Arc Crossing Minimization in Hierarchical Digraphs with Tabu Search

MANUEL LAGUNA[*]

Graduate School of Business, University of Colorado, Campus Box 419, Boulder, CO 80309
Manuel.Laguna@Colorado.Edu

RAFAEL MARTÍ[**] and VICENTE VALLS[***]

Departamento de Estadística e Investigación Operativa
Facultad de Matemáticas, Universidad de Valencia
Dr. Moliner 50, 46100 Burjassot (Valencia) Spain
Marti@mac.uv.es
Vicente.Valls@uv.es

Last revision: October 21, 1996.

Abstract — Graphs are commonly used as a basic modeling tool in areas such as project management, production scheduling, line balancing, business process reengineering, and software visualization. An important problem in the area of graph drawing is to minimize arc crossings in a multi-layer hierarchical digraph. Existing solution methods for this problem are based on simple ordering rules for single layers that may lead to inferior drawings. This paper first introduces an extensive review of relevant work previously published in this area. Then a tabu search implementation is presented that seeks high-quality drawings by means of an intensification phase that finds a local optimum according to an insertion mechanism and two levels of diversification. Computational experiments with 200 graphs with up to 30 nodes per layer and up to 30 layers are presented to assess the merit of the method.

[*] Research was supported by the Modeling Group of U S WEST Advanced Technologies, 4001 Discovery Drive, Boulder, CO 80303.

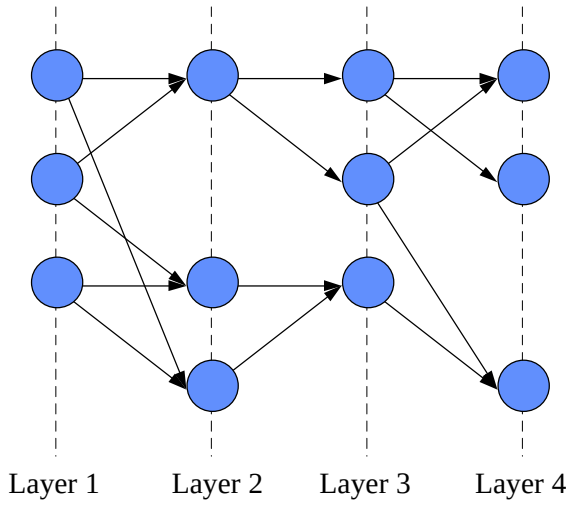
[**] Research was supported by the Conselleria de Educación y Ciencia de la Generalitat Valenciana. Spain.

[***] Research was supported by the Dirección General de Investigación Cientifica y Tecnica (DGICYT). Spain.

Introduction

The field of graph drawing has received increasing interest in the last years. The annotated bibliography on graph drawing by Di Battista, et al. (1994) has more than 300 entries. In particular, drawing directed acyclic hierarchical graphs to minimize arc crossings has received a fair amount of attention since the seminal work by Warfield (1977). Although the perception of how good a graph is in conveying information is fairly subjective, the goal of limiting the number of arc crossings is a well-admitted criterion for a good drawing. In fact, early papers in this area state that "the most crucial problem as far as readability of a graph is that of edge crossing" (Carpano 1980). Directed acyclic hierarchical graphs are also known as *layered digraphs*, *hierarchical digraphs*, or simply *hierarchies.* A hierarchical digraph is a drawing where the vertices are constrained to lie on a set of equally spaced horizontal or vertical lines called *layers* and all arcs flow in the same direction (see Figure 1). The problem of minimizing arc crossings in hierarchical graphs has been proven to be NP-hard even if there are only two layers (Garey and Johnson 1983).

Fig. 1. Hierarchical Digraph.



Warfield (1977) was the first one to propose a heuristic for the problem of arc-crossing minimization. Although his work concentrated on bipartite graphs, he developed the first crossing reduction theory for multi-layer digraphs. His procedure, which was only tested on a 21-vertex graph, introduced dummy vertices which were eliminated before drawing (a procedure that has become standard in graph drawing packages).

While Warfield's heuristic is combinatorial in nature, Delarche's (1979) is numerical as presented in Carpano (1980). This heuristic is known as the relative degree algorithm and at the time

of its development was considered well-suited to treat two-layer cycle free graphs due to its reasonable running times. The basic principle of this heuristic is that the more horizontal arcs the fewer the crossings. Hence given a permutation of vertices in layer 1, the method consists of defining a permutation in layer 2 such that these vertices are placed as much as possible directly to the right of those in layer 1 to which they are connected. Since the method can also be applied backward (i.e., fixing level 2 and defining a permutation in layer 1), the procedure iterates forward and backward until two consecutive iterations result in the same relative position of the vertices in both layers. The position of a given vertex at each iteration is calculated as the arithmetic mean of the positions of its adjacent vertices. Through extensive experimentation with real-life examples, Delarche showed that the heuristic is capable of drawing graphs with 30 to 50 percent less arc crossings when starting with an arbitrary configuration.

Eades and Kelly (1986) present four heuristics for arc crossing minimization in 2-layer graphs: *greedy insertion, greedy switching, splitting, and averaging.* The averaging heuristic is the same as the relative degree algorithm (also known as the barycentric method), while the greedy approaches use local arc crossing minimization to guide the search. In a related study, Eades and Wormald (1986) present the median heuristic. This heuristic determines the position of each vertex as the median position of its adjacent vertices. When the number of adjacent vertices is odd, the position is fully determined. However, when the number of adjacent vertices is even, a number of options are possible. In the original implementation the heuristic selects the position of the vertex on the bottom to be the median value. A weighted average has also been used, as illustrated later. Valls, et al. (1995) and Marti (1995*a*) develop tabu search procedures, which are based on the concept of tabu thresholding (Glover 1993). A computational comparison of 16 procedures on 900 randomly generated bipartite graphs is presented by Marti (1995*b*). This study shows that the procedures based on tabu search dominate all other ones in terms of solution quality at the expense of more computational time. When run-time is critical (e.g., for drawing graphs on a computer screen), the combination of the barycentric method and switching or splitting are recommended.

