Multilevel Reactive Tabu Search for Graph Partitioning

Roberto Battiti, Alan Bertossi and Anna Cappelletti

Dipartimento di Matematica
Università di Trento
Via Sommarive 14
38050 Trento, Italy

E-mail:{battiti, bertossi}@science.unitn.it

FAX: +39 0461 88 1624

Abstract

In this paper, new multilevel heuristics are proposed for finding a balanced bipartition of a graph. Multilevel schemes have been recently introduced for quickly solving the problem on graphs with thousands of nodes, graphs often present in domain-decomposition techniques for parallel computing. These heuristics firstly reduce the original graph by collapsing its nodes and arcs, thus generating a hierarchy of smaller and smaller graphs, then find a good partition of the smallest graph, and finally project and refine the so found partition backwards on the graph hierarchy. The multilevel algorithms proposed in this paper employ a novel greedy method and tabu search techniques during the partitioning and uncoarsening phases. Experimental results on a wide collection of benchmark graphs show that partitions of high quality are found in small computing times. Moreover, the effect of various parameter choices during the partitioning phase is experimentally studied.

Keywords: *graph partitioning, graph bisection, multilevel algorithm, greedy, tabu search.*

1 Introduction

The partitioning problem on an undirected graph $G = (V, E)$, V being the set of *vertices* and E the set of *edges* arises from several applications in parallel computing, such as task scheduling, sparse matrices factorizations [20], and numerical simulations on parallel computers [16, 18, 26]. In general, the graph partitioning problem consists of partitioning the vertices into k disjoint subsets of about the same cardinality, such that the *cut size*, that is, the number of edges whose endpoints are in different subsets, is minimized. In this paper, the particular balanced bipartitioning case is considered (also called *graph bisection* problem), where k is equal to two and the difference of cardinalities between the two subsets of the partition is at most one.

The graph partitioning problem is NP-hard, and it remains NP-hard even when k is equal to 2 or when some unbalancing is allowed [5], as long as an upper bound on the largest vertex subset is required. For large graphs (with more than about 100 vertices), heuristic algorithms which find suboptimal solutions are the only viable option. In the last years, very efficient multilevel schemes [13, 16] have been proposed. These heuristics reduce the original graph by collapsing its nodes and arcs thus generating a hierarchy of smaller and smaller graphs (coarsening phase), then find a good partition of the smallest graph (partitioning phase), and finally project and refine the so found partition backwards on the graph hierarchy (uncoarsening phase) until a partition of the original graph is found.

Multilevel schemes have been the starting point for the current paper. The novel ingredients are new greedy and tabu search heuristics used during the partitioning and uncoarsening phases. The Differential-Greedy algorithm in [3] is very efficient and competitive for partitioning randomly generated graphs of reasonable sizes, but its CPU times become excessive for very large graphs. We were therefore motivated to consider its use in state-of-the-art multilevel schemes. Moreover, the more effective prohibition-based procedures proposed in [2] can be employed during the uncoarsening phase to refine the solutions instead of the widely used Kernighan-Lin (KL) algorithm [16, 13].

A straightforward greedy scheme consists of randomly choosing two *seed* vertices, inserting them into the two subsets V_0 and V_1 of the partition, and then alternately adding to V_0 and V_1 a vertex which causes the minimum increase of the size of the cut f. The Differential-Greedy algorithm [3] is obtained by modifying the selection criterion in order to pick vertices that minimize the *difference* between new edges across the cut and new edges that become internal after the addition. In [3], an efficient implementation of Differential-Greedy is given which is based on a *bucket* data structure, and several experiments are performed demonstrating that the method outperforms all previously proposed greedy schemes, when both solution quality and running times are considered.

The second difference introduced into the multilevel algorithms proposed in this paper is the Tabu Search (TS) heuristic used in the uncoarsening phase. Tabu Search [9, 11] is based on prohibition techniques and "intelligent" schemes, which complement the basic local search with the purpose to continue the search beyond a local optimizer, while actively using the past history of the search to better focus the future explorations. In order to apply TS to a multilevel scheme, two alternative choices have been considered:

- to employ a basic TS routine using a prohibition parameter T (determining how much time a move remains forbidden after the execution of its inverse move) whose value is maintained fixed for the whole search (Fixed-TS or FTS).
- to employ a TS routine based on a randomized and reactive choice of the parameter T, whose value is determined automatically so that it can change during the search depending on the past history and on the local properties of the configuration space of a specific task (Reactive-Randomized-Tabu-Search or RRTS).

To keep the size of this paper limited and to avoid duplications we refer to the cited bibliography (that is also available from WWW in preprint form) for all details about the used algorithms Differential-Greedy [3], FTS and RRTS [2].

The rest of this paper is structured as follows. Section 2 introduces the notation and the formulation of the graph bisection problem. Section 3 illustrates the benchmark graphs considered in this paper. Section 4 is devoted to the new multilevel algorithms for graph bisection and Section 5 presents the experimental results obtained by applying the new multilevel heuristics on the benchmark graphs previously introduced.

2 Definitions and problem formulation

Given an undirected graph $G = (V, E)$, the function

Definition 2.1

$$\pi : V \rightarrow \{0, 1, ..., k-1\}$$

is a partition *of G which distributes the vertices among k distinct subsets* $V_1, V_2, ..., V_k$*. Thus a* k-partitioning $P^k = \{V_0, V_1, ..., V_{k-1}\}$ *consists of k subsets* V_0, V_1, ..., V_{k-1} *such that* $V_0 \cup V_1 \cup ... \cup V_{k-1} = V$ *and* $V_0 \cap V_1 \cap ... \cap V_{k-1} = \emptyset$*. When* $k = 2$*,* P^2 *is said to be a* bipartition*.*

In this paper, we restrict our attention to the problem of finding a bipartition P^2 of a graph. A bipartition is characterized by its *cut*, which is defined as follows.

Definition 2.2 *The set of edges cut by a solution* P^2 *is given by* $E(P^2) = \{e \in E,\ \mathrm{e} = (u, v) \mid u \in V_0,\ v \in V_1\}$*. Thus the edge* e $\in E(P^2)$ *when their two endpoints do not belong to the same subset of the bipartition. The quantity* $f = |E(P^2)|$ *is called* cut size *(or simply* cut*) of* P^2*.*

The problem of graph *bisection* (*or balanced bipartitioning*) *with minimum cut* asks for a bipartition whose two subsets have about the same size such that the number of cut edges is minimum.

Graph bisection (or balanced bipartition) with minimum cut:
Find P^2 such that $||V_0| - |V_1|| \leq 1$ so as to minimize f.

