as an "intelligent" *multistart* algorithm which generates the next starting solution taking into account the status of the current relaxed problem.

Recently Dell'Amico and Maffioli [6] have presented an effective tabu search algorithm, called $EnTaS$, for the 0–1 equicut problem. They consider only feasible partitions s and a neighborhood $PN(s)$ containing all the partitions obtained from s by interchanging a pair of vertices, one from set A and one from set B. Several diversification techniques have been applied: dynamic updating of the tabu tenure, memorization of some promising, but not explored solutions, and restarting. The resulting algorithm has been shown to be one of the most effective for solving unweighted equicut instances (see Section 2.2 for further details).

For the exact solution of weighted equicut problems a branch-and-cut algorithm has been written by Brunetta, Conforti and Rinaldi [2], basically using the theory developed in Conforti, Rao and Sassano [3], [4].

2.2 Comparison of the algorithms

The KL procedure is well known to be the standard benchmark algorithm for equicut problems. In particular a simple multistart algorithm $M\text{-}KL$ can be obtained by generating a random solution s, applying KL to s and repeating these two steps until a stopping criterion results true. The algorithm returns the best solution found.

The variant of KL presented in [10] has not to be considered as a valid competitor since the same authors recognize that the average performances are worst than that of KL.

For the above reasons we have chosen to use $M\text{-}KL$ as one of the competitors for our tabu search algorithm. To identify other possible competitors we first consider the approaches for the 0–1 equicut, and than those for the general equicut.

The 0–1 equicut problem

The SA algorithms presented in [15] and [14] are very preliminary studies of this technique and the computational experiments are limited. On the contrary the SA algorithm of [12] is an accurate implementation based on an extensive computational study. Unweighted graphs of two classes have been used to test the algorithm: (a)

random graph $G_{n,p}$, were n is the number of vertices and $0 < p < 1$ is the probability that the edge between a given pair of vertices exists; (b) *geometric graph* $U_{n,d}$, generated drawing from an uniform distribution n points in an unit square, associating a vertex to each point and adding edge $[u, v]$ to the graph iff the euclidean distance between the u and v is less or equal to d. The expected average vertex degree $\hat{\nu}$ is equal to $p(n-1)$ for the random graphs, whereas it is approximately $n\pi d^2$ for the geometric graphs. The instances considered have $n = 124, 250, 500, 1000$ and $\hat{\nu} = 2.5, 5, 10, 20$. The authors compare SA with $M\text{-}KL$ showing that simulated annealing beats the traditional heuristic for large random graphs ($n = 250$ and $\hat{\nu} \geq 20$ or $n \geq 500$) even when running time is taken into account. It is substantially outperformed for geometric graphs.

The $GRASP$ approach of [16] has been tested on random and geometric graphs and compared with a single run of the KL heuristic. The instances considered have $n = 124, 250, 500, 1000$ and $\hat{\nu} = 2.5, 5, 10, 20, 40, 80$. After an extensive computational study the authors conclude that: "$GRASP$ compares well with the average KL solution when both methods employ the same computational effort. The average solution quality considerably improves with a modest increase in computational effort". Therefore $GRASP$ can be considered the most effective technique with respect to KL and SA, for the 0–1 equicut.

The tabu search $EnTaS$ introduced in [6] has been tested on the same kind of instances used in [16] and compared with the $GRASP$ algorithm. Wide computational experiments led to conclude that "$EnTaS$ beats $GRASP$ on all problems, except on the very sparse geometric graphs" ($n \geq 500$ and $\hat{\nu} \leq 10$).

From the above discussion we have that $GRASP$ remains a possible competitor for our new tabu search algorithm.

The weighted equicut problem

The genetic algorithms (GA) of [20] and [18] have been tested on 144 instances of the general equicut problem, from four classes: (i) dense and (ii) sparse euclidean instances with vertices randomly generated in a 100×100 square; (iii) dense and (iv) sparse instances with weights randomly generated in [0,100]. For each class have been generated 36 instances with n growing from 10 to 80, two at a time. The density of the sparse instances ranges from 20% to 34%. The improved algorithm (see [18]) is shown to dominate the approach of [20]. The authors give percentage errors with respect to a first version of their lagrangian bound. They also compare the GA with their implementation of the SA algorithm of [12] and with a single run of the KL heuristic, implemented according to the original paper [13]. The average computational time for the 80 vertices instances is 5,917 seconds on a Prime 9955 minicomputer. Following Dongarra [7] this computer is at most 7 times slower than a PC486/DX2. Since we will show that our approach determines the optimal solution of instances with up to 100 vertices in less than one second on the PC486/DX2, we do not consider the genetic algorithms as potential competitors.

The tabu search TS presented in [21] has been tested on a set of 144 dense and sparse graphs as those used to test the GA. The authors do not give an explicit comparison with their genetic algorithms, but an inspection of the tables in the three papers shows that algorithm TS produces better solutions than GA and its computational times are small: up to 27 seconds on a PC486/DX2. In [21] the authors also report results for 5 instances with $n = 100, 200, \ldots, 500$ and compare TS with a single run of KL. Algorithm TS results to be always better than KL, so it is the natural competitor for our new tabu search algorithm.

The same kind of problems (144 instances from the four classes (i)–(iv)) have been used to test the lagrangian heuristic LAG of [19]. Again the authors do not give an explicit comparison with their other approaches, but examining the tables one can see that LAG have worse average performances with respect to TS. Moreover the computational times are very high: on average 6,366 seconds on a Prime 9955 are required for solving the instances with $n = 80$. As already done for GA we conclude that LAG is not to be considered a valid competitor for our tabu search algorithm.

Finally a short consideration is required for the exact approach of [2]. This algorithm has been able to solve various kind of weighted instances with up to 100 vertices, within 127,297 seconds on a SUN SPARC 10/41 workstation (this computer is 4 to 5 times faster than a PC486/DX2). It is evident that this algorithm cannot be used for large size problems. We will use the instances optimally solved in [2] to test the performances of the competitors on problems with $n \leq 100$.

2.3 Implementation details

One of the most time consuming operations is to explore the neighborhood in order to identify the most promising neighboring solution. With our neighborhood $PN(s)$ we want to swap the pair (a, b), with $a \in A$ and $b \in B$, which maximizes $D_a + D_b - 2w(a, b)$ (i.e. the decrement of the objective function value, if the exchange is executed). This is the same operation performed by each *elementary iteration* of the KL heuristic (here we call "elementary" an iterations of KL which finds a pair of vertices to be temporarily swapped; a *full iteration*, instead, determines the k pairs to be really swapped and consists of up to $n/2 - 1$ elementary iterations). The search for the best pair, in practice, is also the basic step of the slightest-swap procedure in the $GRASP$ approach. The only difference is that the slightest-swap have to determine the minimum positive decrement of the objective function.

Using a naive implementation which simply determines each pair (a, b) and compares its value with the best pair found so far, we need $O(n^2)$ time to explore the whole neighborhood. In the following we discuss the implementations proposed to speed-up this step and we point out the main differences between the 0–1 equicut and the weighted case.

Let us first consider the 0–1 equicut problem. With the naive implementation a "full iteration" of the KL heuristic requires $O(n^3)$ time. In order to have a better time complexity Kernighan and Lin suggested to sort the two sets $\{D_a : a \in A\}$ and

$\{D_b : b \in B\}$ by non-increasing values (time complexity $O(n \log n)$), and to evaluate the pairs of vertices by scanning down the two sets. More precisely let $MXA = \{a \in A : D_a = max_{i \in A} D_i\}$, $MX1A = \{a \in A : D_a = max_{i \in A} D_i - 1\}$, $MXB = \{b \in B : D_b = max_{j \in B} D_j\}$ and $MX1B = \{b \in B : D_b = max_{j \in B} D_j - 1\}$. After the sorting we have an immediate "good" candidate swap by selecting the first pair, say $(\widehat{a}, \widehat{b})$ (note that $\widehat{a} \in MXA$ and $\widehat{b} \in MXB$). If no edge exists between $\widehat{a}$ and $\widehat{b}$ the swap has the maximum possible gain among all the swaps of the neighborhood, and the search is terminated. Otherwise the value $D_{\widehat{a}} + D_{\widehat{b}} - 2$ of the current swap is used as a lower bound for the maximum gain. We continue the search looking for a pair (a, b) such that $a \in MXA, b \in MXB$ and no edge exists between a and b. If such a pair exists we are done, otherwise we look for a pair with gain $D_{\widehat{a}} + D_{\widehat{b}} - 1$. In any case the maximum number of pairs to be considered is $sw = |MXA| \times |MXB| + |MX1A| \times |MXB| + |MXA| \times |MX1B|$. Although one can easily define instances with $|MXA| = |MXB| = n/2$ (i.e. $sw = n^2/4$), in practice the number of pairs to be evaluated is quite small. Probably for this reason Kernighan and Lin [13] state that "... only a few likely contenders for a maximum gain need be considered", so they erroneously conclude that the time complexity of the improved algorithm is $O(n \log n)$ for each elementary iteration and $O(n^2 \log n)$ for a full iteration.

In [13] the authors also propose a not exact method which does not require to sort the two sets, but looks for at most three vertices in $MXA \cup MX1A$ and three in $MXB \cup MX1B$ and choose the pair to be swapped among the resulting nine pairs. The time complexity for an elementary iteration with this method is $O(n)$.

After each elementary iteration we have to update the D_i values, according to the swap (a, b). This is done by considering all the edges incidents in a and b, so giving an $O(|E|)$ time complexity for each full iteration. The overall time complexity of the full iteration, using the sorting technique and assuming that the value of sw is a negligible constant, is $O(n^2 \log n + |E|)$.