Additional heuristics for 2-layer graphs have been developed and tested by Makinen (1990). He considers two hybrid heuristics which operate exactly as the median method except when the vertex

being positioned has an even degree. The *average median* heuristic assigns the arithmetic mean of the positions occupied by the two middle adjacent vertices. Note that the average position may be a fraction, however, once a number has been assigned to all vertices in a layer, the vertices are ordered according to this number. The second hybrid approach referred to as the *semi median* heuristic, uses the median values to assign the position of a vertex with odd degree and the relative degree algorithm (or averaging) to assign the position of vertices with an even degree.

Carpano (1980) extended the concept of Delarche's relative degree algorithm to the general k-layer case (for $k > 2$). The graphs were represented as a permutation of vertices in each layer. Carpano realized the difficulty of the general problem, because it involved not only finding for each layer the best permutation with respect to its adjacent layers but also the effect of this choice on the configuration as a whole. The adaptation consisted of first finding the "best" permutation for layer 1 applying the relative degree algorithm considering only the vertices in layers 1 and 2. The positions of the vertices in layer 1 was determined by the outcome of this initialization step and remained fixed throughout the rest of the procedure. In layer L_i each vertex was positioned by applying the computation technique of the forward step in the relative degree algorithm considering all the predecessors located not only in layer L_{i-1} but also in layers L_{i-2}, L_{i-3}, ..., L_1.

Sugiyama, et al. (1981) proposed a so-called *barycentric method.* Their approach is similar to Carpano's in that a series of 2-layer subproblems are solved starting from an initial permutation for layer L_1. The procedure moves from "left" to "right" until the subproblem involves layers L_{k-1} and L_k, and then from "right" to "left" until the layers L_1 and L_2 are once again considered. Each subproblem i in the left-to-right sweep consists of finding the "best" permutation for layer L_i considering that the permutation of layer L_{i-1} is fixed. Similarly, each subproblem i in the right-to-left sweep consists of finding the "best" permutation for layer L_i considering that the permutation of layer L_{i+1} is fixed. Each subproblem is solved using the barycentric method originally designed for the 2-layer problem (and also known as the relative degree algorithm or averaging). In this method, the position of each vertex v in layer i is given by the arithmetic mean of the positions of the vertices in layer L_{i+1} (in a left-to-right sweep) that are adjacent to vertex v. Similarly, the average position of the adjacent vertices in layer L_{i-1} determine the barycenter of vertex v in a right-to-left sweep. The original order is preserved when two

vertices have the same barycenter, during the first phase of the procedure. A second phase is used to switch the order of vertices with equal barycenters. With the orderings determined during the second phase, the first phase is then re-applied.

An interesting variant of their method consists of calculating the barycenter of a vertex in layer L_i by regarding connected vertices in both layer L_{i-1} and L_{i+1} during both left-to-right and right-to-left sweeps. However the results of their experiments indicated that the performance of this variant was worse than the original method. The performance difference, they explain, may be due to calculating barycenters by considering vertices whose orders has not yet been improved. In the original method only layers whose orders were already improved are considered in the calculation of barycenters for the current subproblem.

Rowe, et al. (1987) employed an adaptation of Sugiyama's method within their general-purpose browser for directed graphs. In the original work by Sugiyama et al., the vertical position of a vertex was determined by either a *left-barycenter* or a *right-barycenter*. The left-barycenter of a vertex was the average position of its immediate predecessors in the previous layer. Similarly, the right-barycenter of a vertex was the average position of its immediate successors in the next layer. Ordering the vertices in a layer by the left-barycenters (right-barycenters) comes close to minimizing the number of arc crossings with respect to the previous (next) layer.

Sugiyama's method orders vertices in a layer by the left-barycenters when making a left-to-right sweep and by right-barycenters when making a right-to-left sweep. Rowe et al. noticed that this approach caused vertex positions to be unstable in some graphs. This was particularly apparent in situations where the position of a vertex determined by the left-barycenters was very different from the position determined by the right-barycenters. This caused the vertex to move back and forth between the two positions on alternate sweeps, and often the position leading to the smallest number of crossings was between the two extremes. Consequently, Rowe et al. decided to change the position-estimating function after the first two sweeps (left-to-right and right-to-left). The third and subsequent sweeps ordered vertices by considering the average of the right- and the left-barycenters of each vertex. The method terminates when all arc crossings have been eliminated or a fixed number of sweeps (usually 4) have been made.

Gansner, et al. (1988) proposed an iterative procedure based on those previously proposed by Sugiyama et al. and Rowe et al. Gansner et al. introduced a weight function to estimate positions in each layer and a local search. They called the function the *weighted median,* which according to them has two main advantages: it produces fewer crossings by reducing the effects of widely spread vertices and it is inexpensive to compute. The weighted median function is based on the *median method* originally designed by Eades and Wormald (1986) for 2-layer graphs. The weighted median is a position-estimating function that roughly works as follows. In a left-to-right sweep, a vertex in layer L_i is placed in the median position of its adjacent vertices in layer L_{i-1}, provided the number of adjacent vertices is odd. When the number of adjacent vertices is even, the weighted median is between the two middle positions. The exact position is biased toward the side (up or down) where the adjacent neighbors are more tightly packed. Positioning in a right-to-left sweep is done in a similar way (this time considering adjacent vertices in layer L_{i+1}).

Gansner's procedure starts by constructing an initial ordering within layers in a depth-first traversal of the graph beginning with the vertices in the first layer. This ordering guarantees zero arc crossings for series-parallel graphs and those graphs with an underlying tree structure. After the initial ordering is determined, the procedure iterates in order to reduce the number of arc crossings. Each iteration consists of two steps. The first step sweeps the graph either from left to right or from right to left to determine positions according to the weighted median values. After all layers have been reordered, the procedure switches the positions of two consecutive vertices in the ordering defined within a given layer if by doing so the number of crossings is reduced. The switching represents a descent method to find a local optimal ordering for layer L_i considering that the orderings of layers L_{i-1} and L_{i+1} remain fixed. This iterative process is performed a number of times, or until no discernible progress is being made towards reducing the number of arc crossings.

