2-25-1992

Finding a Maximum-Density Planar Subset of a Set of Nets in a Channel

Ronald I. Greenberg
Rgreen@luc.edu

Jau-Der Shih

Recommended Citation

Ronald I. Greenberg and Jau-Der Shih. Finding a maximum-density planar subset of a set of nets in a channel. University of Maryland Institute for Advanced Computer Studies Technical Report UMIACS-TR-92-95 and Computer Science Technical Report CS-TR-2849, February 1992.

Finding a Maximum-Density Planar Subset of a Set of Nets in a Channel

Ronald I. Greenberg and Jau-Der Shih
Institute for Advanced Computer Studies and
Electrical Engineering Department
University of Maryland
College Park, MD 20742
rig@eng.umd.edu and jauder@eng.umd.edu

February 25, 1992

Abstract

We present efficient algorithms to find a maximum-density planar subset of n 2-pin nets in a channel. The simplest approach is to make repeated usage of Supowit's dynamic programming algorithm for finding a maximum-size planar subset, which leads to $O(n^3)$ time to find a maximum-density planar subset. But we also provide an algorithm whose running time is dependent on other problem parameters and is often more efficient. A simple bound on the running time of this algorithm is $O(n \lg n + n(t+1)w)$, where t is the number of two-sided nets, and w is the number of nets in the output. Though the worst-case running time is still $O(n^3)$, this algorithm achieves better results when either t or w is of modest magnitude. In the very special case when there are no two-sided nets, the bound stated above becomes $O(n \lg n + nw)$; this bound can also be achieved in the case of no single-sided nets. In addition, the bounds stated so far can be strengthened by incorporating into the running time the number of edges in certain interval overlap and interval containment graphs.

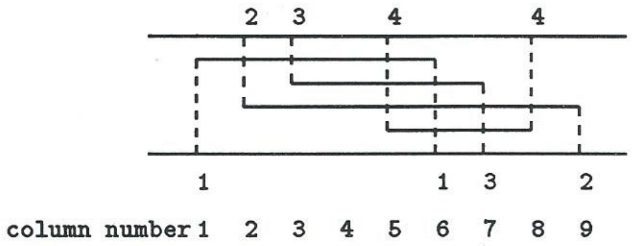

Figure 1: Shown in the figure is a channel routing problem with four nets. Here, $l_1 = 1$, $r_1 = 6$, $l_2 = 2$, $r_2 = 9$, $l_3 = 3$, $r_3 = 7$, $l_4 = 5$, and $r_4 = 8$.

1 Introduction

The channel routing problem has received a great deal of attention in VLSI layout design. Traditionally, channel routers have assumed two or perhaps three layers are available for interconnection. Most of these routers use Manhattan style routing, in which each layer is reserved primarily for wires in the horizontal direction or for wires in the vertical direction. Thus, the routing of each net is composed of horizontal and vertical wire segments in different layers. Routing nets wholely in one layer introduces difficult constraints given the limited resources of just two or three routing layers. But as chip fabrication technology has evolved to incorporate more layers of interconnect, it has become more feasible to deviate from the Manhattan routing style on some layers. If a substantial number of nets can be laid out in a coplanar fashion, it may well be worthwhile to dedicate one of the several chip layers to fully routing those nets. An example of a channel router incorporating such a strategy is MulCh [2]. MulCh tries various assignments of roles to the routing layers and assign nets to a layer or layers in a greedy-style heuristic approach. In this paper, we discuss algorithms for extracting substantial collections of coplanar nets, which may constitute a more powerful heuristic for use in multilayer channel routers.

One approach for obtaining an appropriate collection of coplanar nets is to extract the largest possible number of coplanar nets, which is equivalent to finding a maximum-cardinality independent set in a circle graph [5]. This approach, however, does not consider reducing the density of the remaining nets of the channel, which is a good measure of the difficulty of routing the remaining nets. We therefore propose to extract a set of coplanar nets of maximum density, which we refer to as an MDPS (maximum-density planar set). Then, since the MDPS may include nets from only a limited range of the channel, one can extract additional coplanar nets by continuing to work on the set of nets obtained by removing the original MDPS and the nets intersected by nets in the MDPS. One can either proceed to apply the MDPS algorithm recursively or one may settle for an application of an algorithm for maximum-cardinality independent set, yielding the planar set of largest size containing the original MDPS.