A faster version of the KL heuristic can be obtained, as described in [12], by using *bucket lists*. The main idea with this data structure is to partition the set of the elements to be considered into subsets, according to their values, and to store each subset in a different "bucket". More precisely the interval of all possible values assumed by the elements is partitioned in subintervals each of which is associated to a different bucket. The elements with value belonging to the same subinterval are stored in the same bucket. The buckets are ordered by increasing values using a list of pointers. For the 0–1 equicut the implementation of this structures is particularly simple and effective. Indeed the elements to be stored are the vertices and their values are the differences D_i, whose possible values are the $n-1$ integers in $[-n/2, n/2-1]$, therefore we can use subintervals of unitary length. We can implement a bucket list for set A (or set B) using only three arrays of n elements. The first array has one entry for each integer value in $[-n/2, n/2-1]$: each entry is a pointer to a double linked list which stores the elements of a bucket. The position of the entry gives the value common to all the elements in the associated bucket. The other two arrays of n elements are used to store the pointers of the double linked

list. The elementary operations to be performed with this data structure are the insertion and the removal of one element. Both operations require $O(1)$, therefore building the entire list requires $O(n)$ time. When the bucked lists have been constructed the sets MXA and $MX1A$ (resp. MXB and $MX1B$) are obtained by scanning down the list of pointers to the values D_i, starting from value $n/2-1$. When the first nonempty pointer is found, then the corresponding double linked list stores set MXA, the second nonempty pointer identifies set $MX1A$. Thus we can find the required sets with the improved time complexity $O(n)$ instead of $O(n \log n)$.

The computational experience of the first author on the 0–1 equicut (see [6]) confirms that using the bucket lists one can drastically reduce the running times of the various algorithms.

Now let us consider the equicut problem with generic weights on the edges. It is not difficult to see that the bucket lists loose their efficiency. Indeed, if edge $[\widehat{a},\widehat{b}]$ exists the pair $(\widehat{a},\widehat{b})$ has associated the value $D_{\widehat{a}}+D_{\widehat{b}}-2w(\widehat{a},\widehat{b})$. Since $w(\widehat{a},\widehat{b})$ may be large, better swaps can be obtained with pairs (a,b) having $D_a \ll D_{\widehat{a}}-1$ and $D_b \ll D_{\widehat{b}}-1$, but small (or null) weight $w(a,b)$. Therefore we cannot limit the search to the first buckets. Moreover, said U the maximum weight of an edge, the interval of the possible values for D_i is $[-nU/2, nU/2-1]$, and we need to define buckets of lengths larger than one (possibly different), so giving a more complicate code and a large overwork for maintaining the data structures. Finally, since the elements in each bucket are not sorted, we have to examine the entire bucket to find the better element. We have experimentally observed that the number of pairs of vertices considered in each iteration of the KL algorithm and in the search of a neighborhood, significantly increases when the instance is weighted.

For both the above reasons it results that efficient algorithms for the 0–1 equicut cannot be immediately transformed into algorithms for the weighted case, on the contrary algorithms for the weighted case used to solve the 0–1 equicut have generally poor performances. A confirm of this assertion can be found in [6] where it is shown that algorithm TS of [21], written for the weighted problem, it is not competitive for the 0–1 equicut.

Another difference between the algorithms for the two classes of problems is that algorithms for the 0–1 equicut can use ideas and strategies which are no more useful for the weighted case. This happens with the $GRASP$ approach of [16]: the slightest swap procedure used to optimize an initial solution is effective for the 0–1 equicut, where the idea of moving from a solution to a neighboring one which has the minimum possible difference of value, can lead to good solutions in a not too large amount of iterations. On the contrary in the weighted case this strategy requires a long time to obtain significative improvements in the value of the initial solution (see Section 4 for more details).

According to the results of the above reasoning we have implemented all the algorithms used for ours experiments with the sorting technique.

3 The new tabu search algorithm

In this section we present our new tabu search algorithm for the solution of the weighted equicut problem. We start by resuming the fundamental ideas which are the bases of the tabu search technique.

A tabu search algorithm can be seen as a development of a *local search* procedure. Given a (feasible) solution s we call *neighborhood* of s, and denote it with $\mathcal{N}(s)$, the set of all the solutions which can be obtained with a (simple) modification of s. Any modification which can lead from s to an element in $\mathcal{N}(s)$ is called a *move*. Given a starting solution s a local search based algorithm basically repeats the following steps: (a) compute $\mathcal{N}(s)$; (b) select a solution $s_{\mathcal{N}} \in \mathcal{N}(s)$; and (c) set $s = s_{\mathcal{N}}$, until a stopping criterion becomes true. A pure local search algorithm selects the solution with the best objective function value in $\mathcal{N}(s)$ and stops when such solution is not better than the current one. In this way it performs only improving moves. On the contrary a tabu search algorithm performs even non improving moves, but it imposes some restrictions on the solutions which can be selected. At each iteration the new current solution becomes the best among those in $\mathcal{N}(s)/\mathcal{TN}$, where $\mathcal{TN}$ denotes a set of forbidden (*tabu*) solutions. The quality of a solution is usually computed on the basis of its objective function value but, in more sophisticated implementations, it can be computed on the basis of information recorded during the evolution of the search (see e.g. Laguna and Glover [17] for a survey on tabu search techniques). The general elements which define a tabu search based algorithm are: (i) *memory structures*, which capture relevant information during the search process and which help in determining which solutions belong to $\mathcal{TN}$; and (ii) *strategies*, to use those information in the best possible way. Memory structures are normally classified in two types: *short term memory*, and *long term memory*.

We now describe in detail the main elements of our algorithm.

The short term memory

A *short term memory* is used to record *attributes* of those solutions recently generated in the evolution of the search. This is done to have a (not exact) representative of a solution which requires a small amount of memory to be stored, and a short time to be compared with representatives of other solutions. The attributes of the solutions already visited are used to identify the tabu solutions.

In our algorithm the structure of the neighborhood $PN(s)$ of a solution $s = (A, B)$ is defined by all the moves which interchange a pair of vertices, one from set A and one from set B. Since our algorithm starts the search from a feasible solution it always considers only feasible partitions.

At each iteration we associate to the current solution a couple of attributes: the names and the starting sets (A or B) of the two vertices interchanged by the move. For example, if a move interchanges a from A and b from B we record the attributes "a was in A" and "b was in B". We have implemented the short term memory using two *tabu*

lists: *was_in_A* and *was_in_B*. If the selected vertices are, as above, $a \in A$ and $b \in B$ we put a in *was_in_A* and b in *was_in_B*. For a number of iterations all the selected moves which propose the removal of a from B (or of b from A) are forbidden.

In our implementation the tabu tenure l of the short term memory, i.e. the number of iterations a solution maintains its tabu status or, in our case, the number of iterations a vertex remains in a tabu list, has not a fixed value. Indeed, greater is the value of l, bigger is the set of the forbidden solutions $\mathcal{TN}$ and consequently greater is the number of tabu moves. We set l to the initial value *tabu_tenure* and we modify its value according to the evolution of the search process within an interval of the same length: *tabu_tenure*. The aim of decreasing l is that of *locally* intensifying the search within those regions which are more promising, whilst the aim of increasing l is that of speeding up the leaving from those regions which surround already visited local minima. Let us call an *improving phase* a set of Δip consecutive iterations which lower the objective function value: after any improving phase l is set to $\max(l - 1, \frac{1}{2} tabu\_tenure)$. On the contrary let us call a *worsening phase* a set of Δwp consecutive iterations such that the objective function value does not improve: after any worsening phase l is set to $\min(l + 1, \frac{3}{2} tabu\_tenure)$.

To implement the short term memory structure the algorithm requires only two vectors of n integers. Basing ourselves on a set of preliminary experiments we fixed the value of *tabu_tenure* to 10, and the values of Δip and Δwp to 5 and 3 respectively.

Let us observe that the set $\mathcal{TN}$ of the tabu moves contains as a subset those moves which perform the exact reversal of recently swapped vertices. For this reason in a preliminary version of the algorithm we assigned a tabu status with a longer tabu tenure to such moves. The memory structure for this subset of moves was implemented with a square $(n \times n)$ integer matrix $[t_{ij}]$: if the current move performs the swap of the vertices (a, b) then we assign the number of the current iteration to the cell $t_{a,b}$ of the matrix and we use this information, for a number of following iterations, to forbid the execution of the move which swaps the vertices (b, a). Since, after an extensive set of experiments, we did not observe a sensible improvement in the quality of the solutions which we could obtain with this approach, it was discarded.

As in any standard implementation of a tabu search based algorithm we choose at each iteration the best not-tabu solution in the neighborhood, i.e. the not-tabu pair with largest gain $g = D_a + D_b - 2w(a, b)$ among all pairs (a, b) with $a \in A$ and $b \in B$. This is accomplished, as explained in the previous section, in such a way that not all the pairs are explicitly evaluated.

The aspiration criteria

In tabu search algorithms some *aspiration criteria* are usually introduced to override the tabu status of those moves which are supposed to lead to not already visited or promising solutions. This mechanism is necessary because a few attributes are not enough to completely identify a solution: i.e. a recorded set of attributes is usually shared both by already visited solutions and by unexplored ones. In our implementation we adopted

two aspiration criteria: the first one simply consists of accepting a tabu move if it leads to a solution with an objective function value better than that of the best solution found so far. The second criterion takes into consideration the value of the objective function of three solutions. Let s_{old}, s and s_{new} denote three consecutive solutions (i.e. s has been obtained from s_{old} with a single move and s_{new} belongs to the neighborhood of s). If s is the current solution and $z(s_{new})$ is strictly less than $\min(z(s_{old}), z)$ then any possible tabu restriction on s_{new} is not considered. Using this criterion we limit the restrictions imposed by the two tabu lists to take into account the cases in which, after the swap of two vertices, e.g. (a, b), and the execution of at least another swap not involving a or b, a good solution could be reached by a swap involving either a or b but not both of them.

A similar criterion was proposed in [6] for the 0–1 case. In that paper $z(s_{new})$ has to be strictly less than $\min(z, \alpha z(s_{old}))$, where α was set to 0.99. For the weighted case we also implemented the same criterion and we considered values of α ranging from 0.99 down to 0.85 without observing notable differences with the case in which α was set to 1. For this reason we have chosen the simpler technique discarding the parameter α.