Tamassia, et al. (1988) take on a different approach than the previous efforts to minimize arc crossings. Given a graph, their procedure finds a maximal planar subgraph. A planar embedding for the subgraph is determined whose topology is described by a planar representation. Then, the "nonplanar" arcs (i.e., those arcs in the original graph that are not part of the planar subgraph) are successively added to the planar representation of the subgraph. These arcs are added while

minimizing at each step the number of arc crossings introduced. This procedure has been mainly used to minimize crossings in undirected and mixed graphs adopting the grid standard.

Eades and Lin (1989) and Eades and Sugiyama (1990) mention that an old graph drawing method first used by Tutte (1963) obtains similar results to the barycentric heuristic, without the layer-by-layer sweep. In Tutte's approach, later named *degree weighted barycenter* by Eades, et al. (1990), the positions of the vertices in the first and last layers are fixed. Then for each vertex in each other layer, a coordinate y is calculated as the weighted average of the y coordinates of its adjacent vertices. The weighting is computed from the indegree and the outdegree of the vertex under consideration. The y-value for each vertex can be computed by solving a system of sparse linear equations. Finally, the vertices in each layer are ordered according to y.

Gansner, et al. (1993) describe in more detail the arc-crossing minimization method proposed in Gansner, et al. (1988). They provide a pseudo-code for both the weighted median heuristic and the switching procedure. In addition they recommend to run the entire procedure twice: once for an initial ordering determined by starting with vertices in the first layer and searching out-arcs, and the second time by starting with vertices in the last layer and searching in-arcs. They believe that it is generally worth the extra cost to run the procedure twice, because it allows one to pick the better of two different solutions.

Definitions

A *hierarchical* digraph $H = (V, A, k, g)$ consists of a directed graph (V, A), a positive integer k, and, for each vertex v, an integer $g(v) \in \{1, 2, ..., k\}$, with the property that if $(u, v) \in A$ then $g(u) < g(v)$. The set $\{v : g(v) = i\}$ is the ith layer of H and is denoted by L_i, for $1 \leq i \leq k$. Note that a hierarchical digraph is acyclic.

The *span* of an arc (u, v) is $g(v) - g(u)$. Arcs of span greater than one are long, and a hierarchical graph with no long arcs is *proper*. If a hierarchy is improper, a method which is standard in the graph drawing literature, is to replace each arc of span $s > 1$ by a path of $s - 1$

dummy vertices, and thus obtaining a proper hierarchy. So without losing generality, in this paper we only consider proper hierarchies.

According to these definitions, the problem of minimizing the number of arc crossings becomes the problem of finding the optimal ordering of vertices in L_i for $1 \leq i \leq k$. Let $\Pi_i = \{\pi_i(1), \pi_i(2), ..., \pi_i(|L_i|)\}$ be an *ordering* of the vertices in layer L_i, where $\pi_i(v)$ is the position of vertex v in layer L_i. Also let the ordering of a layer be defined as $\Phi_i = \{\phi_i(1), \phi_i(2), ..., \phi_i(|L_i|)\}$, where $\phi_i(j)$ is the index of the vertex in position j (i.e., $\phi_i(\pi_i(v)) = v$). These two definitions of the ordering of a layer simplify the discussion of our procedure and its implementation details. Considering that the vertices are constrained to lie on a set of equally spaced vertical lines, and each arc is drawn as a straight line between two vertices, a *drawing* of a proper hierarchy is the set of orderings $D = \{\Pi_1, \Pi_2, ..., \Pi_k\}$ or $D = \{\Phi_1, \Phi_2, ..., \Phi_k\}$. A *crossing* is a set $\{(u, v), (w, x)\}$ of two arcs in A such that:

$$u, w \in L_i$$
$$v, x \in L_{i+1}$$
$$\pi_i(u) < \pi_i(w) \text{ and } \pi_{i+1}(v) > \pi_{i+1}(x), \text{ or}$$
$$\pi_i(u) > \pi_i(w) \text{ and } \pi_{i+1}(v) < \pi_{i+1}(x).$$

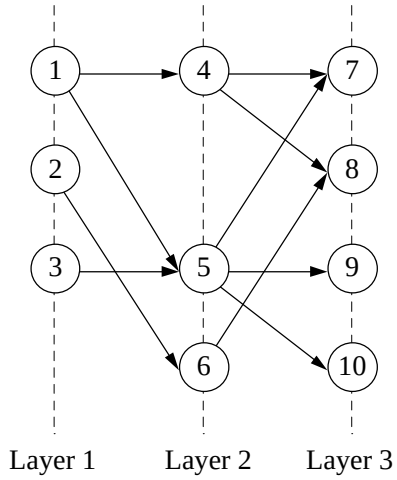
A drawing D^* is *optimal* if there is no other drawing D with fewer crossings.

Suppose that Π_i is a fixed ordering of L_i and u and v are two vertices in L_{i+1}. Then, $K_p(u,v)$ is defined as the number of crossings that arcs going into u (i.e., the arcs of the predecessors adjacent to u) make with arcs going into v when $\pi_{i+1}(u) < \pi_{i+1}(v)$. Note that given $u, v \in L_{i+1}$, the value of $K_p(u,v)$ depends only on the positions $\pi_{i+1}(u)$ and $\pi_{i+1}(v)$ and the ordering of the vertices in the ith layer. Similarly, we define $K_s(u,v)$ for a fixed ordering Π_{i+1} and vertices u and v in layer L_i, as the number of crossings that arcs going out of u (i.e., the arcs of the successors adjacent to u) make with arcs going out of v when $\pi_i(u) < \pi_i(v)$. Also note that $K_p(u,v)$ is defined for all pairs of vertices in L_i for $i = 2, ..., k$ and $K_s(u,v)$ is defined for all pairs of vertices on L_i for $i = 1, ..., k - 1$. Making $K_p(u,v) = 0 \; \forall \; u, v \in L_1$ and $K_s(u,v) = 0 \; \forall \; u, v \in L_k$, we

can define $K(u,v) = K_p(u,v) + K_s(u,v)$ as the total number of crossings that arcs incident to vertex u make with arcs incident to vertex v when u precedes v in the ordering.