3 Benchmark Graphs

The *performance* of the new multilevel heuristics presented in this paper is evaluated on a collection of (unweighted) graphs, whose characteristics are illustrated in Table 1.
The benchmark collection consists of the following 26 graphs:

- seven 2D finite elements meshes (*grid2, airfoil1, 3elt, ukerbe1, whitaker3, crack* and *big*);
- their correspondent dual graphs (*grid2_dual, airfoil1_dual, 3elt_dual, ukerbe1_dual, whitaker3_dual, crack_dual* and *big_dual*) ;
- five De Bruijn networks (*DEBR12, DEBR13, DEBR14, DEBR15, DEBR16*);
- two sparse symmetric matrices from the $Harwell - Boeing$ collection (*bsspwr09, bcsstk13*) [7] plus another sparse matrix from the Nasa collection ($nasa4704$);
- four random graphs ($G1000.01, G1000.02, U1000.20, U1000.40$).

The finite elements meshes and the Harwell-Boeing graphs have been widely used for evaluating and comparing the performance of several partitioning methods. The De Bruijn networks represent interconnection networks used in parallel computers, and are deeply studied because they possess some properties of interest. Indeed, they are regular graphs with small diameter, and are suited for FFT applications. The $bsspwr09$ and $bcsstk13$ matrices are used to model power networks and fluido-dynamics problems,

graph name	number of vertices	number of edges	vertex degree		
			min	average	max
grid2	3296	6432	2	3.90	5
airfoil1	4253	12289	3	5.78	9
3elt	4720	13722	3	5.81	9
ukerbe1	5981	7852	2	2.63	8
whitaker3	9800	28989	3	5.92	8
crack	10240	30380	3	5.93	9
big	15606	45878	3	5.88	10
grid2-dual	3136	6112	2	3.90	4
airfoil1-dual	8034	11813	2	2.94	3
3elt-dual	9000	13278	2	2.95	3
ukerbe1-dual	1866	3538	2	3.79	4
whitaker3-dual	19190	28581	2	2.98	3
crack-dual	20141	30043	2	2.98	3
big-dual	30265	44929	2	2.97	3
DEBR12	4096	8189	2	4.00	4
DEBR13	8192	16381	2	4.00	4
DEBR14	16384	32765	2	4.00	4
DEBR15	32768	65533	2	4.00	4
DEBR16	65536	131069	2	4.00	4
bcspwr09	1723	2394	4	40.88	94
bcsstk13	2003	40940	1	2.78	14
nasa4704	4704	50026	5	21.27	41
G1000.01	1000	5064	2	10.13	23
G1000.02	1000	10107	6	20.21	34
U1000.20	1000	9339	4	18.68	39
U1000.40	1000	18015	9	36.03	58

Table 1: The benchmark graphs used to evaluate the performance of the graph partitioning algorithms.

respectively. The random graphs are of two kinds: $Gn.d$ e $Un.d$. The $Gn.d$ graphs have n vertices, with an edge between two vertices with probability p, so that the resulting average vertex degree is $10 * d$ (thus $p(n-1) = 10 * d$ holds). The $Un.d$ graphs are geometric random graphs with n vertices, uniformly distributed within a unit square, with two vertices connected by an edge whenever their Euclidean distance is less than or equal to t, where $d = n\pi t^2$ is the average vertex degree. All graphs have been obtained from the collection in [6].

4 Multilevel Schemes for Graph Bisection

A multilevel algorithm consists of three different phases. First, starting from the original graph $G_0 = (V_0, E_0)$, a sequence of smaller and smaller graphs is derived, until a graph with few dozens of vertices is obtained. Second, a bisection is found for the smallest graph. Third, the bisection so found is projected and refined backwards until a bisection for the original graph is found (see Fig. 1). Because larger graphs have more degrees of freedom, backward refining usually decreases the cut size.

More formally, a multilevel algorithm acts as follows.
Coarsening Phase

The original graph G_0 is transformed into a sequence of graphs $G_1, G_2, ..., G_m$ such that $|V_0| > |V_1| > |V_2| > ... > |V_m|$.
Partitioning Phase

A bisection P_m is found for the last graph $G_m = (V_m, E_m)$.
Uncoarsening Phase

P_m is projected backwards through the successive bisections P_{m-1},P_{m-2},...,P_1, P_0 and the bisection P_0 is output for the original graph G_0.

Because of the reduced size, it is faster to find a good bisection for the smallest graph G_m than for the original graph G_0. The reduction in complexity is paid in terms of a loss in granularity, because only a small number of bisections of G_0 can be represented in G_m. However, the refinements performed on levels $m-1, m-2, ..., 1, 0$ during the uncoarsening phase usually improve the solution by a sizable factor.

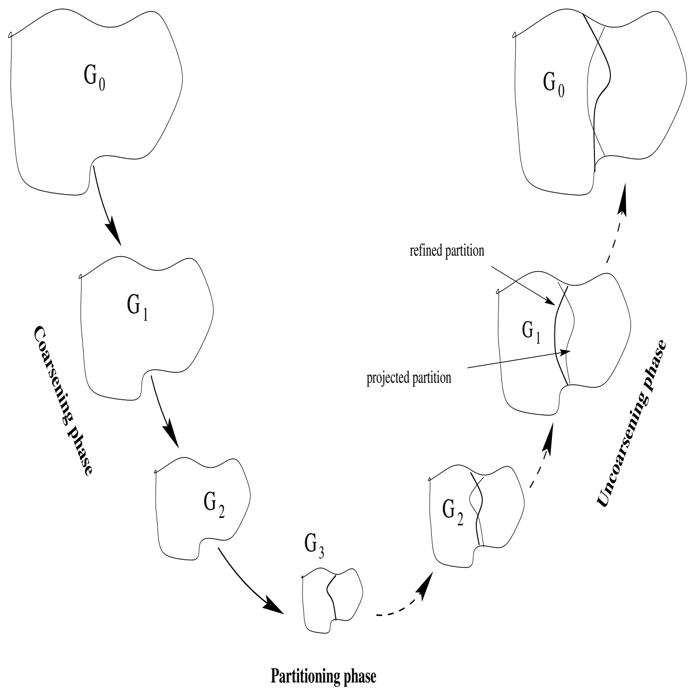


Figure 1: The three phases of a multilevel algorithm for graph bisection.

4.1 The Coarsening Phase

In the multilevel algorithms considered in this paper, the coarsening phase has been realized by means of the following three existing heuristics for finding a maximal weighted matching:

- **Heavy Edge Matching (HEM):** proposed by Karypis and Kumar in [16]. The vertices are considered following a random order. If a vertex u is *unmatched*, then another *unmatched* vertex v is chosen, if any, such that the weight of the edge

(u, v) is maximum among all the edges incident to u. If this node exists, the edge (u, v) is added to the matching.

