Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 127 (2003) 665–678
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Heuristics and meta-heuristics for 2-layer straight line crossing minimization☆

Rafael Martí[a], Manuel Laguna[b]

[a] *Departamento de Estadística e I.O. Universidad de Valencia. Dr. Moliner 50, 46100 Burjassot (Valencia), Spain*

[b] *Leeds School of Business, University of Colorado, 419 UCB, Boulder, CO 80309, USA*

Received 30 January 1998; received in revised form 4 April 2000; accepted 8 October 2001

Abstract

This paper presents extensive computational experiments to compare 12 heuristics and 2 meta-heuristics for the problem of minimizing straight-line crossings in a 2-layer graph. These experiments show that the performance of the heuristics (largely based on simple ordering rules) drastically deteriorates as the graphs become sparser. A tabu search metaheuristic yields the best results for relatively dense graphs, with a GRASP implementation as close second. Furthermore, the GRASP approach outperforms all other approaches when tackling low-density graphs.

1. Introduction

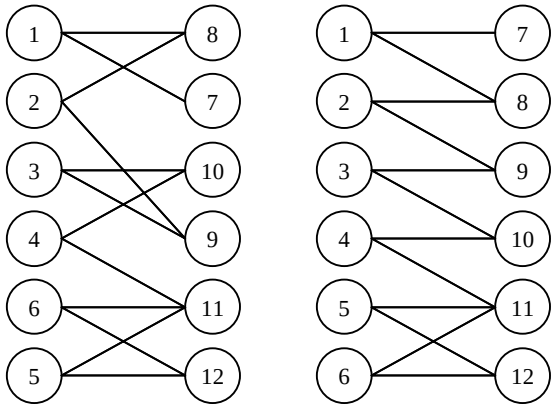
Researches in the graph-drawing field have proposed several aesthetic criteria that attempt to capture the meaning of a "good" map of a graph. Although readability may depend on the context and the map's user, most authors agree that crossing reduction is a fundamental aesthetic criterion in graph drawing. In the context of a 2-layer graph and straight edges, the bipartite drawing problem or BDP consists of ordering the vertices in order to minimize the number of crossings. Fig. 1 shows two drawings of a sample graph with 12 vertices, 6 in each layer, and 12 edges. The drawing labeled "a" has 5 crossings, while the drawing labeled "b" has 1.

A *bipartite graph* $G = (V, E)$ is a simple directed graph where the set of vertices V is partitioned into two subsets L (the left layer) and R (the right layer) and where

☆ Electronic Annexes available. See http://www.elsevier.com/locate/dam.

E-mail addresses: rafael.marti@uv.es (R. Martí), laguna@colorado.edu (M. Laguna).

0166-218X/03/$ - see front matter
PII: S0166-218X(02)00397-9



(a) Crossings = 5 (b) Crossings = 1

Fig. 1. Sample drawings.

$E \subseteq L \times R$. Note that the direction of the arcs has no effect on crossings so throughout this paper we consider G to be an undirected graph, the arcs to be edges and denote G by the triple (L, R, E).

We assume that the vertices in L and R are arranged in the vertical lines $x = 0$ and $x = 1$ respectively, and the edges are straight lines. Thus, a *drawing* of the graph is specified with a unique y-coordinate for each vertex. Let $h(u)$ be the y-coordinate of vertex u in L. Similarly, let $r(u)$ be the y-coordinate of vertex u in R. Given that the number of crossings does not depend on the precise position of vertices but only on the ordering of the vertices, we say that h and r are orderings of L and R, respectively, and denote a drawing by the pair (h, r).

For each vertex $u \in L$ the set of its adjacent vertices (neighbors) is denoted by $N_u = \{v \in R/(u, v) \in E\}$ and its degree by d_u. Similar definitions apply to a vertex $u \in R$. The density of the graph is the quotient between the number of edges on the graph and the number of edges on the complete graph over the same number of vertices. In a bipartite graph $G = (L, R, E)$ the density is $|E|/|L|\,|R|$.

A *crossing* is a set $\{(u, v),\ (w, x)\} \subseteq E$ of two edges where $u, w \in L$ and $v, x \in R$ with either $h(u) < h(w)$ and $r(v) > r(x)$ or $h(u) > h(w)$ and $r(v) < r(x)$. A drawing is *optimal* if there is no other with fewer crossings. The optimal solution to the BDP is an optimal drawing. A drawing (h, r) is left optimal when there is no other ordering h' of L so that (h', r) has fewer crossings. A left optimal drawing is an optimal solution to the level permutation problem (LPP), for which the ordering r of R is fixed. A drawing (h, r) is right optimal if there is no other ordering r' of R so that (h, r') has fewer crossings. As shown by Eades and Kelly [3], if a drawing is simultaneously right and left optimal this does not necessarily imply that the drawing is also optimal with respect to the BDP.

Suppose that r is a fixed ordering of R and i and j are two vertices of L, then $K(i, j)$ is the number of crossings that edges incident to vertex i cause with edges incident to

vertex j, when vertex i precedes vertex j in the ordering of L. Note that given i and $j \in L$, the value of $K(i,j)$ does not depend on the position of the remaining vertices of L, but it does dependent on the ordering of the vertices of R. The value $K(i,j)$ is similarly defined relative to i and j vertices of R, for an ordering h of L.