In order to clarify the previous definitions, consider the hierarchical digraph in Figure 2. Following a suitable procedure to calculate the values of $K_p(u,v)$ and $K_s(u,v)$, the following values are obtained for vertices 5 and 6 (considering that the orderings in layers 1 and 3 are fixed): $K_p(5,6) = 1$, $K_s(5,6) = 2$, $K_p(6,5) = 1$, $K_s(6,5) = 1$. In Valls, et al. (1994) an efficient procedure is presented to calculate $K_p(u,v)$ and $K_s(u,v)$. Because of the symmetry between the set of values $K_s(u,v)$ for $u, v \in L_i$ and the values $K_p(u,v)$ for $u, v \in L_{i+1}$, the total number of crossings due to arcs between L_i and L_{i+1} can be calculated as follows:

Fig. 2. Arc-crossing calculation.



$$\sum_{\pi_i(u)<\pi_i(v)} K_s(u,v) \text{ or } \sum_{\pi_{i+1}(u)<\pi_{i+1}(v)} K_p(u,v). \tag{1}$$

Given a drawing $D = \{\Pi_1, \Pi_2, ..., \Pi_k\}$ of a proper hierarchical digraph, let $K(D)$ be the total number of arc crossings associated with D. Then from (1) it follows that:

$$K(D) = \sum_{i=1}^{k} \sum_{\pi_i(u)<\pi_i(v)} K_s(u,v). \tag{2}$$

Tabu Search

Tabu search is by now a well-known metaheuristic for solving hard combinatorial optimization problems. Although the origins of this technique can be traced back to almost 20 years ago (Glover 1977), the name and the methodology was officially introduced by Glover (1989). Numerous applications of this technique have appeared in the literature (see for example Glover, et al. 1993), along with tutorial papers and book chapters designed to expand the knowledge related to this methodology (Glover and Laguna 1993; Laguna 1994).

In this paper we adapt the tabu search framework to the problem of minimizing arc crossings in hierarchical proper digraphs. Our implementation is based on a simple concept: *intensify* the search by seeking optimal or near-optimal orderings of a single layer (considering that the orderings of adjacent layers are fixed) and achieve *diversification* by means of an importance sampling mechanism to select layers and a switching procedure performed on a randomly selected vertex. It is well-known within the tabu search community that successful implementations of this technique manage to create the right balance between search intensification and diversification. We seek this balance while developing a solution procedure that achieves superior solutions within a competitive running time.

Intensification Phase

The intensification phase deals with finding a local optimum with respect to an insertion move mechanism for a selected layer L_i, considering that the orderings of layers L_{i-1} and L_{i+1} are fixed. The insertion mechanism operates as follows. Suppose that the ordering of a layer L_i is Π_i when this layer is selected to initiate a new intensification phase. Then for each vertex v in L_i, we compute the change in the number of arc crossings caused by inserting v in each position $j \neq \pi_i(v)$ in the layer. (An efficient way of calculating insertion move values is given in the section of computational experiments.) If any of the $|L_i|$-1 trial insertions result in an arc crossing reduction, the insertion with the largest reduction is performed. This process starts from vertex v such that $\pi_i(v) = 1$ and proceeds in an orderly fashion until it reaches vertex w with $\pi_i(w) = |L_i|$. The entire process (or

pass) is repeated until no insertion is found that reduces the number of arc crossings in the current solution. In the case when more than one insertion results in the same maximum reduction of the number of arc crossings, the insertion that places vertex v closest to its barycenter is chosen. The intensification phase finishes with a final pass in which insertions are performed according to the barycentric criterion as long as the objective function value is not increased. The barycenter is calculated as the average position considering both layers L_{i-1} and L_{i+1}, unless $i = 1$ or k in which case only layer L_2 or L_{k-1} is respectively considered.

In sparse graphs, measuring the relative goodness of an insertion is difficult because of the large number of insertions with the same change on the objective function value. The barycentric criterion for tie breaking is particularly helpful to preserve the aggressive nature of the search during the intensification phase.

Diversification Phase

The procedure employs two levels of diversification. The first one has a narrow diversification scope and is performed via layer selection. The second level attempts to move away from the current solution structure and therefore has a broader scope.

The first level of diversification is performed after the intensification phase has been applied to all layers in the digraph with a left-to-right sweep. The sweep guarantees that the intensification phase is applied to all layers in the digraph, and is considered an intermediate phase between the application of the two diversification levels. The first diversification level is an importance sampling procedure that operates as follows. Layers are treated differently according to their level of importance. We consider that the importance of a layer increases with the degree of its vertices, because it is expected that vertices with a larger degree might produce more arc crossings than those with less number of incident arcs. Hence, the level of importance or *attractiveness* of layer L_i is given by:

$$\alpha(L_i) = \sum_{v \in L_i} d(v) \tag{3}$$

where $d(v)$ is the degree of vertex v. Then the probability of selecting layer L_i for intensification is made directly proportional to its α-value during the first-level diversification.

A tabu search mechanism is employed to avoid repeatedly selecting layers that have no opportunity to improve the best solution after an intensification phase, which would decrease the procedure effectiveness and in the limit could cause cycling. Specifically, if a layer L_i was selected and layers L_{i-1} and L_{i+1} have not changed since that intensification phase, selecting L_i again for intensification will not change the current solution (i.e., no insertion could be found that could decrease the number of arc crossings or improve the barycentric position of any vertex). In order to implement a tabu restriction that will enforce this logic, we create a structure $tabu(i)$ for $i = 1, ..., k$. Assume that at iteration *iter* layer L_i is selected for intensification. Then after the phase is completed $tabu(i) = iter$ if the number of arc crossings in the current solution has decreased, and $tabu(i) = -iter$ if the number of crossings remains the same. This updating is done during the entire first-level diversification phase, including the initial left-to-right sweep. The tabu information, however, is not used during that initial sweep. When the probabilistic selection of the first-level diversification is activated, a chosen layer L_i is allowed to enter the intensification phase if either $tabu(i+1) > abs(tabu(i))$ or $tabu(i-1) > abs(tabu(i))$.