The long term memory

In the 0-1 equicut problem many solutions in a given neighborhood can have quite similar objective function values. Such a behavior can be observed even in the weighted case. For this reason we decided to adopt a *long term memory* structure. In this structure we record some of those high quality solutions which were analyzed in some neighborhood but whose value was worst than that of the selected solution. This memory structure, denoted as *Second*, is implemented with an ordered list of fixed length L. Let s_2 denote the not-tabu solution with the best *second* value among those evaluated in the current neighborhood. At each iteration s_2 can be added to the list *Second* if the list is not full or if the worst solution in *Second*, say s_w, has an objective function value worst than that of s_2: in this case s_w is removed from *Second*. When one of three particular conditions, explained later, holds we remove from *Second* the solution with the best objective function value and we continue the search from that solution. In order to restart the search with the same conditions which were present when a solution was put into the list, we associate to each solution in *Second* a copy of the two complete tabu lists and the values of the other parameters.

Note that we do not identify a solution to be added to *Second* in each neighbourhood, since we take advantage of the fast exploration of neighbourhoods presented in the previous section. With this implementation can happen (especially with sparse graphs) that the best solution is the first evaluated, and if it is not-tabu the analysis of the neighborhood can stop. We lost this advantage if we have to continue the search looking for the second best solution. Therefore in these cases no second best solution is evaluated and no updating of the list *Second* is done. The effectiveness of this implementation was proved with a set of preliminary experiments.

The use of restarting from a solution in *Second* has the aim of *intensifying* the search

into not fully explored promising regions. On the other hand the rule of adding to *Second* at most one solution from each neighborhood plays the role of a *diversification* strategy.

In a different set of experiments we studied a version in which, at each iteration, the algorithm evaluates the whole neighbourhood and it updates *Second* with all the solutions $\hat{s}$ such that $z(\hat{s}) < z(s_w)$. In this way *Second* could contain more than one solution belonging to the same neighborhood and the attention is posed only on the quality of the solutions and not on where they have been found. As a consequence the role of the intensification dominated over that of the diversification with an observed decreasing in the overall performances of the algorithm.

In the current version the best solution in *Second* is adopted as a restarting point of the search when one or more of the following conditions is verified:

1. all the solutions in the current neighborhood are tabu and none of them satisfies an aspiration criterion;

2. for a sequence of MC consecutive moves the current objective function value does not improve;

3. for a sequence of MB consecutive moves there has been no improving of the best solution found.

For the parameters involved in the management of the long term memory structure we adopted the following values: L was experimentally set to the minimum between 5 and the average vertex degree, MC was set to 7 and MB was set to 200.

In a set of experiments we also tried to make a flexible use of the long term memory, with a dynamic updating similar to that used for the tabu tenure of the short term memory. We initially set MC to 5 and we increased it up to 10 when a threshold number of solutions from the list *Second* had been used to restart the search. The aim was to reduce the role of the diversification in all the cases in which it seemed to overcome that of the intensification. With the same aim MB was increased to 300, so imposing a restarting only when an extensive and (fruitless) exploration of the current region had been performed. Our experience is that this more complicate way of managing the long term memory does not determine significant improvement in the solution quality.

Another strategy based on long term memory is the following. We impose that every solution with an objective function value equal to that of the current best solution has to be considered tabu. This is done to prevent a cycling behavior on the long term.

Restarting the search

In all cases in which the algorithm requires a solution from the list *Second* but the list is empty the algorithm restarts the search from a new generated solution. This is a way of implementing a *diversification* strategy. The procedure adopted for randomly generating feasible starting solutions is that presented in [6]. We restate it for sake of completeness. Two seed vertices, one from set A and one for set B are selected. Then,

for $n/2 - 1$ iterations select the pair (a, b) of not yet selected vertices which minimize the increment of the objective function value if a is assigned to A and b is assigned to B, and assign them to A and B respectively. In order to enforce the exploration of truly different regions of the space of the solutions the seed vertices are selected in such a way that the algorithm, can produce up to $n(n-1)/2$ different feasible partitions.

From the experience with the flexible use of the long term memory we decided to enforce this kind of diversification strategy by imposing that no more than L (see above) *consecutive* previously evaluated solutions, from list *Second*, can be used as starting point of the search. This apparent use of the brute force was found quite effective especially when considering those instances with one thousand of vertices or more. In that cases the restarting from new solutions spreads out the search whilst the adaptive use of the short and long term memory allows a fast exploration of each new region. Since the procedure which generates the feasible solutions is fast then we save the time required by a single start algorithm for identifying and traversing the "no man's land" which can separate the regions containing good quality solutions.

The pseudo code

Here we present the pseudo code of our algorithm called $WEC\text{-}TS$. In step 1 we select a starting feasible solution. In steps 2 and 3 we initialize the short and the long term memory structures, respectively. In step 4 we initialize the counters used to determine the improving and worsening phases, i.e. imp and wor, and that for counting the number of steps without improving the current best solution, gw. They are updated in steps 16 and 20, respectively. In step 5 we initialize the boolean variable which allows the use of the list *Second*, whilst in step 6 we initialize the variable used to determine the second aspiration criterion, it is updated in step 19. If it is necessary, in steps 7 and 20, we update the best solution found so far. In step 8 it starts the main loop: a single run can consume up to max_time seconds. In steps 9 and 10 we test if there exists at least one solution in the current neighbourhood $PN(s)$ which is not tabu or which satisfies an aspiration criterion. If such a solution does not exist we try to use the long term memory, step 22. In step 11 we update the short memory structures. In steps 12–15 we manage the long term memory structure. In steps 16–18 the tabu tenure of the short term memory is updated according to the rules explained above. In step 21 we determine if the current state of the search requires the use of the long term memory. In steps 23–30 we try to use the long term memory, if it is necessary. In step 25 we determine if the list *Second* is not empty and if it has not been used more than L times. If we cannot use the long term memory we restart the search from a new generated solution, step 30, otherwise we restart the search from the best solution recorded in the list *Second*, and we update the short term memory accordingly (steps 26–29). The algorithm returns the best solution found during the search process.

Procedure *WEC-TS*

$z^* := \infty$;

1. L1: generate a new starting solution s;
2. initialize lists *was_in_A* and *was_in_B*; $l := tabu\_tenure$;
3. $Second := \emptyset$; $num\_Second := 0$;
4. $imp := 0$; $wor := 0$; $gw := 0$;
5. $second\_choice := FALSE$;
6. $z_{old} := \infty$;
7. **if** $z^* > z(s)$ **then** $z^* := z(s)$; $s^* := s$; **endif**;
8. **while** elapsed_time < max_time **do**
9. select the best solution $s' \in PN(s)$ which is not tabu or which satisfies an aspiration criterion; possibly determine the second best solution s_2;
10. **if** $s' \neq \emptyset$ **then**
11. update *was_in_A* and *was_in_B*;
12. **if** $s_2 \neq \emptyset$ **then**
13. **if** $|Second| < L$ **then** put s_2 in $Second$;
14. **else** find s_w the solution with higher value in $Second$;
15. **if** $z(s_2) < z(s_w)$ **then** substitute s_w with s_2 in $Second$;
    endif;
    endif;
16. **if** $z(s') < z(s)$ **then** $imp := imp + 1$; $wor := 0$;
    else $wor := wor + 1$; $imp := 0$; **endif**
17. **if** $imp = \Delta ip$ **then** $l := \max(l - 1, \frac{1}{2} tabu\_tenure)$; **endif**;
18. **if** $wor = \Delta wp$ **then** $l := \max(l + 1, \frac{3}{2} tabu\_tenure)$; **endif**;
19. $z_{old} := z(s)$; $s := s'$;
20. **if** $z^* > z(s)$ **then** $z^* := z(s)$; $s^* := s$; $gw := 0$;
    else $gw := gw + 1$;
21. **if** $gw \geq MB$ or $imp \geq MC$ **then** $second\_choice := TRUE$; **endif**;
22. **else** $second\_choice := TRUE$;
23. **if** $second\_choice$ **then**
24. $second\_choice := FALSE$;
25. **if** $Second \neq \emptyset$ and $num\_Second < L$ **then**
26. choose the solution s' with better value in $Second$;
27. reset *was_in_A* and *was_in_B* with the values stored with s';
28. $num\_Second := num\_Second + 1$;
29. $imp := 0$; $wor := 0$; $s := s'$;
30. **else** goto L1; ($\star$ restarting $\star$)
    endif;
    end while;
31. **return** s^*;

4 Computational Experiments

The tabu search *WEC-TS* of the previous section has been implemented in ANSI C language and tested on a PC 486/66 with 8 Mbyte of main memory. We have also implemented the *GRASP* approach of [16] and a multistart algorithm *M-KL* which iteratively generates a random starting solution and applies to it the *KL* heuristic. For all these three algorithms, we have used the same implementation technique, described in section 2.3, to explore the neighborhood. We have also made a line by line translation from Pascal to C of the original source code of algorithm *TS* of [21] modifying only the stopping criterion. We removed the limit of the $\max(100, 5n)$ iterations and added a bound on the elapsed CPU time.

We first tested the four algorithms on the 259 instances from the library created for the computational experiments of [2]. The first 250 instances of this library are randomly generated instances from five classes: (i) *pure random instances with a fixed percentage of edges*; (ii) *planar grid instances*; (iii) *toroidal grid instances*; (iv) *mixed grid instances* (dense instances with some edges having weights from 1 to 100 and the others having weights from 1 to 10); (v) *instances with negative weights*. The remaining nine instances are real-word instances arising from an application of the finite element method in fluids (see [2] for further details). The problems have from 20 to 100 vertices and density from 3% to 100%. For all the problems the optimal solution value is given. We initially ran our algorithms for two seconds: our tabu search *WEC-TS* and the multistart approach *M-KL* determined the optimal solution, for each of the 259 instances, in less than one second. The two remaining approaches have not been able to optimally solve all the instances, therefore we ran again algorithms *GRASP* and *TS* with a time limit of 60 seconds. With the new run also *TS* determined the optimal solution for all the previously unsolved instances within 46 seconds, whilst *GRASP* has not been able to solve 15 instances. In a final run with a time limit of 120 seconds *GRASP* could optimally solve two more instances out of the 15 not yet solved.