Figure 1 illustrates a channel routing problem and some of the notation to be used. There is a set N of 2-terminal nets; all the terminals lie on the two parallel "sides" of the channel. For net i, the column containing the leftmost terminal is denoted l_i, and the column containing the rightmost terminal is denoted r_i. As shown in the illustration, these terminals may lie on opposite sides or both on top or bottom. Nets with both terminals on the same side of the channel will be referred to as single-sided nets; nets with terminals on opposite sides will be referred to as two-sided nets. We will use t to denote the number of two-sided nets and w to denote the number of nets in the MDPS.

We will assume henceforth that all of the l_i and r_i values are in the range from 1 to $2n$, but we will charge $n \lg n$ time for this assumption. If the input actually comes in the form of a list of net

numbers in all the columns of the channel, and running time is measured as a function of the size of such input, we need not pay the $n \lg n$ cost. But if the input comes in the form of just the l_i and r_i values, we spend $n \lg n$ time sorting them and renumbering.

In the next two sections of this paper, we present two types of algorithms for finding an MDPS of 2-terminal nets in a channel. Section 2 reviews algorithms of Supowit [5] and Buckingham [1, pp. 119–124] for maximum-cardinality planar subset and discusses their use for finding maximum-density planar subsets. Section 3 describes an approach to finding maximum-density planar subsets based on the use of an algorithm by Masuda et. al. that can find maximum cliques of families of interval overlap or interval containment graphs [4]. The approach based on use of Supowit's algorithm, yields an overall running time of $O(n^3)$. Using Buckingham's algorithm, the running time is less on certain types of input problems. The algorithm of Section 3 greatly expands the class of input problems that can be solved more efficiently. A simple bound on the running time is $O(n \lg n + n(t+1)w)$.

2 Using Maximum Independent Set Algorithms for Circle Graphs

In this section, we review algorithms of Supowit and Buckingham for finding maximum independent sets in a circle graph and discuss their use for finding maximum-density planar subsets of the nets in a channel. In either case, the approach to finding an MDPS is simple. Just march across the channel column by column, considering at each stage only those nets that cross the current column. Among those nets, find the maximum-cardinality planar subset by using the algorithm of Supowit or Buckingham. The largest of these maximum-cardinality planar subsets is the maximum-density planar subset.

Two notations we will use are as follows. Let $S|c$ denote the nets contained in the set S that cross the column c. Note that "crossing" nets include those that terminate at the column. Also, let $MP(S)$ denote the maximum-size planar subset of the set S. Now we need only describe the methods of Supowit and Buckingham for finding $MP(N|c)$.

2.1 Supowit's Algorithm

Supowit's algorithm is a dynamic programming algorithm running in time $O(n^2)$. By executing it once on the set $N|c$ for each c, the overall running time to obtain a maximum-density planar subset is $O(n^3)$.

Supowit computes $MP(N|c)$ by computing $MP(N'|c)$ for each subset N' of N obtained by considering only the nets with terminals lying between two specified terminals in a "circular" ordering of the terminals in the channel. More precisely, let $N(i,j)$ denote the nets in N with terminals lying between the i-th and j-th terminals in a circular ordering. Then $MP(N(i,j)|c)$ can be computed from $MP(N(i,j-1)|c)$ and other already computed terms as follows. Within the circular ordering of terminals, let k denote the terminal connected to j and let $[j,k]$ denote the net with terminals j and k. If the net $[j,k]$ does not cross column c, then $MP(N(i,j)|c)$ is just $MP(N(i,j-1)|c)$. Otherwise, $MP(N(i,j)|c)$ may be obtained as the larger of $MP(N(i,j-1)|c)$ and $MP(N(i,k-1)|c) \cup [j,k] \cup MP(N(k+1,j-1)|c)$. Thus a double loop over i and j running from 1 to $2n$ suffices to compute $MP(N|c)$.

2.2 Buckingham's Algorithm

The running time of Buckingham's algorithm is dependent on the number of containments in the circle graph sequence, which we will denote as κ. That is, net $[j,k]$ is contained in $[j',k']$ if

$j' < j < k < k'$ in the circular ordering of the terminals. Then the running time of Buckingham's algorithm is $O(n+\kappa)$. When his algorithm is applied column by column the running time is $O(n^2)$ plus the sum over c the κ values for $N|c$.