The first-level diversification phase terminates after all layers are tabu, i.e., no layer exist that can be chosen for an intensification phase. The solution at this point is a local optimum with respect to the insertion mechanism defined above. When the first-level diversification phase terminates, a diversification at a second level begins. An iteration during the second-level diversification consists of selecting a vertex v in layer L_i for switching. Suppose that the ordering Π_i of layer L_i is such that $\pi_i(u) = \pi_i(v) - 1$ and $\pi_i(w) = \pi_i(v) + 1$, for u, v, and $w \in L_i$. We define two switch moves associated with vertex v, where Π' is the ordering after the move:

$$m^-(v) = \{\pi'_i(v) = \pi_i(u) \text{ and } \pi'_i(u) = \pi_i(v)\} \qquad \forall\, v : \pi_i(v) > 1 \tag{4}$$

$$m^+(v) = \{\pi'_i(v) = \pi_i(w) \text{ and } \pi'_i(w) = \pi_i(v)\} \qquad \forall\, v : \pi_i(v) < |L_i|. \tag{5}$$

The move evaluations are given by:

$$E(m^{-}(v)) = K(u, v) - K(v, u) \quad (6)$$

$$E(m^{+}(v)) = K(v, w) - K(w, v). \quad (7)$$

After a switch move $m^{-}(v)$ is executed on the current drawing D, the number of crossings is given by $K(D) = K(D) - E(m^{-}(v))$. A similar update is performed for a move $m^{+}(v)$. Since the objective is to minimize the total number of crossings, the larger the move value the better. The second-level diversification is performed a fixed number of iterations *maxiter*. After this phase terminates, the tabu structure is initialized, the initial left-to-right sweep is performed and the first-level diversification is started. The procedure continues until the last *nonimp* local optima found at the end of the first-level diversification fail to improve the best-known solution. A summary of the entire procedure is given in Figure 3.

Fig. 3. Pseudo-code of the tabu search procedure.

```
Generate a random initial solution.
Make best_solution equal to the current_solution.
Make lastimp equal to 0.
while (lastimp < nonimp) {
        Initial left-to-right sweep.
        while (at least one non-tabu layer exists) {
                Choose a layer (first-level diversification).
                Find the current_solution applying the
                intensification phase.
        }
        if (current_solution improves best_solution) {
                Make best_solution equal to the current_solution.
                Make lastimp equal to 0.
        } else lastimp = lastimp + 1.
        Apply second-level diversification.
}
```

Computational Experiments

Before applying an arc crossing minimization procedure to acyclic digraphs, graph drawing software performs the following two steps. First, vertices are assigned to layers, then the hierarchy is made proper. In our experiments we consider that any hierarchy can always be transformed into a proper hierarchy, and therefore we concentrate our attention to instances of hierarchical proper digraphs.

From the real-life graph examples reported in the literature, we have examined the size characteristics as determined by the number of layers, number of vertices, number of arcs, maximum number of vertices in a single layer, and the digraph density. We have used this information to determine a range for each of these parameters that would contain the aforementioned examples. The sizes of the resulting instances are feasible for computer screen drawing. However, our experiments will be extended to include larger instances that are better suited for a hard copy of the graph.

The graph generator works as follows. Given the number of layers, the number of vertices in each layer is randomly chosen between 5 and 30. For each vertex v in layer L_i, an arc to a randomly chosen vertex u in layer L_{i+1} is included. This guarantees that each vertex in layers L_1 to L_{k-1} has an outdegree of at least one. In addition, the generator checks that all vertices in the last layer have a indegree of at least one. If a vertex in layer L_k is found with an indegree of zero, an arc is added to a randomly chosen vertex in layer L_{k-1}. The next step is to check the total number of arcs in the fully connected graph with the current number of layers and vertex assignment. The total number of arcs is multiplied by the desired graph density. This gives the total number of arcs that the instance must contain. Since an initial number of arcs are already in the graph, the generator adds the difference between this number and the desired total. The additional arcs are added by randomly choosing two vertices in consecutive layers. The process is repeated until all additional arcs have been included.

We used the generator to create 180 instances with the following characteristics. For each combination of 6, 13, and 20 layers and 0.065, 0.175 and 0.3 graph densities, 20 instances were generated. The interval [0.065, 0.175] contains the density of all real-life examples found in the literature.

For the computational testing, we have employed two versions of our algorithm: TABU1 and TABU2. The TABU2 version attempts to find high-quality solutions within a reasonable computing time, while TABU1 is a fast version that tries to be competitive with respect to computational time when compared to methods based on simple ordering rules. TABU1 and TABU2 differ only in the termination criterion. The termination criterion for TABU2 is given by *nonimp* = 50. TABU1 terminates after the procedure has performed the inner while-loop in Figure 3 a total of three times. In both algorithms, the parameter *maxiter* is set to 25*|*V*| for the second-level diversification.

As far as we know, a computational testing has not been reported in the literature that compares heuristic methods for arc crossing minimization in directed acyclic graphs. However, most drawing systems employ procedures to minimize the total number of arc crossings generally based on the barycentric or median methods. Makinen (1990) compares the barycentric, median, and the hybrids average and semi median methods for 2-layer graphs, and concludes that the barycentric method provides the best results followed by the semi median method. Furthermore, Makinen shows the improvement obtained by applying a greedy switching to the solution given by the barycentric method. This idea has been further extended by means of iterating the combination barycentric or median plus greedy switching a specified number of times (see e. g., Gansner et al. 1988), and thus obtaining improved solutions.

The results obtained by combining ordering rules with exchange mechanisms seem to indicate that one of the best solution procedures could be obtained by iterating the combination of the barycentric method and a greedy switching (BC+SW) or the semi median method and a greedy switching (SM+SW). An earlier study by Gansner, et al. (1988) reports that the best results are found within a small number of iterations. Our experience has been that 6 iterations are sufficient to obtain the best results. We use these combinations, BC+SW and SM+SW, with a termination criterion of 6 iterations to compare our two tabu search versions.