Given two orderings h and r of L and R, respectively, let $C(G,h,r)$ be the total number of edge crossings of G with respect to these orderings. $C(G,h,r)$ can be obtained as the sum of the values $K(i,j)$ with $h(i) < h(j)$ for all the vertices i and j in L (or similarly as the sum of all the $K(i,j)$ with $r(i) < r(j)$ for all vertices i and j in R):

$$C(G,h,r) = \sum_{h(i)<h(j)} K(i,j).$$

The BDP is known to be NP-complete [11]. Both the BDP and the general problem with more than 2 layers have been the subject of study for at least 17 years. Several heuristic algorithms have been proposed throughout the years, beginning with the Relative Degree Algorithm introduced by Carpano [1]. These heuristics are based on simple ordering rules, reflecting the goal of researchers and practitioners of quickly obtaining solutions of reasonable quality. Recent developments, however, have shown that meta-heuristic approaches can be successfully applied to both the BDP and the general multi-layer problem [15,16,18,20]. Also, an exact procedure due to Valls et al. [21] has been developed for the BDP. Additionally, for the special case when the ordering in one of the layers is considered fixed, Jünger and Mutzel [12] have developed a Branch and Cut procedure.

In this paper we undertake to explore the behavior of the most relevant heuristic approaches developed for the BDP. With this goal in mind, we have generated a large number of instances with a wide range of sizes and densities. In the following sections we first provide short descriptions of the heuristics that are more often used in graph drawing systems. We then present a summarized description of two meta-heuristic approaches, one based on GRASP [15] and the other on the tabu search methodology [18]. These sections are followed by the results of our computational testing with more than 3000 graphs.

2. Heuristic approaches

Since the first procedure developed by [1], most heuristics for the BDP are structured in a similar way. Specifically, the procedures first order one layer employing a simple rule, while keeping the position of the vertices in the other layer fixed. Then the other layer is ordered and the process repeats until two successive iterations occur in which the relative positions of the vertices remain the same. The following summarizes the most relevant work in the area to the present.

The best-known approach to the BDP is the so-called Barycenter Method (BC1), which is similar to Carpano's algorithm [19]. In this algorithm, the position of a given vertex is calculated as the arithmetic mean of the positions of its adjacent vertices. The basic principle of this rule is that crossings are likely to be minimized by increasing

the number of horizontal arcs. Gansner [9] refined the barycenter procedure by incorporating a tie-breaking mechanism (BC2).

Eades and Kelly [3] propose the Splitting (SP) and the Greedy Switching (GS) approaches. In the Splitting procedure, a vertex u is chosen and every other vertex v is placed above or below u with the goal of minimizing the number of crossing due to the current positions of u and v. Then the process is recursively applied to order the set of vertices below and above u. (In this description we assume that the vertices are drawn in two parallel columns, each one corresponding to a layer, as shown in Fig. 1.) Greedy Switching considers all consecutive pairs of vertices and exchanges their positions as long as this move decreases the number of crossings.

Eades and Wormald [4] developed the Median heuristic (MED1). This ordering rule is similar to the barycenter method, with the difference that it employs the median instead of the average to compute the position of each vertex. Eades and Wormald [5] later proposed a variant of this method (MED2). A hybrid of the Median and Barycenter methods is presented in Makinen [17] and is known as the Semimedian heuristic (SM).

In our study we have also considered combinations of pairs of algorithms denoted by (A_B), as typically done in some graph drawing systems. In these hybrid approaches, the algorithms are performed in sequence, such that the second procedure (i.e., B) starts from the solution obtained by the first one (i.e., A). The nature of the second algorithm must be such that it can be applied as a descent method capable of finding local optima (such as a switching heuristic). Thus, not all combinations are possible. In particular, we focus our attention to 5 combined algorithms: BC2_GS [9], SM_GS [8], SP_GS, BC2_SP and GS_SP.

3. GRASP approach

The GRASP methodology was developed in the late 1980s, and the acronym was coined by Feo [7]. It was first used to solve computationally difficult set covering problems [6]. Each GRASP iteration consists of constructing a trial solution and then applying an exchange procedure to find a local optimum (i.e., the final solution for that iteration). The details of the two GRASP phases developed by Laguna and Martí [15] follow:

Construction phase. This phase starts by creating a list of unassigned vertices (U), which originally consists of all the vertices in the graph. The first vertex v is randomly selected from all those vertices in U with maximum degree. Then U is updated by deleting v from the list. In subsequent construction steps, the next vertex v is selected from a list of vertices with a degree of no less than 2/3 of the maximum degree of all the vertices in U. Vertex degree, in this case, is calculated with respect to the subgraph given by the partial solution obtained from previous vertex selections. (Note that this is different from the first vertex selection, where maximum degree is calculated with respect to the original graph.)

A chosen vertex v is placed as prescribed by the barycenter calculation. If the position is already occupied by a previously chosen vertex, then vertex v is placed either one

position above or one position below the barycenter (whichever produces the least number of crosses with respect to the partial construction). Once v has been positioned in the partial solution, the vertex is deleted from the list U and the vertex degree calculations (corresponding to the remaining elements of U) are updated accordingly. The construction phase terminates when all vertices have been selected (i.e., $U = \emptyset$).

Improving phase. In each step of the improving phase vertices are selected to be moved. The probability of selecting a vertex increases with its degree. When a vertex is selected, three moves are considered: (1) to insert the vertex one position above the barycenter, (2) to insert the vertex at the barycenter position, and (3) to insert the vertex one position below the barycenter. The vertex is placed in the position that produces the maximum reduction in the number of crosses. If no reduction is possible, then the vertex is not moved. An improving step terminates when all vertices have been considered for insertion. (Note that within the same improving step, some vertices may be moved while others may stay in their original positions.) More steps are performed as long as at least one vertex is moved (i.e., as long as the current solution keeps improving).