- **Modified Heavy Edge Matching (HEM*):** a variant of the previous heuristic, with the purpose of decreasing the degree of the reduced graph [16]. Indeed, it has been observed in [18] that a good bisection for the reduced graph is nearer to a good bisection for the original graph whenever the degree of the reduced graph is small. The vertices are again considered in a random order. Let v be an unmatched vertex and H_v be the set of unmatched vertices connected to v by means of an edge of maximum weight (namely, H_v may include several vertices whenever there are several edges with the same weight incident in v). For each $u \in H_v$, let W_{v-u} be the sum of the weights of the edges connecting u to a vertex adjacent to v. Then v is matched with the vertex $u \in H_v$ for which W_{v-u} is maximum among all the vertices in H_v.

- **Heaviest Edge Matching (HEAV):** proposed by Gupta [10], considers the edges in order of decreasing weights (with ties broken randomly). It behaves better than HEM and HEM* in the last coarsening steps, when the size of the graph has been considerably reduced and thus the sorting on the weights is less computationally costly. Moreover, it is more efficient for graphs with unbalanced edge weights, which happens also after some coarsening steps. Therefore, the best strategy usually consists of applying one heuristic (e.g. HEM) during the first levels and then switch to the HEAV heuristic during the last levels.

4.2 The Partitioning Phase

A partition of G_m has been found in the literature by means of different techniques, like spectral bisection [1, 12, 23, 24] or geometric bisection heuristics [21]. Because the size of G_m is small (usually $|V_m| < 100$), this phase is fast. In this paper, we consider two algorithms for finding a bisection of G_m which are both based on the use of a greedy routine (DIFFERENTIAL-GREEDY) followed by: i) a Tabu Search heuristic with fixed prohibition (algorithm MULTILEVEL-FTS), or ii) a randomized and reactive Tabu Search scheme with a dynamic and history-dependent prohibition parameter (MULTILEVEL-RRTS).

MULTILEVEL-FTS with the right prohibition obtains excellent solutions in very small computing times. However, a suitable value of the prohibition has to be found through a preliminary tuning phase. As it will be shown in Section 5.1, the right choice can be determined *off-line* by studying the curves of the cut size found by the algorithm as a function of the possible prohibitions. In contrast, with MULTILEVEL-RRTS, the choice of the prohibition is no more a task of the programmer, but it is determined automatically by means of a SCORING routine. Because of the processing time required by the SCORING routine MULTILEVEL-RRTS is slower than MULTILEVEL-FTS (of course, if one does not consider the preliminary "hand-made" tuning required by MULTILEVEL-FTS).

4.3 The Uncoarsening Phase

In order to design an effective multilevel algorithm, a fast and effective local refinement scheme is needed during the uncoarsening phase. The KL algorithm is satisfactory to find locally optimal solutions but, unless it is initialized with a good global partition, the resulting local optima can be far from the global optimum. We refine the solutions during the uncoarsening phase by means of Fixed-TS, a Tabu Search routine related to KL [2]. The main differences are that the KL concept of *chain of tentative exchanges* disappears because the current solution is changed immediately after each move is selected, and that worsening moves are also accepted (termination is only after a selected number of iterations are executed). Given a solution $X^{(t)}$ at iteration t, Fixed-TS tries to improve the solution by searching within the set $N_A(X^{(t)})$ of solutions close to $X^{(t)}$ (solutions where a single node is exchanged between the two sets of the current partition) given by:

$$N_A(X^{(t)}) = \{X = \mu \circ X^{(t)} \textit{ such that } \textsc{LastUsed}(\mu^{-1}) < (t - T)\} \qquad (1)$$

where LastUsed(μ) denotes when the move μ has been used for the last time (LastUsed(μ) $= -\infty$ at the beginning). In other words, Fixed-TS only considers the solutions that can be obtained from the current solution by applying a move whose inverse move has not been used during the last T iterations. Additional details about Tabu Search are explained in [2] and the final algorithm is illustrated in Fig. 2.

```
Multilevel-FTS

        $G_0 \leftarrow G$-filegraph;
        $level \leftarrow 0$;
\\ Coarsening Phase:
        while $((level = 0) \,||\, (n_{G_{level}} > 40$ &&
                $n_{G_{level}} < 0.85 * n_{G_{\{level-1\}}}$ &&
                $n_{G_{level}} > \frac{n_{G_{level}}}{2}))$ do
        begin
            $G_{\{level+1\}} \Leftarrow$ Matching ( $G_{level}$, $matching\ method$ );
            $level \leftarrow level + 1$
        end
\\ Partitioning Phase:
        $G_{level} \Leftarrow$ Diff-Greedy;
        $G_{level} \Leftarrow$ Fixed-TS($T_f$, $10 * n_{G_{level}}$);
\\ Uncoarsening Phase:
        for $j \leftarrow level$ downto 1 do
        begin
            $G_{\{j-1\}} \Leftarrow$ save-solution( $G_j$ );
            if $(j < 5)$ then $G_{\{j-1\}} \Leftarrow$ Fixed-TS( $T_f$, $3 * n_{G_{\{j-1\}}}$ );
            else $G_{\{j-1\}} \Leftarrow$ Fixed-TS( $T_f$, $8 * n_{G_{\{j-1\}}}$ );
        end
```

Figure 2: The Multilevel-FTS algorithm.

5 Experimental Results

5.1 Effect of the fixed prohibition on the performance

In order to determine the effect of the fixed prohibition on the *performance*, the solutions found by applying the MULTILEVEL-FTS algorithm described in Fig. 2, are studied as functions of the prohibition parameter T. Some representative results are collected in Fig. 3-6, where the cut size is shown as a function of the *fractional prohibition* T_f. The parameter T_f is defined such that $T = \lfloor T_f n \rfloor$, n being the number of vertices. The results are averages over 100 independent executions (with different seeds for the random number generator). Only values for $T_f \in \,]0, 1/4]$ have been considered. Indeed, because of the problem structure, any 0-1 vertex assignment and its complement (with 0 being substituted by 1 and vice-versa) both denote the same solution, with the two sets of the bisection interchanged. Therefore a value of $T_f > 1/2$ would imply that, after starting from an assignment, one would reach after $(T+1)$ iterations an assignment closer to the complement than to the original one. The chosen threshold of 1/4 completely eliminates these oscillations between an assignment and its complement.