Before reporting the findings of our experiments, we present implementation details that play an important role in making our procedure competitive with respect to computational efficiency. In each intensification phase on a layer L_i, the procedure must compute the change in the number of arc crossings due to the insertion of each vertex v in all positions $j \neq \pi_i(v)$. That is, a total of $|L_i|*(|L_i| - 1)$ evaluations are required during each pass of an intensification phase on layer L_i. Therefore, the intensification phase can be very time consuming if it is not carefully implemented. The evaluation scheme that we are about to describe is one of the key factors that determines the speed of our procedure. The evaluation E of the insertion of vertex $\phi_i(j)$ in position j' is given by the following two expressions:

if $j' > j$ then $$E(\phi_i(j), j') = \sum_{q=j+1}^{j'} \left[K(\phi_i(j), \phi_i(q)) - K(\phi_i(q), \phi_i(j)) \right] \quad (8)$$

if $j' < j$ then $$E(\phi_i(j), j') = \sum_{q=j-1}^{j'} \left[K(\phi_i(q), \phi_i(j)) - K(\phi_i(j), \phi_i(q)) \right] \quad (9)$$

therefore,

if $j' > j$ then $$E(\phi_i(j), j') = E(\phi_i(j), j'-1) + \left[K(\phi_i(j), \phi_i(j')) - K(\phi_i(j'), \phi_i(j)) \right] \quad (10)$$

if $j' < j$ then $$E(\phi_i(j), j') = E(\phi_i(j), j'+1) + \left[K(\phi_i(j'), \phi_i(j)) - K(\phi_i(j), \phi_i(j')) \right]. \quad (11)$$

Since $E(\phi_i(j), j) = 0$, the recursive equations (10) and (11) indicate that the efficient order to compute all possible insertions of vertex $\phi_i(j)$ is $E(\phi_i(j), j')$ for $j' = j+1, ..., |L_i|$ and $j' = j-1, ..., 1$.

It is worth mentioning that given two vertices u and v in layer L_i, the value $K(u, v)$ does not depend on the ordering Φ_i, but instead it depends on the orderings Φ_{i-1} and Φ_{i+1}. Hence, during the intensification phase on layer L_i, the values $K(u, v)$ for $u, v \in L_i$ remain constant. Our implementation, therefore, computes all $K(u, v)$ once at the beginning of each intensification phase.

The experiments reported in this section were performed on a 486-50 personal computer. All procedures were programmed in C and compiled with Microsoft Visual C++ 1.0. We have performed four different experiments. The first one consists of comparing the performance of BC+SW, SM+SW, TABU1, and TABU2 using 60 instances with density 0.065. The instances have been divided into three groups of 20 according to the number of layers (6, 13, and 20). Table 1 shows the average number of crossings obtained by each heuristic. This table shows that the tabu search versions outperform both BC+SW and SM+SW. The relative performance of BC+SW and SM+SW in this experiment agrees with earlier findings reported in the literature.

Table 1. Density 0.065: Average Number of Crossings.

Layers	BC+SW	SM+SW	TABU1	TABU2
6	431.50	442.85	377.30	331.05
13	891.65	908.80	744.45	643.00
20	1174.85	1198.70	993.30	856.75
Average	832.67	850.12	705.02	610.27

In addition to the average number of arc crossings, it is interesting to analyze the deviation of the solutions obtained by each method from the best solutions known for each problem. The best solution (BEST) of a given instance is the one with the minimum number of arc crossings from those found by the four procedures under study. The percentage deviation is calculated as (heuristic solution - BEST)/BEST. Table 2 indicates that TABU2 always finds the best solution to all problem instances. In all cases the average deviation of TABU1 is better than that of either BC+SW or SM+SW.

Table 2. Density 0.065: Average Percentage Deviation.

Layers	BC+SW	SM+SW	TABU1	TABU2
6	0.66	0.65	0.49	0.00
13	0.48	0.48	0.22	0.00
20	0.41	0.43	0.18	0.00
Average	0.52	0.52	0.30	0.00

The previous two tables have established that, when dealing with low-density graphs, the procedure based on tabu search is superior (in terms of solution quality) to those based on ordering rules combined with switching. We must now compare the computational times associated with finding solutions using each procedure. The average computational times (in seconds) are reported in Table 3. As expected, BC+SW and SM+SW are the fastest, with an average running time of 0.31 and 0.34 seconds respectively. However, the TABU1 procedure is quite competitive in time, achieving superior results in less than 1 second of computer time. Although, the average running time for TABU2 is larger, it is still quite reasonable when a higher quality of the resulting drawing is expected.

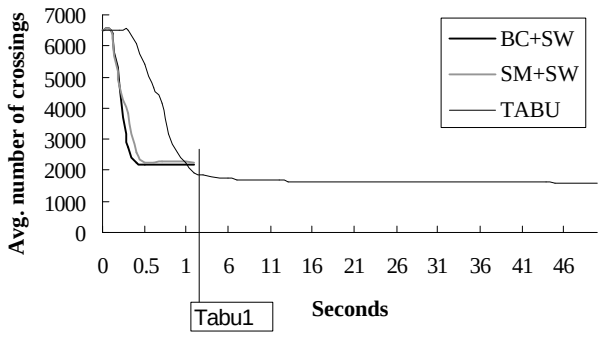
Table 3. Density 0.065: Average Running Time.

Layers	BC+SW	SM+SW	TABU1	TABU2
6	0.14	0.15	0.37	14.51
13	0.31	0.35	0.88	41.28
20	0.49	0.53	1.31	54.96
Average	0.31	0.34	0.86	36.92

Similar results are obtained when comparing all four procedures using denser graphs. Tables 4 to 6 report the testing performed on graphs with a 0.175 density, while Tables 7 to 9 summarize the results for graphs with a density of 0.3. These tables are included in the appendix. Note that the largest instances in these sets have an average of 340 vertices and 1,647 arcs.

An additional experiment is conducted with the goal of studying the progression of all procedures towards finding improved solutions. For this experiment, we generate 20 additional problem instances with density 0.1 and 30 layers. These instances have an average of 510 vertices and 838 arcs. Figure 4 shows the reduction of the average number of arc crossings as time elapses. The procedure based on tabu search is able to quickly find high quality solutions after the first three iterations (TABU1), and continues a path of improvement exhibiting diminishing returns (i.e., a reduction of the improvement margin compared to the additional computational time). For extremely short computational times (e.g., less than 0.5 seconds), BC+SW continues to dominate.

Fig. 4 Average number of crossings vs. running time.



We perform one final experiment to test the merit of employing the solution generated by the BC+SW approach as starting point for our TS procedure. Table 10 in the appendix compares the original average performance of TABU1 and TABU2 with the performance resulting form using the BC+SW starting solutions. The comparison is based on the average obtained by the different measures reported in Tables 1 to 9. This experiment revels that some improvements in average solution quality and speed are realized by initiating the tabu search from the solutions found by BC+SW. Improved outcomes are more significant when considering the application of TABU1 to low density graphs.