From this computational experience we conclude that small (up to 100 vertices) instances of the weighted equicut problem are quite easy to solve with at least two heuristic approaches. On the contrary to prove the optimality of a solution is an hard task since it requires up to 127,297 seconds on a workstation SUN SPARC 10/41 (see again [2]).

To have a better understanding of the behaviour of the heuristic algorithms we randomly generated larger instances. In particular we tried to solve instances defined by *random graphs* $G_{n,p}$ and *geometric graphs* $U_{n,d}$ (see section 2.2 and [12]). The instances considered have $n = 124, 250, 500, 1000$ and $\hat{\nu} = 2.5, 5, 10, 20, 40, 80$, where ν is the expected average vertex degree. For the random graphs we considered three classes of weights for the edges: (α) $w_{ij} \in [1, 10^2]$; (β) $w_{ij} \in [1, 10^3]$; and (γ) $w_{ij} \in [1, 10^5]$. (For the geometric graphs the weights are the euclidean distance of two points in a unitary square.) For each class we generated and solved 20 instances giving to each algorithm a time limit of 4, 12, 36 and 108 seconds for $n = 124, 250, 500$ and 1000, respectively (we have experimentally observed that the average time required to obtain good solutions

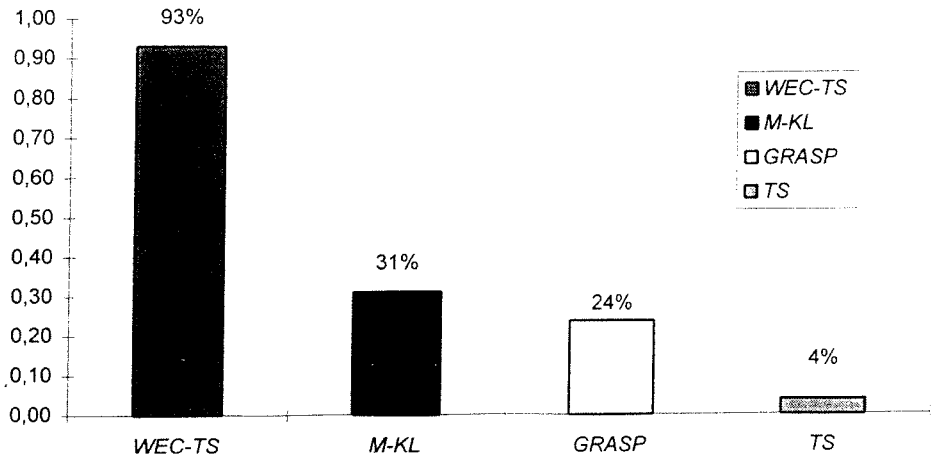

Figure 1: Percentage number of best solution found (BT): grand total

grows approximately with $n^{1.5}$).

To compare the performances of the four algorithms we used the *number of best solution found* (BT) and the *average error* with respect to the best solution value obtained by the four algorithms (avgErr). In the appendix we report six tables with all the details of our computational results. In that tables we also give the time required to find the best solution (Tbt) and the maximum error with respect to the best solution value obtained by the four algorithms (maxErr). Here we discuss the main results by means of four pictures.

In Figure 1 we summarize the results for the first index: we give the percentage number of best solution found, over all the 1920 instances on random and geometric graphs. The sum of the percentages is greater than 100 since for some instances more algorithms obtained the same solution value.

It results that $WEC\text{-}TS$ obtains a much larger number of good solution than the other approaches. The second best heuristic is the classical $M\text{-}KL$, which has performances a little better than $GRASP$. Algorithm TS is not competitive. Worth is noting the difference with the 0–1 case were $GRASP$ is better the $M\text{-}KL$.

In Figures 2 and 3 we report the results obtained for the random graphs (1440 instances) and for the geometric graphs (480 instances). The performances of $WEC\text{-}TS$, in practice, do not change. Algorithms $M\text{-}KL$, $GRASP$ and TS, instead, have a different behaviour for the two classes: they determine less good solutions for random graphs than they do for geometric graphs (about three time less for $M\text{-}KL$, almost four times less for $GRASP$, whereas TS determines a best solution for only one random instance).

In Figure 4 we report the average error (with respect to the smallest solution value) for random graphs (the same index cannot be used for geometric graphs, since there are often solutions with zero or very small value, and the error tents to infinity).

It results again that $WEC\text{-}TS$ has excellent performances, followed by $GRASP$

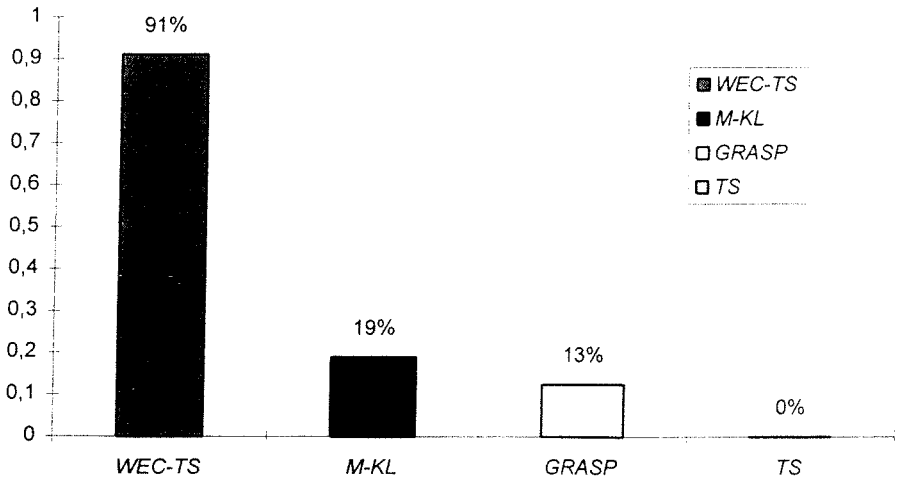

Figure 2: Percentage number of best solution found for random graphs

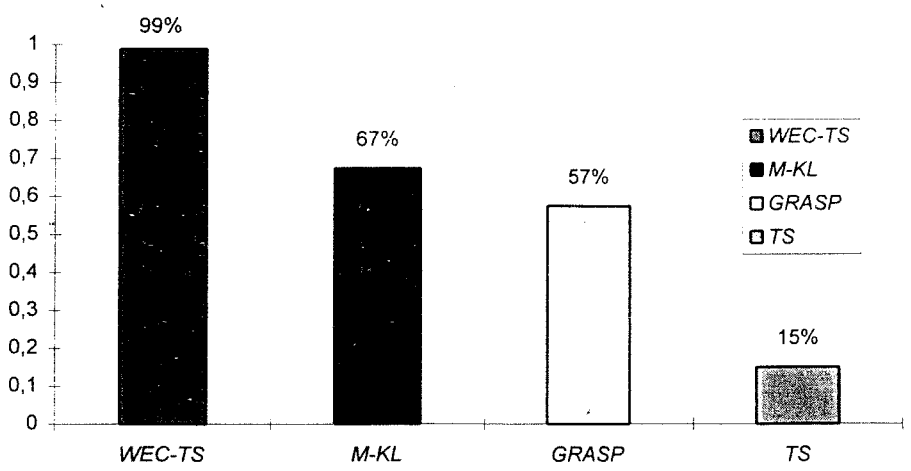

Figure 3: Percentage number of best solution found for random graphs

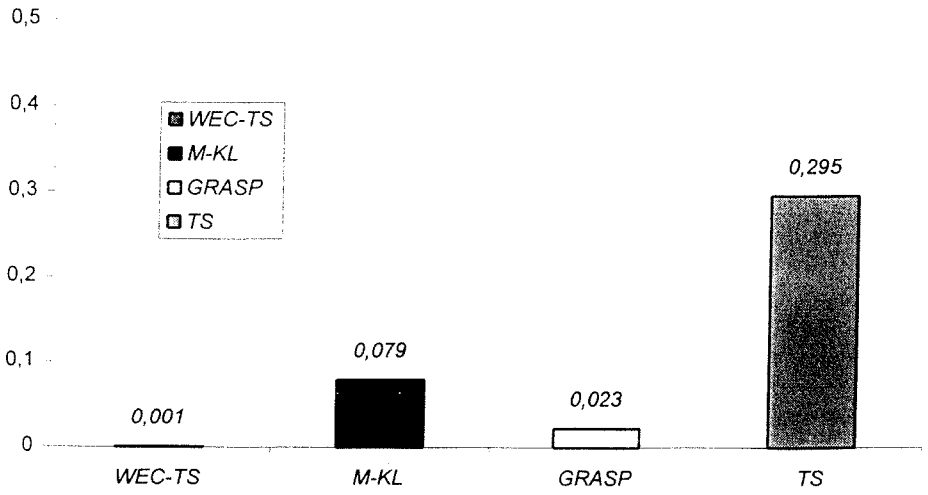

Figure 4: Average errors for random graphs

and $M\text{-}KL$. Note that $M\text{-}KL$ determines more better solutions than $GRASP$, on the random graph, but its average error is larger. On the geometric graphs $M\text{-}KL$ has better performances than $GRASP$. For the 0-1 instances, instead, $GRASP$ is comparable with $M\text{-}KL$ for the random graphs, but it outperforms $M\text{-}KL$ for the geometric graphs (see [6]). Moreover observe that $GRASP$ generally uses all its time to find its better solution (see tables I–IV in the Appendix). This confirms our hypothesis on the behaviour of the $GRASP$ approach (see Section 2.3).

Finally we note that when the range of the weights changes form $[1, 10^2]$ to $[1, 10^5]$ (see tables I–III in the Appendix), the behaviour of the four algorithm remains almost the same, showing a substantial robustness of all the approaches.