From the experimental results one observes that the prohibition T does indeed have a crucial effect on the performance. After considering the heuristically optimal cut for the graphs in Fig. 3-6 (denoted by the straight line at the bottom), one observes that low average cuts are obtained with the proper prohibition value, while very poor cuts can be obtained if the T value is not appropriate. By analyzing the results, one can determine the optimal T_f, depending on the graph characteristics, such as their number of vertices, number of edges, and average vertex degree. For examples, the curves for the finite element meshes and De Bruijn networks share a similar behavior: first, for $T_f < 0.05$, the average cut rapidly decreases, reaching very good average cut sizes; a minimum is then reached for values of T_f near to 0.05; finally, the cut size gradually increases. A qualitatively similar behavior is present in the results for the other graphs.

This common behavior can be explained as follows. As soon as a local minimizer is reached along the search trajectory, all configurations within the *attraction basin* associated to the given minimizer (the points that are mapped to the given point by the local search dynamics) are not of interest for the optimization. In fact the cut values are larger than or equal to the value at the local minimizer, by the definition of attraction basin. Therefore the search algorithm should avoid wasting CPU time in the given basin by activating some *diversification* scheme. Ideally, diversification should bring the search trajectory sufficiently far from the local minimizer so that a new attraction basin, possibly with a lower cut value, can be reached by a descending local search trajectory. Tabu Search achieves diversifications through the prohibition mechanism [4]: the larger T, the larger the Hamming distance that must be reached after starting from a given configuration before possibly coming back closer to the starting configuration. If T is too small, then the search trajectory can visit again an already visited point just after few iterations, and thus the probability to escape from a local minimum is small. On the contrary, if T is too big, after an initial phase only few moves are allowed, and the degree of freedom in choosing moves is limited. Of course, the appropriate T is related to the typical size of the attraction basins around the local minimizers. Because the size is not known *a priori*, either the value has to be determined from *off-line* experiments, or through some simple automated tuning (i.e., *learning*) scheme.

If one considers the matching methods used in the coarsening phase, one observes that the curves shown in the figures present the same behavior for all the three matching heuristics (HEM, HEM*, and HEAV).

From the obtained results (Fig. 3 - Fig. 6) one can derive some conclusions. For all the finite elements meshes, the values of T_f leading to the best results are always less than 0.1. In fact, most of these values fall within the interval [0.03, 0.07]. Thus in general it is convenient to use either $T_f = 0.05$ or $T_f = 0.06$, because these prohibitions always lead to very good bisections. For the correspondent dual graphs, the fractional prohibition giving the best results is slightly larger: in these cases it is convenient to choose $T_f \in [0.05, 0.08]$. Instead, for the De Bruijn networks and random graphs the optimal T_f is 0.07. Finally, the best cuts for the matrices are obtained with even larger values of T_f (but always smaller than 0.15).

5.2 Multilevel-FTS

Based on the results described in the previous section, we applied the Multilevel-FTS algorithm using a fractional prohibition such that: $T_f = 0.05$ for the finite elements meshes, $T_f = 0.07$ for their corresponding dual graphs, as well as for the random and De Brujin graphs, and $T_f = 0.1$ for the sparse matrices. The statistics of 100 executions of Multilevel-FTS are reported in Fig. 7.

Each execution is characterized by a different random value (let us recall that the greedy routine is based on an initial random choice and thus different bisections can result depending on the initial choice). In Fig. 7 one reports, for each matching heuristic, the minimum and average cut of 100 independent executions of Multilevel-FTS. Moreover, the best min cut over the three matching heuristics is shown in yellow (gray for b/w printers), while the best average cut is shown in green (dark gray for b/w printers) . The computing times in seconds are the average CPU times for each single execution on a SUNW Ultra-2 Sparc Sun4u with SunOS Release 5.5.1 operating system. Finally, Fig. 7 reports the best cuts known up to now (on the *Best* column), obtained from Monien and Diekmann [6]. One observes a considerable improvement on the cut size for the finite elements meshes, their corresponding dual graphs, and the random graphs: in eleven cases the *Best* value is improved. On the contrary, the *Best* value is never achieved for the De Bruijn graphs. It is worth noting that for *crack-dual* and all the three matching heuristics even the average cut remains below the best cut known until now. Finally, one can observe that there is no matching heuristic which gives significantly better cuts with respect to the other two and that the computing times are comparable for all the three matching heuristics.

5.3 Randomized and Reactive Prohibition: Multilevel-RRTS

Consider now the Multilevel-RRTS algorithm, which uses a randomized and reactive (i.e., based on feedback) choice of the prohibition. In this way, the choice of the right prohibition is done by the algorithm itself via a “scoring” routine, and no tuning is needed by the programmer. The Multilevel-RRTS algorithm can be obtained from that of Multilevel-FTS simply by changing the way in which the bisection of the smallest graph is computed, namely, by replacing lines 10-11 of Fig. 2 with the following line:

$$G_{level} \Leftarrow \mathrm{RRTS}(\mathit{iterations},\ \mathit{individual});$$

In other words, the bisection of G_{level} is now computed by means of the RRTS (REACTIVE-RANDOMIZED-TABU-SEARCH) routine. All the details of this routine can be found in [2].

Fig. 8 reports the results of 100 executions of MULTILEVEL-RRTS where, in the partitioning phase, RRTS is executed with parameters $\mathit{iterations} = 100 * n_{G_{level}}$ and $\mathit{individual} = 10 * n_{G_{level}}$. As before, Fig. 8 reports the min cut over100 executions, the average cut, and the average CPU times for a single execution. By observing the figure, the same considerations of the previous section can be confirmed. Again, no matching heuristic behaves better than the other two heuristics.

Conclusions

In this paper two new multilevel algorithms for graph bisection have been presented, which differ from the previous multilevel algorithms because of the novel heuristics used during the partitioning and uncoarsening phases. In particular, the DIFF-GREEDY technique previously proposed by the authors is used to generate a first solution on the coarsened graphs, while two versions of a prohibition-based local search (Tabu Search) are used in the refinement steps, leading to the MULTILEVEL-FTS and MULTILEVEL-RRTS algorithms. While the prohibition parameter T is fixed in the first case (and therefore it must selected by a preliminary off-line tuning phase for the different graph kinds), it is dynamic and determined in an automated and on-line way in the second case. In this way the explicit tuning by the user is avoided and the T value can change during the search depending on the properties of a specific task. The new algorithms were successfully tested on a collection of benchmark graphs arising from various applications in the area of parallel computing, leading to improvements in the best known cut values for many graphs, especially for 2D finite elements meshes.

References

[1] S.T. Barnard and H.D. Simon, "A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems," *Proc. 6th SIAM Conf. on Parallel Processing for Scientific Computing*, pp. 711-718, 1995.