When a step fails to improve the current solution, and before abandoning the improving phase, an attempt is made to exchange adjacent vertices in order to find an improved solution. This process is performed on each layer, from top to bottom, according to the vertex order in the current solution.

The GRASP implementation also includes strategies for by-passing the improving phase in order to accelerate the search when constructions are not likely to yield a better solution than the best known (i.e., the incumbent solution recorded during the search). Also, a path relinking process is implemented to increase search efficiency, but this feature has been disallowed for the purpose of our study. (Path Relinking has been proposed in the context of tabu search as described in Glover and Laguna [10].)

4. Tabu search approach

Tabu Search (TS) is a technique that employs adaptive memory and that has been used for solving hard combinatorial optimization problems [10].

The TS adaptation to the BDP developed by Martí [18] has three different search states: Normal, Influential and Opposite, and it oscillates among them according to the search history. In each state there are two alternately applied phases, an Intensification Phase and a Diversification Phase. In the Intensification Phase only improving moves may be performed. In the Diversification phase both non-improving and improving moves may be performed with the aim of escaping from local optima and directing the search to regions in the solution space that have not been explored. The meaning of "improving" is not limited to the objective function evaluation, since it considers factors such as move influence (as determined by the search history and the context).

At each iteration of the intensification phase the procedure considers removing a vertex from its current position and inserting it in its barycenter. The move evaluation in the Normal search state is given by the change in the number of crossings when the move is performed. In the intensification phase the procedure only allows a move

to be made when the number of crossings strictly diminishes. Thus, it is considered that after a specified number of iterations have been reached without improvement, the current ordering is sufficiently close to the local optimum to stop this phase.

The moves defined in the intensification are based on a positioning function while those defined in the diversification are based on permuting consecutive vertices. The use of different moves fulfills the role of altering the terrain visited in a way that is unlikely to occur by always applying the same kind of moves; thus reinforcing the non-monotonic search strategy common to tabu search.

The algorithm starts in the normal state. If the best solution does not improve after a number of iterations, then the state begins to oscillate following the sequence: Normal, Influential, Normal, Opposite, Normal, Influential, etc. When the best solution is improved the state is re-initialized to Normal.

The Intensification and Diversification Phases are alternately applied to all three search states. The move definition in each phase is the same in the three states (as described earlier). In the Normal and Opposite states the move evaluation is given by the change in the number of crossings when the move is performed. In the Influential state the evaluation is given by the change in the solution structure. The objective of this search state is to change the solution structure by moving vertices with large degree, which tend to occupy the same position for many iterations.

Finally, the objective of the Opposite search state is to provide an aggressive diversification scheme that permits the discovery of new solutions in "far" regions usually non visited. Thus, reinforcing the long-term memory diversification.

5. Computational comparison

All the procedures described in the sections above were implemented in C, and all experiments were performed on a Pentium-150 personal computer. Our first set of experiments was done on graph instances generated with the `random_bigraph` code of the Stanford GraphBase by Knuth [13]. This generator is hardware independent so the experiments are fully reproducible.

In our first experiment, the instances have 10 vertices in each layer and a number of edges (m) ranging from 10 to 90. There are a total of 900 instances (100 instances for each edge density). For this relatively small graph size the optimal solution can be found with the branch-and-bound code developed by Valls et al. [21]. Tables 1–3 report, respectively, the average number of crossings, the average deviation from the optimal solutions (see OPT in Table 1), and the average CPU seconds.

These tables show that the best solution quality is obtained by the tabu search method (TS), which is able to match all known optimal solutions. However, TS employs more computational time than the other methods (15 seconds on average, compared to less than a second for all other methods). GRASP is very competitive, considering its average percent deviation from optima of 0.27% achieved on an average of 0.09 s. BC2_SP is the best of the combined algorithms with 30.5% average percent deviation from optima, while SP is the best of the simple procedures (33.4%) closely followed by BC2 (34.2%).

Table 1
Average number of crossings on 10 + 10 nodes

$m=$	10	20	30	40	50	60	70	80	90	Average
opt	0.29	11.62	56.6	146.89	276.78	463.17	698.35	1008.38	1405.57	451.96
bc1	2.21	20.32	68.13	160.31	289.03	476.03	712	1026.72	1424.17	464.32
bc2	2.21	20	66.51	159.1	288.07	475.52	711.08	1024.27	1422.12	463.20
med1	2.05	26.1	83.8	188.51	329.3	530.21	769.61	1099.03	1496.26	502.76
med2	1.7	23.85	79.16	181.54	316.87	511.13	748.1	1066.88	1450.4	486.62
sm	2.38	21.33	71.17	167.5	302.84	494.33	728.73	1052.5	1442.04	475.86
gs	9.67	33.17	81.17	164.21	293.75	481.39	714.58	1021.46	1424.48	469.32
sp	1.93	21.45	69.26	158.55	290.61	481.48	712.03	1022.6	1419.86	464.19
bc2_gs	2.21	19.92	66.02	158.55	287.21	474.75	709.68	1023.46	1421.21	462.55
sm_gs	2.34	20.5	67.43	160.32	290.6	477.59	710.88	1023.03	1423.91	464.06
sp_gs	1.93	21.45	69.26	158.55	290.6	481.48	712.03	1022.6	1419.86	464.19
bc2_sp	1.99	19.35	64.77	157.17	286.43	474.35	709.69	1023.42	1421.11	462.03
gs_sp	2.11	20.25	68.64	158.56	291.46	479.18	713.71	1021.37	1424.13	464.37
tabu	0.29	11.62	56.6	146.89	276.78	463.17	698.35	1008.38	1405.57	451.96
grasp	0.29	11.77	57.09	147.20	276.99	463.47	698.46	1008.46	1405.64	452.15