Conclusions

In this paper we start by reviewing the relevant work related to arc crossing minimization in hierarchical digraphs, and then we have developed an algorithm based on the tabu search. Our implementation consists of an intensification phase that seeks local optimal orderings of layers using an insertion move, and two levels of diversification. The first level of diversification is a

strategy for selecting layers for intensification, while the second one escapes local optimality by means of switching moves. We have considered running the procedure with two different termination criteria (TABU1 and TABU2).

Computational testing was performed on a set of 200 randomly generated graphs, chosen to include the range of densities and sizes previously reported in the literature. In addition, our experiments expand to graph sizes that have not been tested before, including graphs with up to 571 vertices and 2,241 arcs. Comparisons were made with procedures that have shown to be effective for arc crossing minimization, i.e., the barycentric and the semi media methods with switching.

The results of our experiments lead us to conclude that in terms of solution quality the procedures ranking is TABU2, TABU1, BC+SW, and SM+SW. This ranking holds for all the investigated densities, although the differences become smaller at higher densities. The ranking also confirms that procedures based on the barycentric method are superior than those based on the semi median method (a conclusion drawn in earlier studies). In terms of computational time, the tabu search version TABU1 is quite competitive with the procedures based on simple ordering rules plus switching. Specifically, TABU1 required a maximum running time of 1.87, 2.9, and 7 seconds for graph densities of 0.065, 0.175, and 0.3, respectively. This allows TABU1 to be considered a powerful procedure for real-time drawing (e.g., drawing on a computer screen). When higher quality drawings are important at the cost of additional computational time, TABU2 obtains solutions with fewer arc crossings with a maximum running time of 131, 157, and 209 seconds, for graph densities of 0.065, 0.175, and 0.3, respectively. This procedure may be appropriate for drawings, such as those typical of large projects or process mapping, whose printouts are used for analysis, monitoring and control.

Although the most important aesthetic criterion in graph drawing is that of minimizing arc crossings, future work could be directed to extend our procedure to handle additional criteria such as minimization of arc bends or minimization of arc lengths.

Appendix

Table 4. Density 0.175: Average Number of Crossings.

Layers	BC+SW	SM+SW	TABU1	TABU2
6	2041.15	2072.00	1961.85	1894.90
13	4842.30	4852.85	4522.65	4360.90
20	6234.10	6243.75	5750.80	5547.50
Average	4372.52	4389.53	4078.43	3934.43

Table 5. Density 0.175: Average Percentage Deviation.

Layers	BC+SW	SM+SW	TABU1	TABU2
6	0.22	0.21	0.09	0.00
13	0.13	0.12	0.04	0.00
20	0.12	0.13	0.04	0.00
Average	0.16	0.16	0.06	0.00

Table 6. Density 0.175: Average Running Time.

Layers	BC+SW	SM+SW	TABU1	TABU2
6	0.20	0.24	0.58	16.94
13	0.57	0.62	1.47	51.01
20	0.87	0.94	2.13	84.39
Average	0.54	0.60	1.39	50.78

Table 7. Density 0.3: Average Number of Crossings.

Layers	BC+SW	SM+SW	TABU1	TABU2
6	5821	5791	5706	5635
13	19827	19745	19399	19203
20	23583	23476	22915	22625
Average	16410	16337	16007	15821

Table 8. Density 0.3: Average Percentage Deviation.

Layers	BC+SW	SM+SW	TABU1	TABU2
6	0.04	0.04	0.02	0.00
13	0.03	0.03	0.01	0.00
20	0.05	0.04	0.02	0.00
Average	0.04	0.04	0.01	0.00

Table 9. Density 0.3: Average Running Time.

Layers	BC+SW	SM+SW	TABU1	TABU2
6	0.40	0.46	0.92	28.44
13	1.39	1.54	3.16	109.35
20	1.87	2.04	3.95	138.07
Average	1.22	1.35	2.68	91.95

Table 10. Comparison of average performance.

	TABU1		TABU2	
Table	**Original**	**BC+SW**	**Original**	**BC+SW**
1	705.02	668.92	610.27	600.30
2	0.30	0.20	0.00	0.00
3	0.86	0.72	36.92	33.59
4	4078.43	4028.95	3934.43	3935.42
5	0.06	0.05	0.00	0.01
6	1.39	1.16	50.78	48.83
7	16007	15979.23	15821	15820.72
8	0.01	0.01	0.00	0.00
9	2.68	2.19	91.95	88.86

References

Carpano, M-J. (1980) "Automatic Display of Hierarchized Graphs for Computer-Aided Decision Analysis," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. SMC-10, no. 11, pp. 705-715.

Delarche, M. (1979) "Quelques outils infographiques pou l'analyse structurale de systemes," Dr. Ing. thesis, Univ. Grenoble.

Di Battista, G. D., P. Eades, R. Tamassia, and I. G. Tollis (1994) "Algorithms for Drawing Graphs: an Annotated Bibliography," Department of Computer Science, Brown University.

Eades, P. and D. Kelly (1986) "Heuristics for Drawing 2-Layered Networks," *ARS Combinatoria,* vol. 21-A, pp. 89-98.

Eades, P. and X. Lin (1989) "How to Draw a Directed Graph," *Proc. IEEE Workshop on Visual Languages* (VL'89), pp. 13-17.

Eades, P., X. Lin, and R. Tamassia (1990) "An Algorithm for Drawing a Hierarchical Graph," *Proc. Second Canadian Conference on Computational Geometry* (ed. J. Urrutia), University of Ottawa, pp. 142-146.

Eades, P. and K. Sugiyama (1990) "How to Draw a Directed Graph," *Journal of Information Processing,* vol. 13, no. 4, pp. 424-437.

Eades, P. and N. C. Wormald (1986) "The Median Heuristic for Drawing Two-Layered Networks," Technical Report 69, Department of Computer Science, University of Queensland.

Gansner, E. R., E. Koutsofios, S. C. North, and K. P. Vo (1993) "A Technique for Drawing Directed Graphs," *IEEE Transactions on Software Engineering,* vol. 19, no. 3, pp. 214-230.