Motivated by the above results we have generated larger instances ($n = 2000$ and $n = 4000$) to study the characteristics of the algorithms with very large weighted equicut problems.. We considered random and geometric graphs, the same six values of $\hat{\nu}$ used above, and the range $[1, 10^2]$ for the weights of the edges. It was necessary to change a little the codes since our PC has only eight Mbytes of RAM. Indeed, with the original implementation we have used both a full $n \times n$ matrix and a *forward star* data structure to store the costs of the edges. With this implementation we need $O(1)$ time to determine the weight of a single edge and $O(|E|)$ time to update the D_j values (see Section 2.3). With the larger instances we had to remove the $n \times n$ matrix, so finding the weight of a single edge requires $O(\log \hat{\nu})$ time, using a binary search. Sample experiments performed with the new implementation, on instances with $n \leq 1000$, have shown that the number of iterations executed by the algorithms, within a given amount of time, in practice do not change. For the large problems we have generated and solved five instances for each different kind of graph, giving to each algorithm a time limit of 324 and 972 for $n = 2000$ and $n = 4000$, respectively. The results for the large instances are similar to those obtained for $n \leq 1000$, just observe that for random graphs $M\text{-}KL$ determines

more better solutions than $GRASP$, but its average error is far larger.

Acknowledgment

We would like to thank professor Francesco Maffioli for a number of usefull discussions and suggestions. We are also indebted with Dr. Erik Rolland who make us available the Pascal version of algorithm TS, and with Dr. Lorenzo Brunetta who gave us the library of the 259 instances exactly solved.
This work has been partially supported by research contracts MPI 40% and 60% of the Italian Ministry of University and Scientific Research.

References

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

[2] L. Brunetta, M. Conforti, and G. Rinaldi. "A branch-and-cut algorithm for the equicut problem", *Mathematical Programming*, to appear.

[3] M. Conforti, M. Rao, and A. Sassano. "The equipartition polytope I: Formulation, dimension and basic facets", *Mathematical Programming* 49 (1990) 49–71.

[4] M. Conforti, M. Rao, and A. Sassano. "The equipartition polytope II: Valid inequalities and facets", *Mathematical Programming* 49 (1990) 71–90.

[5] W. Dai and E. Kuh. "Simultaneous floor planning and global routing for hierarchical building block layout", *IEEE Transactions on CAD ICs Systems* (1987).

[6] M. Dell'Amico and F. Maffioli. "A new tabu search approach to the 0–1 equicut problem", in: I.H. Osman and P. Kelly (eds), *Meta-Heuristics 1995: The State of the Art*, Kluwers Academic Publishers, to appear.

[7] J. Dongarra. "Performances of various computers using standard linear equations software", CS-89-85, University of Tennesee, Computer Science Department, Knoxville, 1995.

[8] A. Dunlop and B. Kernighan. "A procedure for placement of standard-cell vlsi circuits", *IEEE Transactions on CAD ICs Systems* 1 (1985) 92–98.

[9] T. A. Feo and M. Kellaf. "A class of bounded approximation algorithms for graph partitioning", *Networks* 20 (1990) 181–195.

[10] C. Fiduccia and R. Mattheyses. "A linear-time heuristic for improving network partitions", in: ACM/IEEE *19-th Design Autom. Conference* Las Vegas (1982) 175–181.

[11] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

[12] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon. "Optimization by simulated annealing: An experimental evaluation; Part I, graph partitioning", *Operations Research* 37 (1989) 865–892.

[13] B. W. Kernighan and S. Lin. "An efficient heuristic procedure for partitioning graphs", *Bell System Technical Journal* 49 (1970) 291–307.

[14] S. Kirkpatric. "Optimization by simulated annealing: Quantitative studies", *Journal of Statistical Physics* 34 (1984) 975–986.

[15] S. Kirkpatric, C. D. Gelatt Jr., and M. Vecchi. "Optimization by simulated annealing", *Science* 220 (1983) 671–680.

[16] M. Laguna, T. A. Feo, and H. Elrod. "A greedy randomized adaptive search procedure for the two-partition problem", *Operations Research* 42 (1994) 677–687.

[17] M. Laguna and F. Glover. "Tabu search", in: C. R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, Oxford, 1993.

[18] H. Pirkul and E. Rolland. "New heuristic solution procedures for the uniform graph partitioning problem: Extensions and evaluation", *Computers & Operations Research* 21 (1994) 895–907.

[19] H. Pirkul and E. Rolland. "A lagrangean based heuristic uniform graph partitioning problem", Working Paper, University of California, Riverside, 1995.

[20] E. Rolland and H. Pirkul. "Heuristic solution procedures for the graph partitioning problem", in: O. Balci, R. Sharda, and S. Zenios (eds.), *Computer Science and Operations Research: New Developments in Their Interfaces*, Pergamon Press, Oxford, 1992.

[21] E. Rolland, H. Pirkul, and F. Glover. "A tabu search for graph partitioning", in: Metaheuristics in optimization, *Annals of Operation Research*, to appear.

Appendix

Table I. Random graphs; class (α): range = [1,10^2].

		124				250			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	17	2	6	0	15	0	10	0
	Tbt	1.054	2.383	1.233	1.561	4.806	5.582	4.577	2.614
	maxErr	0.039	0.442	0.157	1.951	0.026	0.514	0.047	1.724
	avgErr	0.003	0.099	0.034	0.953	0.004	0.295	0.012	1.185
5	BT	20	10	6	0	16	3	1	0
	Tbt	0.599	1.823	1.911	1.848	6.449	5.624	7.212	3.746
	maxErr	0.000	0.041	0.055	0.345	0.026	0.068	0.100	0.338
	avgErr	0.000	0.008	0.015	0.235	0.002	0.040	0.050	0.259
10	BT	19	11	5	0	20	0	0	0
	Tbt	0.627	1.446	1.658	2.049	5.326	6.103	6.975	1.960
	maxErr	0.002	0.015	0.023	0.157	0.000	0.036	0.053	0.206
	avgErr	0.000	0.004	0.008	0.081	0.000	0.013	0.028	0.129
20	BT	20	16	7	1	18	2	1	0
	Tbt	0.481	1.346	1.716	0.679	4.548	6.333	6.335	2.552
	maxErr	0.000	0.003	0.015	0.186	0.009	0.013	0.026	0.110
	avgErr	0.000	0.000	0.004	0.062	0.000	0.005	0.015	0.060
40	BT	20	18	4	0	19	3	0	0
	Tbt	0.508	1.429	1.864	1.636	3.721	5.784	6.615	4.616
	maxErr	0.000	0.003	0.008	0.045	0.000	0.008	0.011	0.067
	avgErr	0.000	0.000	0.003	0.022	0.000	0.003	0.008	0.036
80	BT	20	15	12	0	18	4	1	0
	Tbt	0.579	2.082	1.834	2.097	4.372	6.681	8.144	7.687
	maxErr	0.000	0.001	0.003	0.035	0.001	0.005	0.011	0.036
	avgErr	0.000	0.000	0.000	0.013	0.000	0.001	0.005	0.017

		500				1000			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	15	0	7	0	16	0	4	0
	Tbt	12.935	19.838	20.372	5.077	51.579	63.304	44.556	25.418
	maxErr	0.036	1.021	0.083	2.089	0.014	1.052	0.099	2.107
	avgErr	0.004	0.547	0.028	1.425	0.002	0.688	0.032	1.375
5	BT	19	0	1	0	20	0	0	0
	Tbt	20.401	15.759	21.801	18.120	54.384	46.799	90.507	14.952
	maxErr	0.006	0.096	0.063	0.400	0.000	0.117	0.077	0.361
	avgErr	0.000	0.063	0.038	0.294	0.000	0.072	0.043	0.294
10	BT	17	3	0	0	20	0	0	0
	Tbt	16.062	17.210	25.140	3.684	53.739	68.631	99.758	19.774
	maxErr	0.006	0.038	0.061	0.207	0.000	0.037	0.056	0.161
	avgErr	0.000	0.019	0.036	0.133	0.000	0.018	0.033	0.122
20	BT	19	1	0	0	19	1	0	0
	Tbt	18.450	18.447	23.121	6.571	53.433	66.616	107.102	27.634
	maxErr	0.005	0.026	0.037	0.134	0.003	0.018	0.046	0.146
	avgErr	0.000	0.008	0.026	0.075	0.000	0.008	0.030	0.065
40	BT	20	0	0	0	19	1	0	0
	Tbt	14.219	21.019	28.653	13.854	36.581	58.076	107.909	45.311
	maxErr	0.000	0.011	0.025	0.063	0.001	0.010	0.036	0.061
	avgErr	0.000	0.005	0.014	0.036	0.000	0.005	0.023	0.041
80	BT	18	2	0	0	17	3	0	0
	Tbt	13.231	23.687	27.601	23.350	36.651	48.904	108.000	84.923
	maxErr	0.001	0.006	0.017	0.047	0.001	0.006	0.021	0.035
	avgErr	0.000	0.003	0.010	0.027	0.000	0.003	0.015	0.023

PC486/DX2 seconds; 20 instances for each entry.

Table II. Random graphs; class (α): range = [1,10^3].