[2] R. Battiti and A. Bertossi, "Greedy, Prohibition, and Reactive Heuristics for Graph Partitioning," *IEEE Transactions on Computers*, in press, 1999. Preprint version available at:
`http://rtm.science.unitn.it/~battiti/archive/gp.ps.gz`.

[3] R. Battiti and A. Bertossi, "Differential Greedy for the 0-1 Equicut Problem." In: *Network Design: Connectivity and Facilities Location.* D.Z. Du and P.M. Pardalos (Eds.). DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Volume 40, 1998, pp. 3–21. Preprint version available at:
`http://rtm.science.unitn.it/~battiti/archive/gp-dimacs.ps.gz`.

[4] R. Battiti, "Reactive Search: Toward self-tuning heuristics." In: *Modern Heuristic Search Methods*, chapter 4, V.J. Rayward-Smith (Ed.), pp. 61-83, John Wiley and Sons Ltd, 1996. Preprint version available at:
`http://rtm.science.unitn.it/~battiti/archive/adt-book-chapter.ps.gz`.

[5] T.N. Bui and C. Jones, "Finding good approximate vertex and edge partitions is NP-hard," *Information Processing Letters* 42, pp. 153-159, 1992.

[6] R. Diekmann, (organizer) "Graph Partitioning - Graph Collection", AG-Monien, University of Paderborn, available at:
`http://www.uni-paderborn.de/fachbereich/AG/monien/RESEARCH/PART/graphs.html`

[7] I.S. Duff, R.G.G. Grimmes, and J.G. Lewis, "Sparse Matrix Test Problems," *ACM Trans. on Math. Software* 15(1), pp. 1-14, 1989.

[8] M.R. Garey, D.S. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theoretical Computer Science* 1, pp. 237-267, 1976.

[9] F. Glover, "Tabu Search - part I," *ORSA Journal on Computing* 1(3), pp. 190-260, 1989.

[10] A. Gupta, "Fast and effective algorithms graph partitioning and sparse-matrix ordering," *IBM Journal of Research and Development* 41(1/2), pp. 171-184, 1997.

[11] P. Hansen and B. Jaumard, "Algorithms for the maximum satisfiability problem," *Computing* 44, pp. 279-303, 1990.

[12] B. Hendrickson and R. Leland, "An improved spectral graph partitioning algorithm for mapping parallel computations," Tech. Rep. SAND92-1460, SANDIA National Laboratories, Albuquerque, NM, 1992.

[13] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," Tech. Rep. SAND93-1301, SANDIA National Laboratories, Albuquerque, NM, 1993.

[14] B. Hendrickson and R. Leland, "The Chaco user's guide," Tech. Rep. SAND94-2692, SANDIA National Laboratories, Albuquerque, NM, 1993.

[15] M.R. Karp, "Reducibility among combinatorial problems,". In: R.E.Miller and J.W.Thatcher (Eds.), *Complexity of Computer Computations*, Plenum Press, New York, pp. 85-104, 1972.

[16] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," Techn. Rep. 95-035, University of Minnesota, Department of Computer Science, 1995.

[17] G. Karypis and V. Kumar, "Multilevel k-way Partitioning Scheme for Irregular Graphs," Techn. Rep. 95-064, University of Minnesota, Department of Computer Science, 1995.

[18] G. Karypis and V. Kumar, "Analysis of Multilevel Graph Partitioning.," Technical Report SAND93-1301, Sandia National Laboratories, 1995.

[19] B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.* 49, pp. 291-307, 1970.

[20] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjamin/Cummings Publishing Company, Redwood City, CA, 1994.

[21] G.L. Miller, S.H. Teng, and S.A. Vavasis, "A unified geometric approach to graph separators," *Proceedings of 31st Annual Symposium of Foundations of Computer Science*, pp. 538-547, 1991.

[22] A.G. Monien and R. Diekman, "A Local Graph partitioning Heuristic Meeting Bisection Bounds," *Proc. 8th SIAM Conf. on Parallel Processing for Scientific Computing*, 1997.

[23] A. Pothen, H.D. Simon, and K.P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications* 11(3), pp. 430-452, 1990.

[24] A. Pothen, H.D. Simon, L. Wang, and S.T. Barnard, "Toward a fast implementation of spectral nested dissection," *Proc. Supercomputing '92*, pp. 42-41, 1992.

[25] R. Preis and R. Diekmann, "The Party Partitioning Library, User guide," Techn. Rep. TR-RSFB-96-024, Univ. of Paderborn, Germany, 1996.

[26] R. Van Driessche, "Algorithms foe static and dynamic Load Balancing on Parallel Computers," PhD thesis, KU Leuven, 1995.

[27] C. Walshaw and M. Berzins, "Dynamic load-balancing for PDE solvers on adaptive unstructured meshes," *Concurrency: Practice and Experience* 7(1), pp. 17-28, 1995.