Table 2
Average percent deviation from optima on 10 + 10 nodes

m	10	20	30	40	50	60	70	80	90	Average
bc1 (%)	188.3	85.0	21.0	9.3	4.4	2.7	1.9	1.8	1.3	35.1
bc2 (%)	188.3	81.8	18.1	8.4	4.1	2.6	1.8	1.6	1.2	34.2
med1 (%)	170.5	148.3	50.1	28.6	19.1	14.6	10.3	9.0	6.5	50.8
med2 (%)	135.7	126.3	41.7	23.8	14.5	10.4	7.2	5.8	3.2	40.9
sm (%)	204.8	97.4	26.2	14.0	9.5	6.8	4.3	4.4	2.6	41.1
gs (%)	905.1	211.8	44.2	11.8	6.2	3.9	2.3	1.3	1.3	132.0
sp (%)	157.7	98.7	23.2	8.0	5.0	3.9	1.9	1.4	1.0	33.4
bc2_gs (%)	188.3	81.0	17.2	8.0	3.7	2.5	1.6	1.5	1.1	33.9
sm_gs (%)	200.8	88.0	19.4	9.1	5.1	3.1	1.8	1.5	1.3	36.7
sp_gs (%)	157.7	98.7	23.2	8.0	5.0	3.9	1.9	1.4	1.0	33.4
bc2_sp (%)	166.3	76.3	14.7	7.1	3.5	2.4	1.6	1.5	1.1	30.5
gs_sp (%)	177.7	81.6	21.9	8.1	5.3	3.5	2.2	1.3	1.3	33.6
tabu (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
grasp (%)	0.00	1.23	0.82	0.20	0.07	0.06	0.02	0.01	0.00	0.27

These tables also show that the performance of heuristics quickly deteriorates as the graph become sparser. In high-density graphs the performance of all the heuristic methods are very similar (see $m = 90$ column in Table 2). These results lead us to the observation that the lower the density the more difficult the problem. Moreover, considering that low-density graphs are more commonly found in real-world applications, we make this graphs the focus of our attention.

In our second experiment we generate 900 additional instances with number of vertices in each layer equal to the number of edges. We generate sets of 100 instances with number of edges ranging from 10 to 90 in increments of 10. The BEST row

Table 3
Average CPU seconds on 10 + 10 nodes

m	10	20	30	40	50	60	70	80	90	Average
bc1	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01
bc2	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.01	0.01
med1	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.00	0.01
med2	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.00
sm	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.01	0.01	0.01
gs	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.00
sp	0.00	0.00	0.00	0.00	0.01	0.01	0.02	0.03	0.03	0.01
bc2_gs	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01
sm_gs	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.01
sp_gs	0.00	0.00	0.01	0.01	0.02	0.02	0.02	0.03	0.03	0.01
bc2_sp	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.02	0.02	0.01
gs_sp	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.02	0.01
tabu	0.43	0.82	1.51	2.50	3.93	8.12	17.50	39.35	60.81	15.00
grasp	0.01	0.02	0.04	0.06	0.08	0.11	0.14	0.18	0.18	0.09

Table 4
Average number of crossings on sparse graphs

m	10	20	30	40	50	60	70	80	90	Average
best	0.29	1.07	2.25	3.6	5.2	7.06	8.68	10.26	15.03	5.9
bc1	2.21	8.52	16.76	26.04	39.06	51.5	70.85	78.56	97.83	43.5
bc2	2.21	8.57	16.53	25.81	38.13	50.29	70.06	79.45	95.94	43.0
med1	2.05	9.31	24.87	42.91	69.54	96.23	137.1	182.5	225.3	87.8
med2	1.7	8.25	23.09	40.6	67.2	92.24	132.2	178	219.9	84.8
sm	2.38	8.16	18.46	31.13	49.41	65.25	91.83	108.8	136.6	56.9
gs	9.67	62.9	161.1	316.8	515.1	755.6	1071	1425	1818	681.7
sp	1.93	11.71	26.06	49.18	82.46	107.5	152.2	199.1	254.3	98.3
bc2_gs	2.21	8.5	16.38	25.36	37.47	49.31	68.43	77.55	94.11	42.1
sm_gs	2.34	7.99	17.96	30.28	48.65	63.99	90.15	106.9	134.4	55.9
sp_gs	1.93	11.71	26.06	49.18	82.46	107.5	152.2	199.1	254.3	98.3
bc2_sp	1.99	8.41	15.96	25.28	37.39	48.82	68.04	77.27	93.89	41.9
gs_sp	2.11	11.48	26.76	50.31	82.21	113.4	151.7	201.2	244.2	98.1
tabu	0.29	1.07	2.39	4.91	7.88	12.02	15.82	20.14	27.4	10.2
grasp	0.29	1.09	2.27	3.62	5.23	7.09	8.74	10.26	15.14	5.9

represents the minimum number of crossings found for each instance after running all procedures during the experiment. (We cannot assess how close the BEST values are from the optimal solutions, so we are only using these values as a way of comparing the methods.)