		124				250			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	14	4	5	0	15	0	8	0
	Tbt	1.237	2.254	0.881	1.601	5.004	5.888	4.395	3.579
	maxErr	0.041	0.184	0.067	1.975	0.038	0.584	0.063	1.730
	avgErr	0.005	0.060	0.027	0.916	0.005	0.318	0.020	1.226
5	BT	19	11	5	0	16	2	2	0
	Tbt	1.118	1.473	1.608	1.865	7.058	6.044	6.153	3.669
	maxErr	0.015	0.021	0.055	0.384	0.028	0.083	0.101	0.318
	avgErr	0.001	0.005	0.019	0.252	0.002	0.037	0.046	0.260
10	BT	20	8	3	0	18	2	0	0
	Tbt	0.482	1.396	1.719	2.171	3.890	5.713	7.528	1.547
	maxErr	0.000	0.012	0.022	0.176	0.008	0.036	0.054	0.207
	avgErr	0.000	0.003	0.011	0.077	0.001	0.013	0.029	0.131
20	BT	20	16	6	0	18	3	0	0
	Tbt	0.489	1.432	1.917	0.679	5.219	7.148	6.863	2.738
	maxErr	0.000	0.001	0.013	0.187	0.001	0.023	0.030	0.098
	avgErr	0.000	0.000	0.004	0.061	0.000	0.005	0.013	0.052
40	BT	20	16	9	0	20	2	0	0
	Tbt	0.564	1.789	1.936	1.537	4.242	6.001	6.929	3.301
	maxErr	0.000	0.004	0.008	0.042	0.000	0.006	0.015	0.081
	avgErr	0.000	0.000	0.001	0.022	0.000	0.002	0.008	0.039
80	BT	20	11	6	0	19	0	1	0
	Tbt	0.274	1.761	1.659	2.128	4.232	4.979	6.037	7.464
	maxErr	0.000	0.003	0.004	0.023	0.001	0.004	0.010	0.037
	avgErr	0.000	0.000	0.001	0.012	0.000	0.002	0.004	0.021

		500				1000			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	16	0	5	0	15	0	5	0
	Tbt	15.247	22.164	14.510	2.947	49.554	61.248	41.057	15.430
	maxErr	0.028	0.980	0.120	2.123	0.041	1.120	0.112	2.187
	avgErr	0.003	0.559	0.039	1.441	0.006	0.712	0.049	1.440
5	BT	17	2	1	0	20	0	0	0
	Tbt	16.359	20.199	21.027	19.172	52.427	62.684	85.438	12.792
	maxErr	0.008	0.116	0.096	0.396	0.000	0.155	0.138	0.386
	avgErr	0.001	0.057	0.037	0.275	0.000	0.069	0.049	0.293
10	BT	19	1	0	0	19	1	0	0
	Tbt	18.734	20.894	21.653	4.227	63.988	63.712	104.719	14.303
	maxErr	0.007	0.042	0.062	0.204	0.003	0.035	0.066	0.222
	avgErr	0.000	0.019	0.037	0.138	0.000	0.019	0.040	0.146
20	BT	18	2	0	0	18	2	0	0
	Tbt	15.963	21.069	24.538	9.504	40.417	71.511	108.000	34.377
	maxErr	0.001	0.023	0.050	0.104	0.003	0.020	0.048	0.100
	avgErr	0.000	0.009	0.026	0.065	0.000	0.008	0.035	0.061
40	BT	17	3	0	0	19	1	0	0
	Tbt	12.950	19.769	31.084	14.665	49.780	54.397	107.994	46.898
	maxErr	0.003	0.011	0.025	0.081	0.002	0.011	0.033	0.067
	avgErr	0.000	0.005	0.015	0.035	0.000	0.005	0.024	0.037
80	BT	15	5	0	0	18	2	0	0
	Tbt	15.558	19.137	25.942	26.362	54.716	68.681	108.000	90.990
	maxErr	0.003	0.006	0.014	0.029	0.001	0.006	0.020	0.034
	avgErr	0.000	0.002	0.009	0.021	0.000	0.003	0.016	0.022

PC486/DX2 seconds; 20 instances for each entry.

Table III. Random graphs; class (α): range = $[1,10^5]$.

		124				250			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	14	3	6	0	16	1	7	0
	Tbt	1.071	2.396	1.049	1.408	5.642	5.595	4.407	3.052
	maxErr	0.046	0.200	0.093	2.101	0.035	0.482	0.063	1.737
	avgErr	0.006	0.070	0.027	0.923	0.004	0.300	0.021	1.225
5	BT	19	10	6	0	18	1	1	0
	Tbt	0.759	1.588	1.633	1.973	7.450	6.430	6.550	5.035
	maxErr	0.015	0.025	0.055	0.399	0.008	0.083	0.101	0.339
	avgErr	0.001	0.006	0.019	0.248	0.001	0.035	0.048	0.277
10	BT	20	8	3	0	18	1	1	0
	Tbt	0.577	1.404	1.944	0.841	5.236	4.988	6.987	0.979
	maxErr	0.000	0.012	0.019	0.230	0.007	0.026	0.054	0.207
	avgErr	0.000	0.003	0.010	0.139	0.000	0.010	0.027	0.137
20	BT	20	16	6	0	19	2	0	0
	Tbt	0.540	1.429	2.200	0.384	5.753	6.092	6.192	1.586
	maxErr	0.000	0.001	0.026	0.187	0.001	0.014	0.029	0.109
	avgErr	0.000	0.000	0.005	0.078	0.000	0.006	0.015	0.061
40	BT	20	17	8	0	19	2	0	0
	Tbt	0.497	1.778	1.936	0.537	4.241	6.226	6.734	2.322
	maxErr	0.000	0.004	0.009	0.068	0.000	0.008	0.016	0.081
	avgErr	0.000	0.000	0.002	0.044	0.000	0.003	0.008	0.049
80	BT	20	13	8	0	19	0	1	0
	Tbt	0.382	1.809	1.837	0.850	3.563	5.910	6.271	4.717
	maxErr	0.000	0.001	0.003	0.063	0.001	0.005	0.009	0.042
	avgErr	0.000	0.000	0.001	0.026	0.000	0.002	0.005	0.025

		500				1000			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	16	0	5	0	15	0	5	0
	Tbt	16.774	20.693	13.783	5.457	49.828	57.013	49.112	15.536
	maxErr	0.066	1.050	0.127	2.125	0.023	1.122	0.112	2.206
	avgErr	0.006	0.546	0.040	1.441	0.003	0.704	0.041	1.444
5	BT	19	0	1	0	20	0	0	0
	Tbt	18.399	21.979	20.916	16 277	50.091	43.714	94.201	14.432
	maxErr	0.004	0.102	0.072	0.377	0.000	0.147	0.137	0.386
	avgErr	0.000	0.060	0.039	0.277	0.000	0.076	0.051	0.297
10	BT	19	0	1	0	18	2	0	0
	Tbt	23.989	16.518	26.027	4.664	54.794	63.427	104.130	13.350
	maxErr	0.001	0.043	0.064	0.203	0.014	0.032	0.070	0.221
	avgErr	0.000	0.018	0.037	0.135	0.001	0.019	0.042	0.141
20	BT	18	2	0	0	18	2	0	0
	Tbt	14.573	18.997	25.782	7.122	44.647	64.570	108.000	26.199
	maxErr	0.002	0.020	0.040	0.090	0.003	0.016	0.045	0.100
	avgErr	0.000	0.010	0.024	0.068	0.000	0.008	0.034	0.063
40	BT	20	0	0	0	19	1	0	0
	Tbt	8.597	20.051	25.521	10.497	35.864	55.673	108.000	41.405
	maxErr	0.000	0.010	0.024	0.081	0.003	0.011	0.034	0.065
	avgErr	0.000	0.005	0.015	0.043	0.000	0.005	0.023	0.042
80	BT	17	3	0	0	20	0	0	0
	Tbt	13.909	20.478	27.364	21.397	49.808	64.974	108.000	82.770
	maxErr	0.002	0.007	0.016	0.030	0.000	0.007	0.023	0.035
	avgErr	0.000	0.002	0.009	0.023	0.000	0.003	0.017	0.024

PC486/DX2 seconds; 20 instances for each entry.

Table IV. Geometric graphs.

		124				250			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	20	20	19	7	20	12	18	1
	Tbt	0.068	0.389	0.118	2.138	0.217	4.567	0.662	3.499
5	BT	20	16	12	0	20	0	6	0
	Tbt	0.481	1.513	0.838	1.579	3.187	6.476	5.393	2.017
10	BT	20	20	16	1	20	18	11	0
	Tbt	0.202	0.267	0.650	1.383	1.010	3.497	3.925	5.809
20	BT	20	20	19	7	20	20	12	2
	Tbt	0.102	0.112	0.396	0.827	0.442	0.928	3.642	2.366
40	BT	20	20	20	12	20	20	16	7
	Tbt	0.074	0.076	0.407	0.642	0.439	0.394	2.573	3.121
80	BT	20	20	19	11	20	20	16	9
	Tbt	0.056	0.113	0.607	1.104	0.327	0.369	2.577	4.358
		500				1000			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	20	0	18	0	20	0	19	0
	Tbt	1.249	21.465	2.418	7.186	5.055	71.617	19.611	25.977
5	BT	20	0	1	0	20	0	1	0
	Tbt	9.428	14.918	14.979	10.776	34.348	51.018	48.507	16.851
10	BT	20	5	9	0	19	1	3	0
	Tbt	5.251	19.898	12.397	4.252	24.923	63.163	53.886	16.065
20	BT	19	20	8	0	17	11	1	0
	Tbt	5.312	10.038	12.215	7.481	32.146	53.066	69.697	27.347
40	BT	20	20	9	6	19	20	3	1
	Tbt	3.115	1.813	19.442	12.836	8.577	17.238	75.806	47.784
80	BT	20	20	11	3	20	20	8	5
	Tbt	1.523	1.419	15.373	20.638	7.903	10.689	64.039	79.662

PC486/DX2 seconds; 20 instances for each entry.

Table V. Random graphs; class (α): range = [1,10^2].