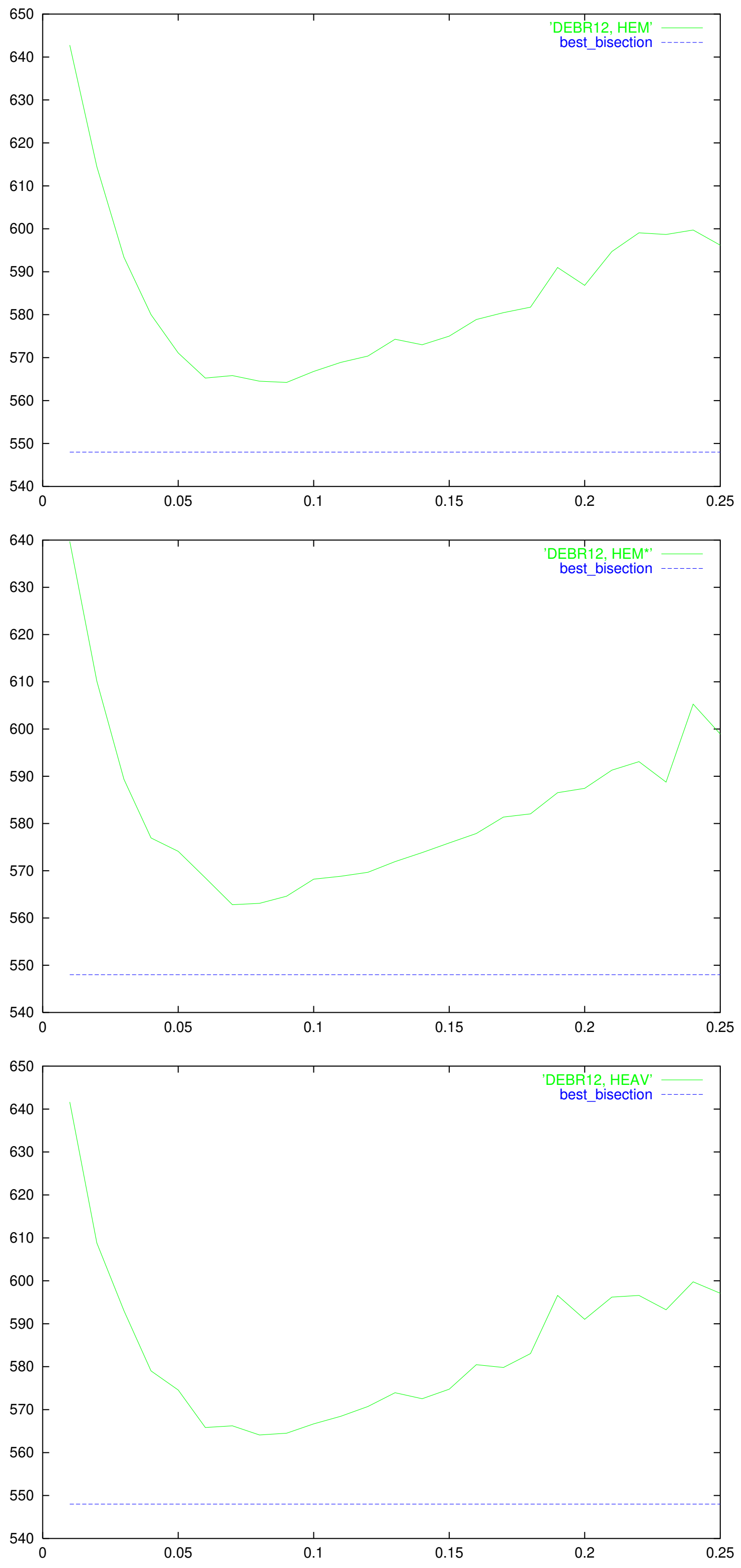


Figure 3: DEBR12 partitioned by the MULTILEVEL-FTS algorithm, using the HEM, HEM*, and HEAV matching heuristics.

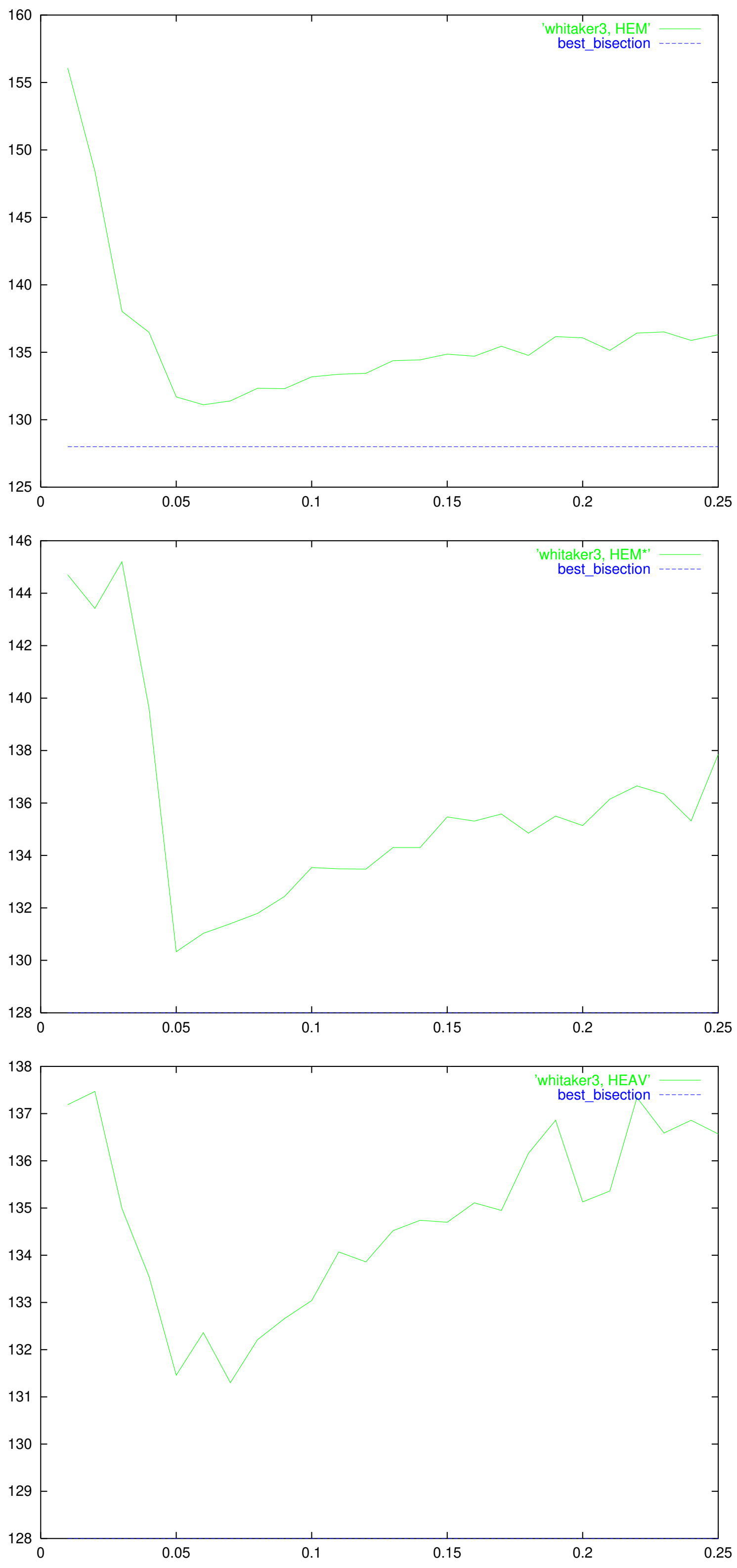
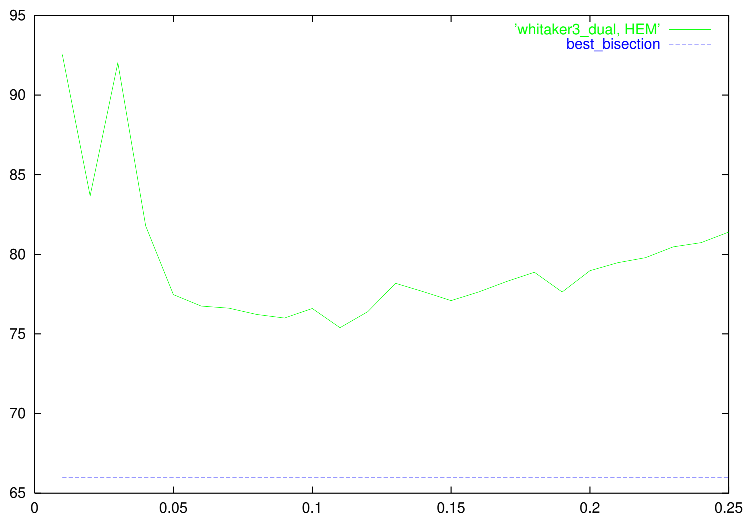
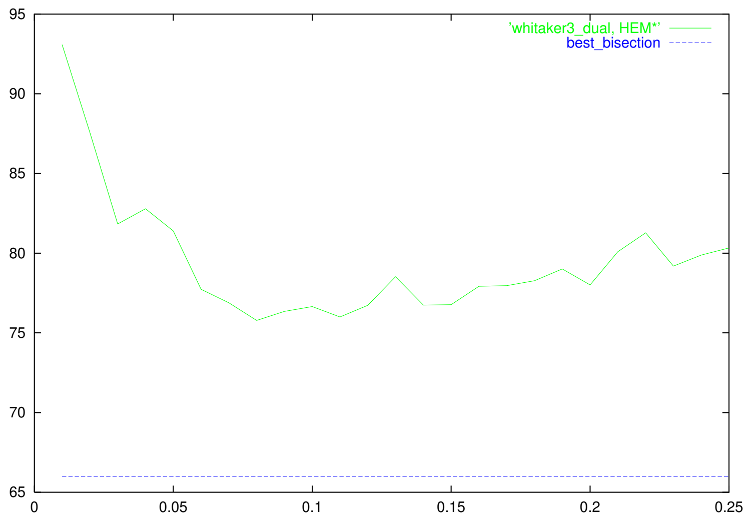