Tables 4–6 show the remarkable performance of GRASP when tackling instances with low edge density. An average percent deviation of 0.2% is achieved by GRASP, which compares quite well with the average deviation of 97.5% corresponding to the tabu search procedure. The computational effort associated with the GRASP is still modest (2.01 s). The performance of the simple and combined algorithms is clearly

Table 5
Average percent deviation from the best known on sparse graphs

m	10	20	30	40	50	60	70	80	90	Average
bc1 (%)	188	619	926	1090	1253	1330	1259	1081	1120	985
bc2 (%)	188	622	927	1073	1248	1306	1249	1088	1104	978
med1 (%)	171	645	1466	1821	2250	2710	2537	2844	2779	1914
med2 (%)	136	558	1357	1702	2163	2599	2442	2770	2694	1825
sm (%)	205	596	1056	1319	1709	1765	1659	1459	1614	1265
gs (%)	905	5146	10542	16814	20878	24114	23242	25237	27121	17111
sp (%)	158	887	1584	2233	3058	3181	2960	3194	3358	2290
bc2_gs (%)	188	617	919	1057	1226	1288	1222	1066	1087	963
sm_gs (%)	201	583	1024	1294	1689	1739	1625	1427	1590	1241
sp_gs (%)	158	887	1584	2233	3058	3181	2959	3194	3358	2290
bc2_sp (%)	166	612	891	1051	1224	1281	1217	1061	1085	954
gs_sp (%)	178	871	1578	2378	3038	3340	2990	3305	3246	2325
tabu (%)	0.0	0.0	11.3	70.7	116.9	158.3	158.3	178.8	183.3	97.5
grasp (%)	0.0	0.3	0.5	0.3	0.3	0.2	0.1	0.0	0.2	0.2

Table 6
Average CPU seconds on sparse graphs

m	10	20	30	40	50	60	70	80	90	Average
bc1	0.00	0.01	0.01	0.02	0.03	0.05	0.07	0.10	0.13	0.05
bc2	0.00	0.00	0.01	0.02	0.03	0.05	0.06	0.09	0.12	0.04
med1	0.00	0.00	0.01	0.01	0.02	0.03	0.05	0.06	0.08	0.03
med2	0.00	0.00	0.01	0.02	0.02	0.03	0.05	0.06	0.08	0.03
sm	0.00	0.01	0.01	0.02	0.04	0.06	0.08	0.11	0.16	0.06
gs	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00
sp	0.00	0.00	0.01	0.01	0.02	0.03	0.05	0.07	0.09	0.03
bc2_gs	0.00	0.00	0.01	0.02	0.03	0.05	0.07	0.09	0.12	0.04
sm_gs	0.00	0.00	0.01	0.02	0.04	0.06	0.08	0.12	0.16	0.06
sp_gs	0.00	0.00	0.01	0.01	0.02	0.03	0.05	0.07	0.09	0.03
bc2_sp	0.00	0.01	0.01	0.03	0.04	0.06	0.09	0.13	0.18	0.06
gs_sp	0.00	0.00	0.01	0.01	0.02	0.04	0.05	0.07	0.09	0.03
Tabu	1.65	3.24	5.87	8.53	11.62	15.14	19.08	21.86	27.09	12.68
grasp	0.00	0.05	0.18	0.44	0.93	1.86	3.07	4.65	6.91	2.01

inferior in this experiment, with average deviations several orders of magnitude larger than those achieved by the GRASP approach.

A third experiment was performed in denser graphs (relative to our second experiment). An additional set of instances was generated with equal number of vertices in each layer (ranging from 10 to 100) and twice as many edges as vertices in each layer (hence, ranging from 20 to 200). In Tables 7–9, we note that the TS procedure outperforms all other heuristics. The average deviation from the best-known values is 0.6% for the TS procedure, while GRASP obtains an average deviation of 10.6%. (TS, however uses 26 s versus 3.9 seconds for GRASP.) The rest of the heuristics under

Table 7
Average number of crossings

m	20	40	60	80	100	120	140	160	180	200	Average
best	11.6	45.4	105.6	178.6	282.2	424.3	570.3	742.5	960.1	1145.2	446.6
bc1	20.3	76.9	164.9	274.0	403.3	601.3	784.5	1012.1	1248.3	1507.6	609.3
bc2	20.0	75.8	163.5	270.0	397.8	596.9	780.5	1000.4	1225.8	1482.6	601.3
med1	26.1	110.7	251.8	445.7	706.1	1027.7	1418.3	1824.2	2312.6	2868.7	1099.2
med2	23.9	105.4	246.3	437.0	694.5	1015.4	1399.1	1806.6	2290.5	2846.5	1086.5
sm	21.3	82.5	191.1	319.5	489.8	722.2	974.7	1261.7	1592.5	1906.4	756.2
gs	33.2	211.7	569.0	1158.8	1888.5	2827.0	4042.9	5401.5	6934.6	8693.9	3176.1
sp	21.5	92.7	210.2	356.5	560.0	779.1	1059.2	1351.2	1685.5	2028.2	814.4
bc2_gs	19.9	74.8	160.6	264.4	388.9	583.2	761.9	977.0	1195.5	1446.6	587.3
sm_gs	20.5	79.2	184.4	308.8	474.2	701.6	948.5	1229.8	1552.7	1859.1	735.9
sp_gs	21.5	92.7	210.1	356.4	559.8	778.7	1058.7	1350.8	1684.8	2027.2	814.0
bc2_sp	19.4	74.1	159.6	262.0	384.9	575.1	752.0	962.4	1179.3	1422.1	579.1
gs_sp	20.3	92.5	209.8	362.6	538.0	801.4	1059.1	1364.7	1674.9	2042.2	816.5
tabu	11.6	45.5	105.9	179.3	285.0	426.9	573.8	746.7	967.6	1154.4	449.7
grasp	11.8	48.3	115.7	197.3	317.4	475.8	641.7	835.0	1061.6	1308.7	501.3