		2000				4000			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	5	0	0	0	4	0	1	0
	Tbt	207.892	186.926	136.344	22.002	538.968	611.556	799.780	132.810
	maxErr	0.000	0.953	0.113	1.552	0.013	0.929	0.083	1.403
	avgErr	0.000	0.786	0.064	1.462	0.003	0.860	0.044	1.319
5	BT	5	0	0	0	5	0	0	0
	Tbt	221.672	200.874	321.666	89.530	770.864	690.854	973.592	207.876
	maxErr	0.000	0.115	0.056	0.350	0.000	0.153	0.055	0.307
	avgErr	0.000	0.088	0.028	0.297	0.000	0.107	0.043	0.273
10	BT	5	0	0	0	5	0	0	0
	Tbt	203.136	101.284	324.480	74.480	612.288	681.452	974.682	240.308
	maxErr	0.000	0.017	0.037	0.155	0.000	0.029	0.036	0.109
	avgErr	0.000	0.014	0.031	0.117	0.000	0.022	0.027	0.100
20	BT	4	1	0	0	3	2	0	0
	Tbt	169.238	164.396	324.488	130.460	569.712	630.952	972.850	371.418
	maxErr	0.001	0.009	0.032	0.062	0.001	0.008	0.028	0.067
	avgErr	0.000	0.005	0.026	0.052	0.000	0.002	0.024	0.060
40	BT	4	1	0	0	1	4	0	0
	Tbt	138.502	250.802	324.524	170.744	218.318	713.294	973.750	649.124
	maxErr	0.005	0.007	0.027	0.050	0.005	0.001	0.023	0.041
	avgErr	0.001	0.002	0.022	0.036	0.001	0.000	0.019	0.035
80	BT	5	0	0	0	4	1	0	0
	Tbt	95.932	164.320	324.434	314.580	169.650	588.552	973.846	972.154
	maxErr	0.000	0.004	0.020	0.024	0.001	0.005	0.016	0.037
	avgErr	0.000	0.002	0.016	0.022	0.000	0.002	0.014	0.034

PC486/DX2 seconds; 5 instances for each entry.

Table VI. Geometric graphs.

		2000				4000			
ν	index	*WEC-TS*	*M-KL*	*GRASP*	*TS*	*WEC-TS*	*M-KL*	*GRASP*	*TS*
2.5	BT	5	0	5	0	5	0	5	0
	Tbt	22.698	193.424	21.852	32.504	112.268	480.860	97.250	177.720
5	BT	5	0	0	0	4	0	1	0
	Tbt	53.026	128.820	176.882	39.162	400.504	702.864	495.854	154.648
10	BT	5	0	0	0	5	0	0	0
	Tbt	160.330	186.580	258.484	63.230	241.016	414.350	885.676	260.662
20	BT	4	1	0	0	5	0	0	0
	Tbt	83.128	280.658	275.222	122.866	449.588	711.564	974.584	504.878
40	BT	5	3	0	0	5	1	0	0
	Tbt	78.916	248.640	324.358	213.280	311.736	603.262	973.628	891.578
80	BT	5	5	0	0	5	4	0	0
	Tbt	26.530	32.770	324.534	324.004	584.896	417.194	922.780	972.330

PC486/DX2 seconds; 5 instances for each entry.

1. Maria Cristina Marcuzzo [1985] "Yoan Violet Robinson (1903-1983)", pp. 134

2. Sergio Lugaresi [1986] "Le imposte nelle teorie del sovrappiù", pp. 26

3. Massimo D'Angelillo e Leonardo Paggi [1986] "PCI e socialdemocrazie europee. Quale riformismo?", pp. 158

4. Gian Paolo Caselli e Gabriele Pastrello [1986] "Un suggerimento hobsoniano su terziario ed occupazione: il caso degli Stati Uniti 1960/1983", pp. 52

5. Paolo Bosi e Paolo Silvestri [1986] "La distribuzione per aree disciplinari dei fondi destinati ai Dipartimenti, Istituti e Centri dell'Università di Modena: una proposta di riforma", pp. 25

6. Marco Lippi [1986] "Aggregations and Dynamic in One-Equation Econometric Models", pp. 64

7. Paolo Silvestri [1986] "Le tasse scolastiche e universitarie nella Legge Finanziaria 1986", pp. 41

8. Mario Forni [1986] "Storie familiari e storie di proprietà. Itinerari sociali nell'agricoltura italiana del dopoguerra", pp. 165

9. Sergio Paba [1986] "Gruppi strategici e concentrazione nell'industria europea degli elettrodomestici bianchi", pp. 56

10. Nerio Naldi [1986] "L'efficienza marginale del capitale nel breve periodo", pp. 54

11. Fernando Vianello [1986] "Labour Theory of Value", pp. 31

12. Piero Ganugi [1986] "Risparmio forzato e politica monetaria negli economisti italiani tra le due guerre", pp. 40

13. Maria Cristina Marcuzzo e Annalisa Rosselli [1986] "The Theory of the Gold Standard and Ricardo's Standard Comodity", pp. 30

14. Giovanni Solinas [1986] "Mercati del lavoro locali e carriere di lavoro giovanili", pp. 66

15. Giovanni Bonifati [1986] "Saggio dell'interesse e domanda effettiva. Osservazioni sul cap. 17 della General Theory", pp. 42

16. Marina Murat [1986] "Betwin old and new classical macroeconomics: notes on Lejonhufvud's notion of full information equilibrium", pp. 20

17. Sebastiano Brusco e Giovanni Solinas [1986] "Mobilità occupazionale e disoccupazione in Emilia Romagna", pp. 48

18. Mario Forni [1986] "Aggregazione ed esogeneità", pp. 13

19. Sergio Lugaresi [1987] "Redistribuzione del reddito, consumi e occupazione", pp. 17

20. Fiorenzo Sperotto [1987] "L'immagine neopopulista di mercato debole nel primo dibattito sovietico sulla pianificazione", pp. 34

21. M. Cecilia Guerra [1987] "Benefici tributari nel regime misto per i dividendi proposto dalla commissione Sarcinelli: una nota critica", pp. 9

22. Leonardo Paggi [1987] "Contemporary Europe and Modern America: Theories of Modernity in Comparative Perspective", pp. 38

23. Fernando Vianello [1987] "A Critique of Professor Goodwin's 'Critique of Sraffa'", pp. 12

24. Fernando Vianello [1987] "Effective Demand and the Rate of Profits. Some Thoughts on Marx, Kalecki and Sraffa", pp. 41

25. Anna Maria Sala [1987] "Banche e territorio. Approccio ad un tema geografico-economico", pp. 40

26. Enzo Mingione e Giovanni Mottura [1987] "Fattori di trasformazione e nuovi profili sociali nell'agricoltura italiana: qualche elemento di discussione", pp. 36

27. Giovanna Procacci [1988] "The State and Social Control in Italy During the First World War", pp. 18

28. Massimo Matteuzzi e Annamaria Simonazzi [1988] "Il debito pubblico", pp. 62

29. Maria Cristina Marcuzzo (a cura di) [1988] "Richard F. Kahn. A discipline of Keynes", pp. 118

30. Paolo Bosi [1988] "MICROMOD. Un modello dell'economia italiana per la didattica della politica fiscale", pp. 34

31. Paolo Bosi [1988] "Indicatori della politica fiscale. Una rassegna e un confronto con l'aiuto di MICROMOD", pp. 25

32. Giovanna Procacci [1988] "Protesta popolare e agitazioni operaie in Italia 1915-1918", pp. 45

33. Margherita Russo [1988] "Distretto Industriale e servizi. Uno studio dei trasporti nella produzione e nella vendita delle piastrelle", pp. 157

34. Margherita Russo [1988] "The effect of technical change on skill requirements: an empirical analysis", pp. 28

35. Carlo Grillenzoni [1988] "Identification, estimations of multivariate transfer functions", pp. 33

36. Nerio Naldi [1988] "'Keynes' concept of capital", pp. 40

37. Andrea Ginzburg [1988] "locomotiva Italia?", pp. 30

38. Giovanni Mottura [1988] "La 'persistenza' secolare. Appunti su agricoltura contadina ed agricoltura familiare nelle società industriali", pp. 40

39. Giovanni Mottura [1988] "L'anticamera dell'esodo. I contadini italiani della 'restaurazione contrattuale' fascista alla riforma fondiaria", pp. 40

40. Leonardo Paggi [1988] "Americanismo e riformismo. La socialdemocrazia europea nell'economia mondiale aperta", pp. 120

41. Annamaria Simonazzi [1988] "Fenomeni di isteresi nella spiegazione degli alti tassi di interesse reale", pp. 44

42. Antonietta Bassetti [1989] "Analisi dell'andamento e della casualità della borsa valori", pp. 12

43. Giovanna Procacci [1989] "State coercion and worker solidarity in Italy (1915-1918): the moral and political content of social unrest", pp. 41

44. Carlo Alberto Magni [1989] "Reputazione e credibilità di una minaccia in un gioco bargaining", pp. 56

45. Giovanni Mottura [1989] "Agricoltura familiare e sistema agroalimentare in Italia", pp. 84

46. Mario Forni [1989] "Trend, Cycle and 'Fortuitous cancellation': a Note on a Paper by Nelson and Plosser", pp. 4

47. Paolo Bosi , Roberto Golinelli , Anna Stagni [1989] "Le origini del debito pubblico e il costo della stabilizzazione", pp. 26

48. Roberto Golinelli [1989] "Note sulla struttura e sull'impiego dei modelli macroeconometrici", pp. 21

49. Marco Lippi [1989] "A Shorte Note on Cointegration and Aggregation", pp. 11

50. Gian Paolo Caselli e Gabriele Pastrello [1989] "The Linkage between Tertiary and Industrial Sector in the Italian Economy: 1951-1988. From an External Dependence to an International One", pp. 40

51. Gabriele Pastrello [1989] "Francois quesnay: dal Tableau Zig-zag al Tableau Formule: una ricostruzione", pp. 48

52. Paolo Silvestri [1989] "Il bilancio dello stato", pp. 34

53. Tim Mason [1990] "Tre seminari di storia sociale contemporanea", pp. 26

54. Michele Lalla [1990] "The Aggregate Escape Rate Analysed throught the Queueing Model", pp. 23

55. Paolo Silvestri [1990] "Sull'autonomia finanziaria dell'università", pp. 11

56. Paola Bertolini, Enrico Giovannetti [1990] "Uno studio di 'filiera' nell'agroindustria. Il caso del Parmigiano Reggiano", pp. 164

57. Paolo Bosi, Roberto Golinelli, Anna Stagni [1990] "Effetti macroeconomici, settoriali e distributivi dell'armonizzazione dell'IVA", pp. 24

58. Michele Lalla [1990] "Modelling Employment Spells from Emilia Labour Force Data", pp. 18

59. Andrea Ginzburg [1990] "Politica Nazionale e commercio internazionale", pp. 22

60. Andrea Giommi [1990] "La probabilità individuale di risposta nel trattamento dei dati mancanti", pp. 13

61. Gian Paolo Caselli e Gabriele Pastrello [1990] "The service sector in planned economies. Past experiences and future prospectives", pp. 32