Buckingham's algorithm works by making one pass over the circular ordering of terminals, finding the largest planar subset of nets with terminals lying between the beginning of the ordering and the current position. Doing this requires a recursive call upon reaching the second terminal of each net that contains other nets; the recursive call is applied to the sequence of terminals contained within that net. Analysis of this algorithm yields the $O(n+\kappa)$ running time.

3 A More Efficient Algorithm

This section gives an algorithm for finding an MDPS that is often more efficient than the algorithms in Section 2. It incorporates an algorithm by Masuda, et. al. [4] for finding maximum cliques in interval overlap graphs. (Actually, the algorithm finds the size of the maximum clique in each of the subgraphs of the interval overlap graph obtained by restricting attention to nets crossing a particular column, and this is the capability we require.) The algorithm can also be readily adapted to work with interval containment graphs, which is an additional capability we require. In this section, we begin by providing background on the algorithm of Masuda, et. al.; then we show how to use that algorithm to find an MDPS.

3.1 Overlap Graphs, Containment Graphs, and Maximum Cliques

We begin with the definition of interval overlap graphs and interval containment graphs. Let $F = \{I_1, I_2, \ldots, I_n\}$ be a family of closed intervals on the real line. A graph $G_c = (V, E)$ is called an interval containment graph for F if there is a one-to-one correspondence between V and F such that two vertices in V are adjacent to each other if and only if one of the corresponding intervals in F contains the other. (The interval $I_1 = [l_1, r_1]$ contains the interval $I_2 = [l_2, r_2]$ if $l_1 \leq l_2 \leq r_2 \leq r_1$.) A graph $G_o = (V, E)$ is called an interval overlap graph for F if there is a one-to-one correspondence between V and F such that two vertices in V are adjacent to each other if and only if the corresponding intervals in F overlap each other but neither one contains the other. (Two intervals $I_1 = [l_1, r_1]$ and $I_2 = [l_2, r_2]$ overlap each other if $l_1 \leq l_2 \leq r_1 \leq r_2$ or $l_2 \leq l_1 \leq r_2 \leq r_1$.) It is convenient also for us to be able to refer to the interval graph G_i, which is the union of G_c and G_o.

The connection between these graphs and the nets in a channel is obtained by simply viewing each net as corresponding to an interval extending from its left terminal to its right terminal. Hence, we will use sets of nets and sets of intervals interchangeably in most notations. For example, $G_o(F)$ will represent the interval overlap graph corresponding to a set of intervals F or a set of nets F; similarly $G_c(F)$ is the corresponding interval containment graph. Also, we will continue using the notation of Section 2 to refer to restrictions to a column, *e.g.,* $F|c$ will denote the intervals contained in the set F that cross column c.

The algorithm of Masuda et. al. finds, for each column c in the channel, the size of the maximum clique of $G_o(F|c)$. The running time given in [4] is $O(n_F \lg n_F + \min\{m_F, n_F w_F\})$, where n_F is the number of intervals in F, m_F is the number of edges in $G_o(F)$, and w_F is the size of the maximum clique of $G_o(F)$. But, under the assumption that no two intervals have a common right endpoint or a common left endpoint and all endpoints are between 1 and $2n_F$, the algorithm runs in time $O(n_F + \min\{m_F, n_F w_F\})$. The restriction on common endpoints will be met because of the particular classes of intervals to which we will apply the algorithm of Masuda et. al. in Section 3.2. Also the limitation on the range of the terminal positions can easily be enforced given

the assumption that the terminal positions in the entire channel were sorted up front as mentioned in Section 1. In our further description of the algorithm of Masuda et. al. and our adaptation of it, it is also important to realize that given the limited range of terminal positions, we can do such tasks as sorting a set of nets in order of left endpoint in time linear in the number of nets (by bin sorting).

A key observation of Masuda et. al. is that finding maximum cliques is equivalent to solving certain instances of the longest increasing subsequence (LIS) problem, which we review here. Let $\pi = [\pi(1), \pi(2), \ldots, \pi(s)]$ be a sequence of distinct numbers. If a subsequence $Y = [\pi(i_1), \pi(i_2), \ldots, \pi(i_t)]$ of π satisfies $\pi(i_1) < \pi(i_2) < \cdots < \pi(i_t)$, then it is called an increasing subsequence (IS). A longest increasing subsequence (LIS) of π is an IS with maximum length among all IS's of π.