Gansner, E. R., S. C. North, and K. P. Vo (1988) "DAG — A Program that Draws Directed Graphs," *Software — Practice and Experience,* vol. 18, no. 11, pp. 1047-1062.

Garey, M. R. and D. S. Johnson (1983) "Crossing Number is NP-Complete," *SIAM J. Algebraic and Discrete Methods,* vol. 4, no. 3, pp. 312-316.

Glover, F. (1977) "Heuristics for Integer Programming Using Surrogate Constraints," *Decision Science,* vol. 8, pp. 156-166.

Glover, F. (1989) "Tabu Search — Part I," *ORSA Journal on Computing,* vol. 1, no. 3, pp. 190-206.

Glover, F. and M. Laguna (1993) "Tabu Search," *Modern Heuristic Techniques for Combinatorial Problems*, C. R. Reeves (Ed.), Blackwell Scientific Publications, Oxford, pp. 70-150.

Glover, F., M. Laguna, E. Taillard, D. de Werra (1993) "Tabu Search," *Annals of Operations Research*, vol. 41.

Laguna, M. (1994) "A Guide to Implementing Tabu Search," *Investigación Operativa*, vol. 4, no. 1, pp. 5-25.

Makinen, E, (1990) "Experiments on Drawing 2-Level Hierarchical Graphs," *Intern. J. Computer Math.*, vol. 36, pp. 175-181.

Marti, R (1995*a*) "An Aggressive Search Procedure for the Bipartite Drawing Problem," to appear in *Proceedings of the Metaheuristics International Conference*, Breckenridge, CO.

Marti, R. (1995*b*) "The Bipartite Drawing Problem," Technical Report, Departamento de Estadistica e Investigacion Operativa, Universidad de Valencia, Spain.

Rowe, L. A., M. Davis, E. Messinger, C. Meyer, C. Spirakis, and A. Tuan (1987) "A Browser for Directed Graphs," *Software — Practice and Experience*, vol. 17, no. 1, pp. 61-76.

Sugiyama, K., S. Tagawa, and M. Toda (1981) "Methods for Visual Understanding of Hierarchical System Structures," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-11, no. 2, pp. 109-125.

Tamassia, R., G. D. Battista, and C. Batini (1988) "Automatic Graph Drawing and Readability of Diagrams," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 61-79.

Tutte, W. (1963) "How to Draw a Graph," *Proceedings of the London Mathematical Society*, vol. 3, no. 13, pp. 743-768.

Valls, V., R. Martí, and P. Lino (1994) "A Branch and Bound Algorithm for Arc Crossing Minimization in Bipartite Graphs," to appear in *European Journal of Operational Research*.

Valls, V., R. Martí, and P. Lino (1995) "A Tabu Thresholding Algorithm for Arc Crossing Minimization in Bipartite Graphs," *Annals of Operations Research*, vol. 60.

Warfield, J. (1977) "Crossing Theory and Hierarchy Mapping," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-7, no. 7, pp. 505-523.

Arc Crossing Minimization in Hierarchical Digraphs with Tabu Search

MANUEL LAGUNA

Graduate School of Business, University of Colorado, Campus Box 419, Boulder, CO 80309
Manuel.Laguna@Colorado.Edu

RAFAEL MARTÍ and VICENTE VALLS

Departamento de Estadística e Investigación Operativa
Facultad de Matemáticas, Universidad de Valencia
Dr. Moliner 50, 46100 Burjassot (Valencia) Spain
Marti@mac.uv.es
Vicente.Valls@uv.es

Scope and Purpose — The conventional wisdom that a picture is worth more than a thousand words is particularly true when referred to graphs. In project management, for example, an activity network is more useful than a list of all activities in terms of understanding the complexity of a particular project. In business process reengineering, the use of process maps facilitate the understanding of processes and suggest opportunities for radical redesigns. The difficulty of drawing meaningful graphs (i.e., those that can be used for analysis), however, increases with the number of activities, tasks, or jobs considered within a given context. A graph drawing becomes difficult to analyze and understand when the activities (usually represented by vertices) are placed in such a way that the relationships among them (usually represented by arcs) create a large number of crossings or overlaps. Therefore, in order to create "readable" (sometimes referred to as "pretty") graph drawings, one would like to place activities in order to reduce the number of crossings caused by the connections from one activity to another. Since graphs in practical settings tend to be large, an automated procedure to deal with the arc crossing minimization problem is desirable. In this paper we present an effective procedure for the problem of minimizing arc crossings in graphs, where the effectiveness of the procedure is measured by the combination of drawing quality and speed.

Biographical Sketches

MANUEL LAGUNA is an Assistant Professor of Operations Management in the College of Business and Administration and Graduate School of Business Administration of the University of Colorado at Boulder. He received master's and doctoral degrees in Operations Research and Industrial Engineering from the University of Texas at Austin. He was the first U S WEST postdoctoral fellow in the Graduate School of Business at the University of Colorado. He has done extensive research in the interface between artificial intelligence and operations research to develop solution methods for problems in the areas of production planning and inventory control, routing and network design in telecommunications, and automated drawing. Dr. Laguna co-edited the Tabu Search volume of *Annals of Operations Research*, and he is currently editor of the *Journal of Heuristics* and *Combinatorial Optimization: Theory and Practice.* He is a member of the Institute for Operations Research and the Management Sciences, the Institute of Industrial Engineers, and the International Honor Society Omega Rho.

RAFAEL MARTI is an assistant professor in the Department of Statistics and Operations Research at the University of Valencia in Spain. He has a doctoral degree in Mathematics from the University of Valencia, for which he received a "Doctoral Thesis Award." His areas of research include project scheduling, graph drawing and the design and development of heuristic procedures in combinatorial optimization. He has worked in several research projects supported by the Spanish Government. He is a member of the Spanish Statistic and Operations Research Society (SEIO).

VICENTE VALLS received a doctoral degree in Mathematics from the University of Valencia, Spain, in 1981. He is a Professor in the Statistics and Operational Research Department of the University of Valencia. His current research interests include project scheduling, production, automatic drawing, metaheuristics and graph theory. He is currently advisory editor of the Journal of Heuristics and associate editor of TOP.