62. Giovanni Solinas [1990] "Competenze, grandi industrie e distretti industriali,. Il caso Magneti Marelli", pp. 23

63. Andrea Ginzburg [1990] "Debito pubblico, teorie monetarie e tradizione civica nell'Inghilterra del Settecento", pp. 30

64. Mario Forni [1990] "Incertezza, informazione e mercati assicurativi: una rassegna", pp. 37

65. Mario Forni [1990] "Misspecification in Dynamic Models", pp. 19

66. Gian Paolo Caselli e Gabriele Pastrello [1990] "Service Sector Growth in CPE's: An Unsolved Dilemma", pp. 28

67. Paola Bertolini [1990] "La situazione agro-alimentare nei paesi ad economia avanzata", pp. 20

68. Paola Bertolini [1990] "Sistema agro-alimentare in Emilia Romagna ed occupazione", pp. 65

69. Enrico Giovannetti [1990] "Efficienza ed innovazione: il modello "fondi e flussi" applicato ad una filiera agro-industriale", pp. 38

70. Margherita Russo [1990] "Cambiamento tecnico e distretto industriale: una verifica empirica", pp. 115

71. Margherita Russo [1990] "Distretti industriali in teoria e in pratica: una raccolta di saggi", pp. 119

72. Paolo Silvestri [1990] " La Legge Finanziaria. Voce dell'enciclopedia Europea Garzanti", pp. 8

73. Rita Paltrinieri [1990] "La popolazione italiana: problemi di oggi e di domani", pp. 57

74. Enrico Giovannetti [1990] "Illusioni ottiche negli andamenti delle Grandezze distributive: la scala mobile e l''appiattimento' delle retribuzioni in una ricerca", pp. 120

75. Enrico Giovannetti [1990] "Crisi e mercato del lavoro in un distretto industriale: il bacino delle ceramiche. Sez I", pp. 150

76. Enrico Giovannetti [1990] " Crisi e mercato del lavoro in un distretto industriale: il bacino delle ceramiche. Sez. II", pp. 145

78. Antonietta Bassetti e Costanza Torricelli [1990] "Una riqualificazione dell'approccio bargaining alla selezioni di portafoglio", pp. 4

77. Antonietta Bassetti e Costanza Torricelli [1990] "Il portafoglio ottimo come soluzione di un gioco bargaining", pp. 15

79. Mario Forni [1990] "Una nota sull'errore di aggregazione", pp. 6

80. Francesca Bergamini [1991] "Alcune considerazioni sulle soluzioni di un gioco bargaining", pp. 21

81. Michele Grillo e Michele Polo [1991] "Political Exchange and the allocation of surplus: a Model of Two-party competition", pp. 34

82. Gian Paolo Caselli e Gabriele Pastrello [1991] "The 1990 Polish Recession: a Case of Truncated Multiplier Process", pp. 26

83. Gian Paolo Caselli e Gabriele Pastrello [1991] "Polish firms: Pricate Vices Pubblis Virtues", pp. 20

84. Sebastiano Brusco e Sergio Paba [1991] "Connessioni, competenze e capacità concorrenziale nell'industria della Sardegna", pp. 25

85. Claudio Grimaldi, Rony Hamaui, Nicola Rossi [1991] "Non Marketable assets and hauseholds' Portfolio Choice: a Case of Study of Italy", pp. 38

86. Giulio Righi, Massimo Baldini, Alessandra Brambilla [1991] "Le misure degli effetti redistributivi delle imposte indirette: confronto tra modelli alternativi", pp. 47

87. Roberto Fanfani, Luca Lanini [1991] "Innovazione e servizi nello sviluppo della meccanizzazione agricola in Italia", pp. 35

88. Antonella Caiumi e Roberto Golinelli [1992] "Stima e applicazioni di un sistema di domanda Almost Ideal per l'economia italiana", pp. 34

89. Maria Cristina Marcuzzo [1992] "La relazione salari-occupazione tra rigidità reali e rigidità nominali", pp. 30

90. Mario Biagioli [1992] "Employee financial participation in enterprise results in Italy", pp. 50

91. Mario Biagioli [1992] "Wage structure, relative prices and international competitiveness", pp. 50

92. Paolo Silvestri e Giovanni Solinas [1993] "Abbandoni, esiti e carriera scolastica. Uno studio sugli studenti iscritti alla Facoltà di Economia e Commercio dell'Università di Modena nell'anno accademico 1990/1991", pp. 30

93. Gian Paolo Caselli e Luca Martinelli [1993] "Italian GPN growth 1890-1992: a unit root or segmented trend representatin?", pp. 30

94. Angela Politi [1993] "La rivoluzione fraintesa. I partigiani emiliani tra liberazione e guerra fredda, 1945-1955", pp. 55

95. Alberto Rinaldi [1993] "Lo sviluppo dell'industria metalmeccanica in provincia di Modena: 1945-1990", pp. 70

96. Paolo Emilio Mistrulli [1993] "Debito pubblico, intermediari finanziari e tassi d'interesse: il caso italiano", pp. 30

97. Barbara Pistoresi [1993] "Modelling disaggregate and aggregate labour demand equations. Cointegration analysis of a labour demand function for the Main Sectors of the Italian Economy: 1950-1990", pp. 45

98. Giovanni Bonifati [1993] "Progresso tecnico e accumulazione di conoscenza nella teoria neoclassica della crescita endogena. Una analisi critica del modello di Romer", pp. 50

99. Marcello D'Amato e Barbara Pistoresi [1994] "The relationship(s) among Wages, Prices, Unemployment and Productivity in Italy", pp. 30

100. Mario Forni [1994] "Consumption Volatility and Income Persistence in the Permanent Income Model", pp. 30

101. Barbara Pistoresi [1994] "Using a VECM to characterise the relative importance of permanent and transitority components", pp. 28

102. Gian Paolo Caselli and Gabriele Pastrello [1994] "Polish recovery form the slump to an old dilemma", pp. 20

103. Sergio Paba [1994] "Imprese visibili, accesso al mercato e organizzazione della produzione", pp. 20

104. Giovanni Bonifati [1994] "Progresso tecnico, investimenti e capacità produttiva", pp. 30

105. Giuseppe Marotta [1994] "Credit view and trade credit: evidence from Italy", pp. 20

106. Margherita Russo [1994] "Unit of investigation for local economic development policies", pp. 25

107. Luigi Brighi [1995] "Monotonicity and the demand theory of the weak axioms", pp. 20

108. Mario Forni e Lucrezia Reichlin [1995] "Modelling the impact of technological change across sectors and over time in manufactoring", pp. 25

109. Marcello D'Amato and Barbara Pistoresi [1995] "Modelling wage growth dynamics in Italy: 1960-1990", pp. 38

110. Massimo Baldini [1995] "INDIMOD. Un modello di microsimulazione per lo studio delle imposte indirette", pp. 37

111. Paolo Bosi [1995] "Regionalismo fiscale e autonomia tributaria: l'emersione di un modello di consenso", pp. 38

112. Massimo Baldini [1995] "Aggregation Factors and Aggregation Bias in Consumer Demand", pp. 33

113. Costanza Torricelli [1995] "The information in the term structure of interest rates. Can stocastic models help in resolving the puzzle?" pp. 25

114. Margherita Russo [1995] "Industrial complex, pôle de développement, distretto industriale. Alcune questioni sulle unità di indagine nell'analisi dello sviluppo." pp. 45

115. Angelika Moryson [1995] "50 Jahre Deutschland. 1945 - 1995" pp. 21

116. Paolo Bosi [1995] "Un punto di vista macroeconomico sulle caratteristiche di lungo periodo del nuovo sistema pensionistico italiano." pp. 32

117. Gian Paolo Caselli e Salvatore Curatolo [1995] "Esistono relazioni stimabili fra dimensione ed efficienza delle istituzioni e crescita produttiva? Un esercizio nello spirito di D.C. North." pp. 11

118. Mario Forni e Marco Lippi [1995] "Permanent income, heterogeneity and the error correction mechanism." pp. 21

119. Barbara Pistoresi [1995] "Co-movements and convergence in international output. A Dynamic Principal Components Analysis" pp. 14

120. Mario Forni e Lucrezia Reichlin [1995] "Dynamic common factors in large cross-section" pp. 17

121. Giuseppe Marotta [1995] "Il credito commerciale in Italia: una nota su alcuni aspetti strutturali e sulle implicazioni di politica monetaria" pp. 20

122. Giovanni Bonifati [1995] "Progresso tecnico, concorrenza e decisioni di investimento: una analisi delle determinanti di lungo periodo degli investimenti" pp. 25

123. Giovanni Bonifati [1995] "Cambiamento tecnico e crescita endogena: una valutazione critica delle ipotesi del modello di Romer" pp. 21

124. Barbara Pistoresi e Marcello D'Amato [1995] "La riservatezza del banchiere centrale è un bene o un male? ,Effetti dell'informazione incompleta sul benessere in un modello di politica monetaria." pp. 32

125. Barbara Pistoresi [1995] "Radici unitarie e persistenza: l'analisi univariata delle fluttuazioni economiche." pp. 33

126. Barbara Pistoresi e Marcello D'Amato [1995] "Co-movements in European real outputs" pp. 20

127. Antonio Ribba [1996] "Ciclo economico, modello lineare-stocastico, forma dello spettro delle variabili macroeconomiche" pp. 31

128. Carlo Alberto Magni [1996] "Repeatable and una tantum real options a dynamic programming approach" pp. 23

129. Carlo Alberto Magni [1996] "Opzioni reali d'investimento e interazione competitiva: programmazione dinamica stocastica in optimal stopping" pp. 26

130. Carlo Alberto Magni [1996] "Vaghezza e logica fuzzy nella valutazione di un'opzione reale" pp. 20

131. Giuseppe Marotta [1996] "Does trade credit redistribution thwart monetary policy?Evidence from Italy" pp. 20