The correspondence between maximum cliques and longest increasing subsequences is obtained as follows. First we sort the set of intervals $F = \{I_1, I_2, \ldots, I_n\}$ in order of ascending right endpoint and renumber the intervals so that $r_1 \leq r_2 \leq \cdots \leq r_n$. Next, create for each c, a sequence τ_c of the intervals numbers from $F|c$ sorted in ascending order of their left endpoints. Now there is a one-to-one correspondence between IS's of τ_c and cliques of $G_o(F|c)$.

The results for interval overlap graphs can be applied to interval containment graphs by simply considering longest decreasing subsequences instead of longest increasing subsequences. (Finding the maximum clique of an interval containment graph $G_c(F)$ can be done in $O(n_F \lg n_F)$ time [3], but that algorithm does not find the size of each $G_c(F|c)$, a capability we require in Section 3.2.) The algorithm of Masuda et. al. that finds all the requisite longest increasing subsequences will serve equally well to find the longest decreasing subsequences for the interval containment graph in $O(n_F + \min\{m_F, n_F w_F\})$ time once the sorting is accounted for up front. Again, n_F is the number of nets, m_F is the number of edges in the graph under consideration, and w_F is the size of the maximum clique.

3.2 The Algorithm

In this section we explain how to use the maximum clique techniques to find an MDPS of a set of nets in a channel. For simplicity, we show in detail only how to find the size of an MDPS. By keeping track of decisions made in determining the size, one can readily determine the actual set of nets in the MDPS.

Given a set of 2-pin nets in a channel, we divide the nets into four subsets *ST*(single-sided nets connected to top), *SB*(single-sided nets connected to bottom), *DR*(double-sided right going nets) and *DL*(double-sided left going nets), as shown in Figure 2.

In accordance with our earlier comments, we assume that the nets in each of the four subsets are sorted in ascending order of right endpoints. Also, define $MC(F)$ to be a maximum clique of the interval containment graph $G_c(F)$, if $F \subset ST$ or $F \subset SB$; $MC(F)$ to be a maximum clique of the interval overlap graph $G_o(F)$, if $F \subset DL$ or $F \subset DR$. In addition, define a clique of an MDPS to be a subset of an MDPS for which there exists some column that is crossed by every net in the subset.

Remark 1 *SB and ST are mutually coplanar. Furthermore, the problem of finding a maximum-density planar subset of SB or ST is equivalent to finding a maximum clique of the interval containment graph for SB or ST, respectively.*

Remark 2 *If $x \in DR$, $y \in DL$, and there exists some column c such that $l_x \leq c \leq r_x$, and $l_y \leq c \leq r_y$, then x and y must intersect each other. Thus, a maximum clique of an MDPS may contain either elements of DR or elements of DL, but not both.*

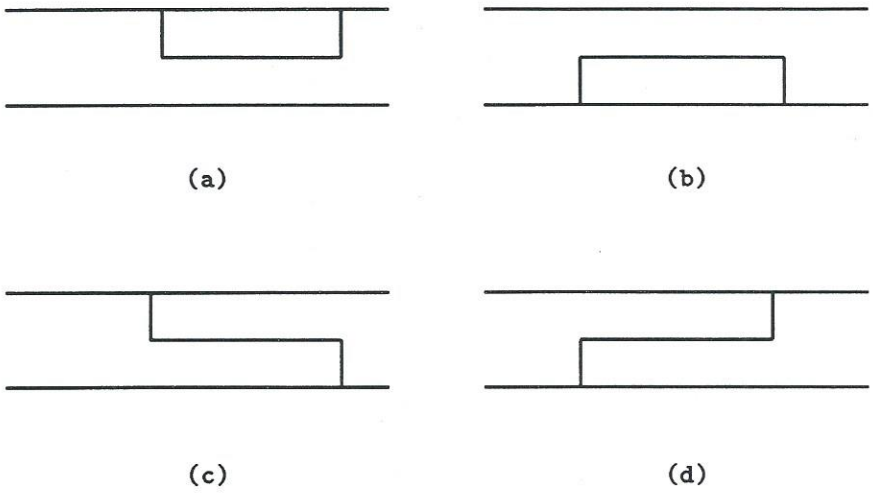

Figure 2: Four types of nets: (a) ST type (b) SB type (c) DR type (d) DL type

Remark 3 *The problem of finding an MDPS of DL or DR is equivalent to finding a maximum clique of the interval overlap graph for DL or DR, respectively.*

For simplicity, let's consider a channel with only 2-sided nets first. From Remark 2 and 3, we only have to find a maximum clique of the interval overlap graphs for DR and DL, and the larger one is an MDPS. To find either maximum clique, we need only $O(n + \min\{m, nw\})$ time, where n, m, and w represent nets, overlap edges, and output size for the corresponding graph. For simplicity, we express the overall running time (including the sorting up front) using n, m, and w as the overall nets, overlap edges, and output size, yielding $O(n \lg n + \min\{m, nw\})$ time to find an MDPS.