Table 8
Average percentage deviation

m	20	40	60	80	100	120	140	160	180	200	Average
bc1 (%)	85.0	76.5	59.9	58.5	45.3	43.5	39.7	37.9	31.6	32.5	51.0
bc2 (%)	81.8	73.8	57.6	56.5	43.2	42.4	38.8	36.2	29.1	30.3	49.0
med1 (%)	148.3	160.3	147.0	159.9	155.7	147.7	153.3	150.6	146.3	154.4	152.4
med2 (%)	126.3	148.0	141.3	155.0	151.7	144.8	149.9	148.2	143.9	152.4	146.1
sm (%)	97.4	93.6	86.1	85.5	76.2	72.7	73.8	72.8	68.4	68.7	79.5
gs (%)	211.8	402.0	461.3	578.8	584.9	585.5	628.7	644.8	640.1	672.9	541.1
sp (%)	98.7	115.2	104.8	108.5	103.0	88.0	89.4	85.9	79.1	79.8	95.2
bc2_gs (%)	81.0	71.5	54.8	53.2	39.9	39.1	35.5	32.9	25.9	27.1	46.1
sm_gs (%)	88.0	85.9	79.5	79.1	70.7	67.8	69.1	68.5	64.2	64.4	73.7
sp_gs (%)	98.7	115.2	104.7	108.4	103.0	87.9	89.4	85.8	79.0	79.7	95.2
bc2_sp (%)	76.3	69.9	53.9	51.7	38.5	37.2	33.7	30.9	24.2	25.0	44.1
gs_sp (%)	81.6	117.8	105.8	112.5	94.7	92.6	89.3	87.6	77.8	80.4	94.0
tabu (%)	0.0	0.4	0.3	0.5	1.1	0.6	0.7	0.6	0.7	0.8	0.6
grasp (%)	1.2	6.4	9.9	10.8	12.7	12.8	13.0	13.1	11.3	14.8	10.6

consideration yield average percent deviations in excess of 40%. As before, BC2_SP and BC2 are the best of the combined and single procedures respectively. Their average CPU times are 0.2 s.

Fig. 2 shows the average percentage distance of all algorithms to the best solution known on sparse and dense graphs. The percent of greedy switching heuristic (GS) in sparse graphs is in fact 17,111%, however it has been scaled down to 3000% for plotting. The procedures in Fig. 2 are shown in ascending order, as given by their percent deviation on sparse graphs.

Table 9
Average CPU seconds

m	20	40	60	80	100	120	140	160	180	200	Average
bc1	0.0	0.0	0.0	0.1	0.1	0.2	0.2	0.3	0.5	0.7	0.2
bc2	0.0	0.0	0.0	0.1	0.1	0.2	0.2	0.3	0.5	0.7	0.2
med1	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.2	0.1
med2	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.2	0.2	0.1
sm	0.0	0.0	0.0	0.0	0.1	0.1	0.2	0.3	0.4	0.5	0.2
gs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
sp	0.0	0.0	0.0	0.0	0.1	0.2	0.2	0.3	0.5	0.6	0.2
bc2_gs	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.3	0.5	0.7	0.2
sm_gs	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.3	0.4	0.5	0.2
sp_gs	0.0	0.0	0.0	0.0	0.1	0.2	0.2	0.3	0.5	0.6	0.2
bc2_sp	0.0	0.0	0.0	0.1	0.1	0.2	0.3	0.5	0.7	0.9	0.3
gs_sp	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.3	0.5	0.6	0.2
tabu	0.8	2.3	4.0	15.1	28.9	23.6	22.1	53.1	38.4	72.8	26.1
grasp	0.0	0.2	0.5	1.0	1.9	3.1	4.8	6.5	9.1	12.4	3.9

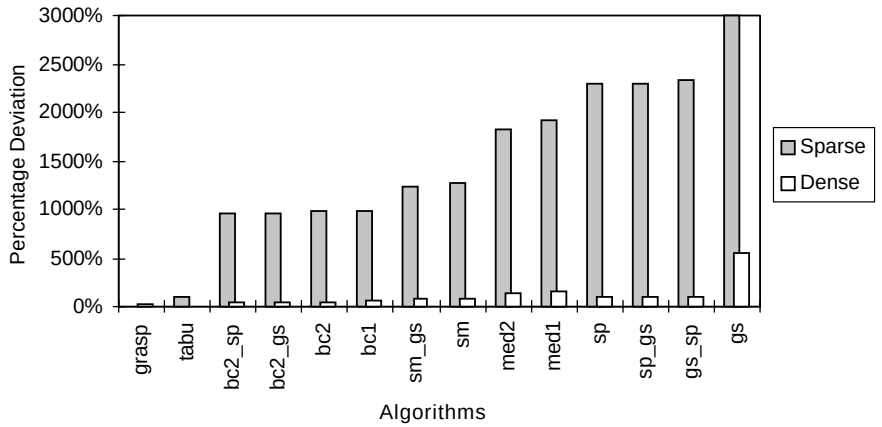


Fig. 2. Average percent deviation from the best solution known.

In Tables 10–12 we repeat the second experiment on sparse graphs comparing tabu search and GRASP to BC2_SP and BC2 with 25 starts from random orderings.

Note that an average percent deviation of 428% is achieved by BC2 from 25 random starts, which improves the average deviation of 970% corresponding to the single BC2. GRASP still dominates the approach of starting BC variants from random solutions, since the average deviation for GRASP is 0.2% and their average CPU times are equivalent (1.59 s for bc2_sp25 compared to 2.01 s for GRASP).