Figure 4: whitaker3 partitioned by the Multilevel-FTS algorithm, using the HEM, HEM*, and HEAV matching heuristics.





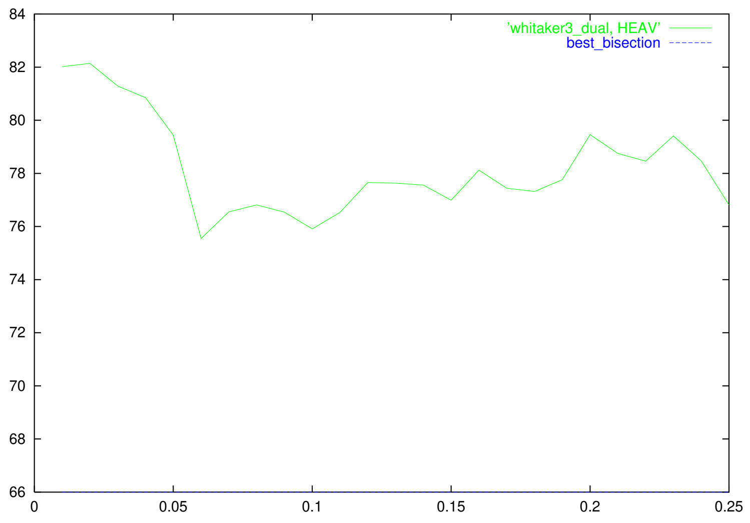
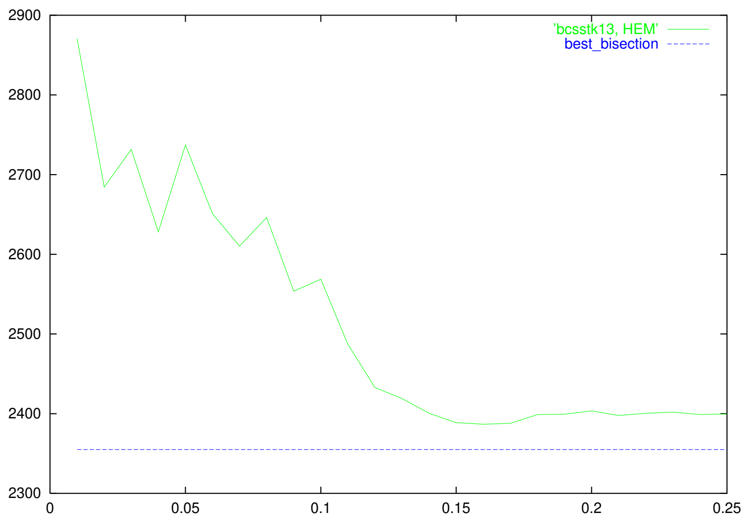
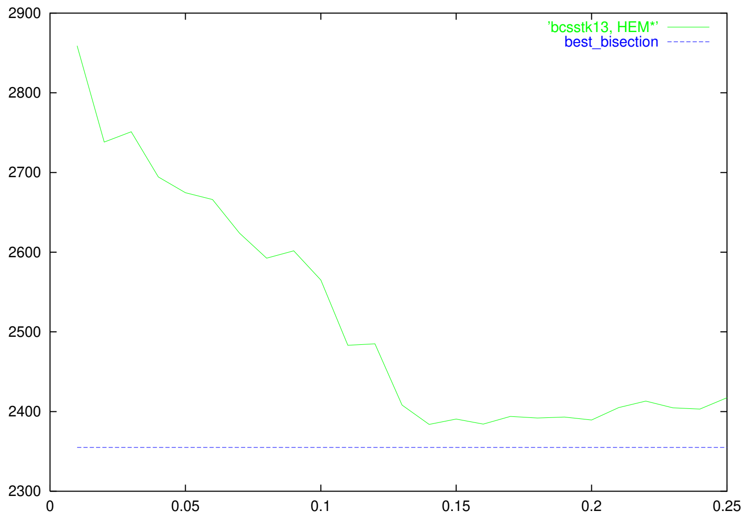


Figure 5: whitaker3_dual partitioned by the Multilevel-FTS algorithm, using the HEM, HEM*, and HEAV matching heuristics.