Since all our previous experimentation was done with the same random graph generator, we now perform a new set of experiments with a different generator. The goal of this experiment is to verify that our observations are not biased by the particular structure of the graphs generated by the `random_bigraph` routine. The new procedure

Table 10
Average number of crossings on sparse graphs

m	10	20	30	40	50	60	70	80	90	Average
best	0.3	1.1	2.3	3.6	5.2	7.1	8.7	10.3	15.0	5.9
bc2 25	0.3	2.2	6.5	12.7	19.3	27.5	36.2	47.3	60.2	23.6
bc2_sp 25	0.3	2.2	6.4	12.4	19.0	26.3	36.7	46.7	57.1	23.0
tabu	0.3	1.1	2.4	4.9	7.9	12.0	15.8	20.1	27.4	10.2
grasp	0.3	1.1	2.3	3.6	5.2	7.1	8.7	10.3	15.1	6.0

Table 11
Average percent deviation from the best known on sparse graphs

m	10	20	30	40	50	60	70	80	90	Average
bc2 25 (%)	2.0	98.4	275.1	455.3	582.5	634.4	553.3	631.4	621.6	428.2
bc2_sp 25 (%)	1.0	97.7	270.6	453.3	563.9	595.9	582.4	606.4	595.0	418.5
tabu (%)	0.0	0.0	11.3	70.7	116.9	158.3	158.3	178.8	183.3	97.5
grasp (%)	0.0	0.3	0.5	0.3	0.3	0.2	0.1	0.0	0.2	0.2

Table 12
Average CPU seconds on sparse graphs

m	10	20	30	40	50	60	70	80	90	Average
bc2 25	0.02	0.10	0.23	0.45	0.78	1.19	1.73	2.41	3.37	1.14
bc2_sp 25	0.04	0.15	0.34	0.65	1.10	1.67	2.43	3.40	4.57	1.59
tabu	1.65	3.24	5.87	8.53	11.62	15.14	19.08	21.86	27.09	12.68
grasp	0.00	0.05	0.18	0.44	0.93	1.86	3.07	4.65	6.91	2.01

constructs a graph in two steps using three parameters: the number of vertices in the left layer, the number of vertices in the right layer and the graph density. In the first step, the procedure forces all vertices to have a minimum degree of one. For each vertex in the left layer, it randomly selects a vertex on the right side and generates the corresponding edge; then for each vertex in the right layer with no incident edge, it randomly selects a vertex from the left layer and generates the corresponding edge. In the second step, it randomly generates the remaining edges necessary to give the graph the desired density. This is done by randomly selecting a vertex from the left side and one from the right. If there is no edge between them, it generates one.

We use this random graph generator to create a set of 300 instances. The graphs are divided in 15 groups of 20 instances. Each group has a different graph size but the same density. The graph sizes vary from $n = 14$ to $n = 100$. The exact procedure of Valls et al. [21] was applied to 140 of the smaller instance to determine their optimal solutions. The results of this additional experiment did not contradict our previous observations. The tabu search procedure is more effective than alternative approaches when dealing with denser graphs. The average deviation from optimality for the tabu

search implementation was 0.11%. In addition, the tabu search implementation yielded all the best-known solution to the 160 problems for which the optimal solution is not known. The performance of the heuristic approaches varied with the best being BC2_SP and the worst MED1.

6. Conclusions

A computational comparison of 14 existing methods for the BDP has been presented. Overall, experiments with 3100 graphs were performed to compare the procedures. This extensive experimentation allows us to conclude that the GRASP implementation seems better suited for relatively low-density graphs. As the density increases the tabu search approach [18] seem to be more appropriate, but if run time is critical a combination of the barycenter and splitting heuristics may be a better choice.

It is worth mentioning that Dell'Amico and Maffioli [2], who implemented and tested a TS procedure for the 2-partition problem, reached a similar conclusion with respect to GRASP and TS implementations. Their TS code was able to outperform the GRASP implementation of Laguna et al. [14] in all but the sparsest graphs.

The advantage of GRASP in sparse graphs seems to reside in the construction phase. TS approaches rely on move evaluations to find promising search directions, however, in sparse graphs, neighborhoods contain many moves with the same move value and tie-breaking mechanism are not immediately obvious. Although we do not have statistical evidence to justify this conjecture, we have observed this behavior in a sample set of instances. Determining the conditions under which multiple constructions followed by a limited search is a preferable approach to a long search from an initial solution is without a doubt an interesting topic for future studies.

References

[1] M.J. Carpano, Automatic display of hierarchized graphs for computer aided decision analysis, IEEE Trans. Systems Man Cybernetics 10 (11) (1980) 705–715.
[2] M. Dell'Amico, F. Maffioli, A new tabu search approach to the 0-1 equicut problem, in: I.H. Osman, J.P. Kelly (Eds.), Metaheuristics: Theory and Applications, 1996, pp. 361–377.
[3] P. Eades, D. Kelly, Heuristics for drawing 2-layered networks, Ars Combin. 21 (1986) 89–98.
[4] P. Eades, N.C. Wormald, The median heuristic for drawing 2-layered networks, Technical Report 69, Department of Computer Science, University of Queensland, 1986.
[5] P. Eades, N.C. Wormald, Edge crossing in drawings of bipartite graphs, Algorithmica 11 (1994) 379–403.
[6] T. Feo, M.G.C. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Oper. Res. Lett. 8 (1989) 67–71.
[7] T. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures, J. Global Optimization 2 (1995) 127.
[8] E.R. Gansner, E. Koutsofios, S.C. North, K.P. Vo, A technique for drawing directed graphs, IEEE Trans. Software Eng. 19 (3) (1993) 214–230.
[9] E.R. Gansner, S.C. North, K.P. Vo, DAG—a program that draws directed graphs, Software Practice Experience 18 (11) (1988) 1047–1062.
[10] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, 1997.
[11] D.S. Johnson, The NP-completeness column: an ongoing guide, J. Algorithms 3 (1982) 288–300.

[12] M. Jünger, P. Mutzel, Exact and Heuristic Algorithms for 2-Layer Straight Line Crossing Minimization, Lecture Notes in Computer Science, Graph Drawing 95, Springer, Berlin, 1995, pp. 337–348.
[13] D.E. Knuth, The Stanford GraphBase: A Platform for Combinatorial Computing, Addison-Wesley, New York, 1993.
[14] M. Laguna, T.A. Feo, H.C. Elrod, A greedy randomized adaptive search procedure for the two-partition problem, Oper. Res. 42 (4) (1994) 677–687.
[15] M. Laguna, R. Martí, GRASP and path relinking for 2-layer straight line crossing minimization, INFORMS J. Comput. 11 (1) (1999) 44–52.
[16] M. Laguna, R. Martí, V. Valls, Arc crossing minimization in hierarchical digraphs with tabu search, Comput. Oper. Res. 24 (12) (1997) 1175–1186.
[17] E. Makinen, Experiments on drawing 2-level hierarchical graphs, Internat. J. Comput. Math. 36 (1990) 175–182.
[18] R. Martí, A tabu search algorithm for the bipartite drawing problem, Eur. J. Oper. Res. 106 (1998) 558–569.
[19] K. Sugiyama, S. Tagawa, M. Toda, Methods for visual understanding of hierarchical system structures, IEEE Trans. Systems Man Cybernet. 11 (2) (1981) 109–125.
[20] V. Valls, R. Martí, P. Lino, A tabu thresholding algorithm for arc crossing minimization in bipartite graphs, Ann. Oper. Res. 63 (1996) 233–251.
[21] V. Valls, R. Martí, P. Lino, A branch and bound algorithm for arc crossing minimization in bipartite graphs, Eur. J. Oper. Res. 90 (1996) 303–319.

Update

Discrete Applied Mathematics

Volume 140, Issue 1–3, 15 May 2004, Page ix–x

DOI: https://doi.org/10.1016/S0166-218X(04)00158-1

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 140 (2004) ix–x
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

ERRATUM

Editorial Note: DAM Software Section☆

Dr. C.J. Leonard, Publishing Editor

Due to an oversight by the Publisher, the following articles have appeared in *Discrete Applied Mathematics* without any indication that these papers were supposed to be in the Mathematical Software Section of the journal. The Publisher wishes to apologise for this oversight to the authors of the articles and also to the Mathematical Software Section editors.

The construction of cubic and quartic planar maps with prescribed face degrees
Discrete Applied Mathematics, Volume 128, Issues 2–3, 1 June 2003, Pages 541–554
Gunnar Brinkmann, Thomas Harmuth and Oliver Heidemeier
X-ref: 10.1016/S0166-218X(02)00549-8

Local search algorithms for the k-cardinality tree problem
Discrete Applied Mathematics, Volume 128, Issues 2–3, 1 June 2003, Pages 511–540
Christian Blum and Matthias Ehrgott
X-ref: 10.1016/S0166-218X(02)00548-6

Heuristics and meta-heuristics for 2-layer straight line crossing minimization
Discrete Applied Mathematics, Volume 127, Issue 3, 1 May 2003, Pages 665–678
Rafael Martí and Manuel Laguna
X-ref: 10.1016/S0166-218X(02)00397-9

Fortran subroutines for computing approximate solutions of weighted MAX-SAT problems using GRASP
Discrete Applied Mathematics, Volume 100, Issues 1–2, 15 March 2000, Pages 95–113
Mauricio G.C. Resende , Leonidas S. Pitsoulis and Panos M. Pardalos
X-ref: 10.1016/S0166-218X(99)00171-7

Algorithms and codes for dense assignment problems: the state of the art
Discrete Applied Mathematics, Volume 100, Issues 1–2, 15 March 2000, Pages 17–48

☆ PII of original article S0166-218X(02)00549-8 S0166218X02005486 S0166218X02003979 S0166218X99001717 S0166218X99001729 S0166218X99000505 S0166218X99000487

0166-218X/$ - see front matter
doi:10.1016/S0166-218X(04)00158-1

Mauro Dell'Amico and Paolo Toth
X-ref: 10.1016/S0166-218X(99)00172-9

Separating lifted odd-hole inequalities to solve the index selection problem
Discrete Applied Mathematics, Volume 92, Issues 2–3, June 1999, Pages 111–134
Alberto Caprara and Juan José Salazar González
X-ref: 10.1016/S0166-218X(99)00050-5

A software package of algorithms and heuristics for disjoint paths in Planar Networks
Discrete Applied Mathematics, Volume 92, Issues 2–3, June 1999, Pages 91–110
Ulrik Brandes, Wolfram Schlickenrieder, Gabriele Neyer, Dorothea Wagner and Karsten Weihe
X-ref: 10.1016/S0166-218X(99)00048-7

Please note that submissions for the Mathematical Software Section are welcomed and should be addressed to Professors S. Martello and P. Toth at smartello@deis.unibo.it and ptoth@deis.unibo.it

Further information on submission for the Software Section and other sections can be found in the Guide for Authors at the back of this journal, or at the Elsevier Author Gateway http://authors.elsevier.com/