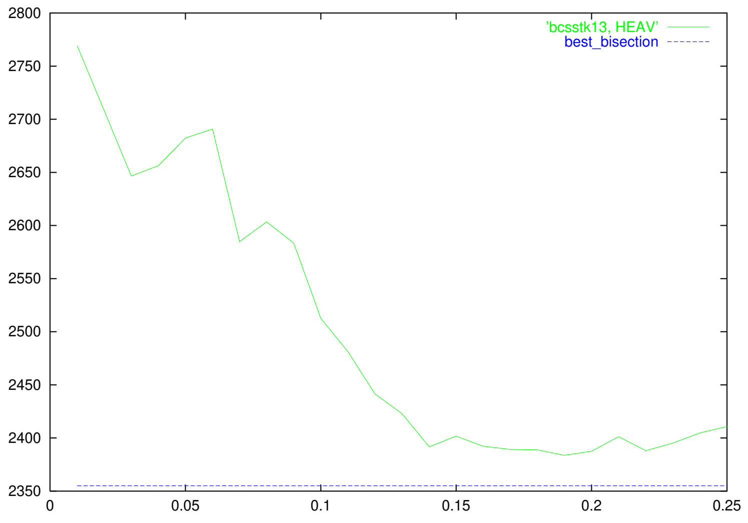


Figure 6: bcsstk13 partitioned by the MULTILEVEL-FTS algorithm, using the HEM, HEM*, and HEAV matching heuristics.

	MULTILEVEL- FTS									
	matching HEM			matching HEM*			matching HEM+HEAV			Best
	min	average	CPU	min	average	CPU	min	average	CPU	
grid2	34	34.4	0.23	34	34.27	0.25	34	34.35	0.27	34
airfoil1	75	81.55	0.35	75	81.68	0.38	75	82.13	0.35	77
3elt	90	102.23	0.4	90	100.29	0.41	90	101.08	0.42	90
ukerbe1	27	28.81	0.48	27	29.25	0.5	27	29.22	0.5	28
whitaker3	128	131.81	0.88	127	130.6	0.91	127	131.54	0.87	128
crack	185	194.17	1.07	184	192.89	1.16	186	194.65	1.09	190
big	142	158.43	1.54	140	160.2	1.6	141	156.41	1.56	142
grid2-dual	32	32.53	0.22	32	32.74	0.22	32	32.64	0.25	32
airfoil1-dual	37	45.51	0.65	38	45.77	0.64	37	44.86	0.62	42
3elt-dual	46	57.14	0.73	46	55.91	0.74	46	57.53	0.73	49
ukerbe1-dual	21	22.83	0.13	21	23.48	0.13	21	22.63	0.13	21
whitaker3-dual	66	75.26	1.87	66	76.17	1.98	67	75.79	1.98	66
crack-dual	80	92.09	2.04	82	92	2.08	80	91.44	2.04	93
big-dual	75	91.49	3.62	74	93.05	3.73	73	92.57	3.80	89
bcspwr09	9	14.75	0.17	9	14.55	0.19	9	14.05	0.19	9
bcsstk13	2355	2573.18	1.6	2355	2568.07	1.84	2355	2505.01	1.84	2355
nasa4704	1292	1362.08	1.39	1311	1359.61	1.5	1304	1360.52	1.45	1292
DEBR12	556	567.68	0.51	556	567.22	0.51	556	565.62	0.52	548
DEBR13	1070	1080.12	1.2	1070	1080.34	1.29	1064	1078.61	1.41	1032
DEBR14	1960	2022.38	3.42	1960	2014.74	3.6	1958	2021.7	3.84	1928
DEBR15	3686	3762.79	10.9	3674	3759.31	10.98	3680	3758.4	12.4	3608
DEBR16	6876	7100.8	36.9	6870	7008.34	37.66	6882	7078.7	47.25	6646
U1000.20	222	238.03	0.25	222	237.34	0.27	222	239.25	0.26	222
U1000.40	737	742.07	0.51	737	749.92	0.56	737	746.96	0.51	737
G1000.01	1374	1394.14	0.19	1373	1394.43	0.3	1376	1392.8	0.7	1384
G1000.02	3397	3430.71	0.31	3394	3428.59	0.64	3397	3430.59	1.12	3408

Figure 7: Results of MULTILEVEL-FTS using the HEM, HEM*, and HEM+HEAV matching heuristics.

	MULTILEVEL-RRTS									
	matching HEM			matching HEM*			matching HEM+HEAV			Best
	min	average	CPU	min	average	CPU	min	average	CPU	
grid2	34	36.51	0.9	34	36.59	0.79	34	36.12	0.93	34
airfoil1	74	85.41	1.62	74	86.25	1.42	74	83.65	1.35	77
3elt	90	103.48	1.64	90	105.93	1.95	90	104.27	1.6	90
ukerbe1	27	31.24	1.23	27	30.67	1.27	27	31.67	1.31	28
whitaker3	127	135.75	2.76	127	135.39	2.77	127	143.33	2.92	128
crack	184	196.95	3.25	185	197.98	3.65	184	198.05	4.35	190
big	139	168.7	4.66	140	174.84	4.07	139	172.2	4.54	142
grid2-dual	32	35.47	0.76	32	34.66	0.75	32	34.28	0.83	32
airfoil1-dual	37	46.06	1.20	36	46.63	1.24	37	45.85	1.22	42
3elt-dual	45	61.71	1.45	46	61.08	1.62	45	58.11	1.45	49
ukerbe1-dual	21	23.46	0.61	21	23.97	0.76	21	23.52	0.57	21
whitaker3-dual	67	79.92	2.98	65	80.11	3.36	66	80.72	3.43	66
crack-dual	81	94.93	3.05	81	94.44	2.91	78	93.14	3.25	93
big-dual	76	99.18	5.27	74	104.22	5.47	73	98.74	6.2	89
bcspwr09	9	14.09	0.65	9	13.62	0.68	9	11.71	0.93	9
bcsstk13	2355	2579.79	43.42	2355	2604.59	36.8	2355	2574.74	44.06	2355
nasa4704	1292	1372.75	16.38	1292	1369.92	16.65	1292	1374.14	15.06	1292
DEBR12	556	600.56	2.53	556	586.6	2.61	556	591.9	2.67	548
DEBR13	1072	1123.06	4.68	1072	1121.9	4.72	1068	1112.39	5.32	1032
DEBR14	1958	2118.32	9.32	1958	2096.42	10.89	1960	2095.86	10.73	1928
DEBR15	3688	3918.48	22.59	3692	3925.68	20.59	3690	3939.06	29.56	3608
DEBR16	6882	7352.48	51.6	6880	7370.86	50.37	/	/	/	6646
U1000.20	222	247.52	4.4	222	253.17	4.57	222	251.35	3.62	222
U1000.40	737	782.98	12.21	737	789.96	12.06	737	776.55	11.71	737
G1000.01	1372	1414.12	1.76	1373	1414.27	1.84	1382	1421.62	2.86	1384
G1000.02	3403	3453.83	2.74	3387	3452.89	3.05	3406	3454.8	3.7	3408

Figure 8: Results of MULTILEVEL-RRTS using the HEM, HEM*, and HEM+HEAV matching heuristics.