Now let's consider a channel with only single-sided nets. In this case, we find the maximum-density planar subsets column by column. Suppose we want to find the maximum-density planar subsets at column j. From Remark 1, it is equivalent to finding maximum cliques in the interval containment graphs for ST and SB at every column. So the algorithm works as follows: For $c = 1$ to $2n$, find $|MC(ST|c)|$ and $|MC(SB|c)|$, and then the size of the MDPS is $\max_{1 \le c \le 2n}\{|MC(ST|c)| + |MC(SB|c)|\}$. Since it takes $O(n + \min\{m, nw\})$ time to find $|MC(ST|c)|$'s and $|MC(SB|c)|$'s, the algorithm has $O(n \lg n + \min\{m, nw\})$ time complexity, with n, m, and w as before except that we are now dealing with interval containment edges.

Finally, let's consider a channel with both single-sided and double-sided nets. Because a clique of an MDPS cannot contain both elements of DL and DR, we first compute an MDPS containing no elements of DR; then compute an MDPS containing no elements of DL, the larger one is an MDPS.

Let $DL = \{\nu_1, \ldots, \nu_{n_{DL}}\}$. In what follows, we will establish some relationship among DL, ST, and SB.

Remark 4 *If ν_i is in the maximum clique of an MDPS, then for any element x of ST in the maximum clique of the MDPS, $r_x < r_{\nu_i}$, and for any element y of SB in the maximum clique of the MDPS, $l_y > l_{\nu_i}$. Moreover, if $\nu_i, \nu_{i+1}, \ldots, \nu_j$ are all in the maximum clique of an MDPS, then $r_x < \min\{r_{\nu_i}, \ldots, r_{\nu_j}\}$, and $l_y > \max\{l_{\nu_i}, \ldots, l_{\nu_j}\}$.*

Now define ST_{ν_i}, SB_{ν_i}, and DL_{ν_i} in the following way:

$$ST_{\nu_i} = \{a | a \in ST, r_a < r_{\nu_i}\}$$

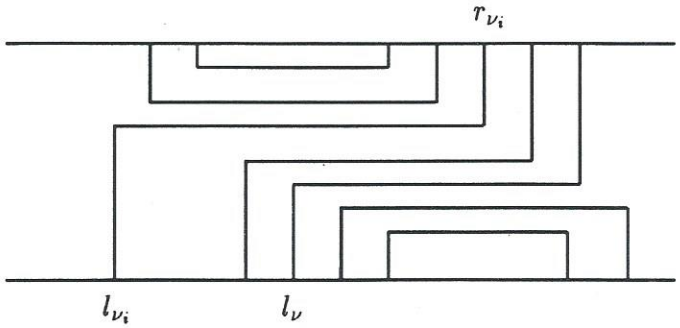

Figure 3: In this figure, $C_{i,j}$ contains 3 two-sided nets, r_{ν_i} and l_ν are shown as above.

$$\begin{aligned} SB_{\nu_i} &= \{a | a \in SB, l_a > l_{\nu_i}\} \\ DL_{\nu_i} &= \{\nu_x | \nu_x \in DL, x \geq i, l_{\nu_i} \leq l_{\nu_x} < r_{\nu_i}\} \end{aligned}$$

Also define P_i to be the maximum clique of an MDPS containing ν_i such that if there exists other $\nu_x \in MDPS$, then $i < x$. If the MDPS contains two-sided nets, then $|MDPS| = \max_{1 \leq j \leq n_{DL}}\{P_j\}$.

With these definitions we obtain the following two results from Remark 4:

Lemma 1 *If ν_i is in the maximum clique of an MDPS, then no elements of $ST - ST_{\nu_i}$ or $SB - SB_{\nu_i}$ are in that clique.* ■

Lemma 2 *Let $C_{i,j}$ be the maximum clique of an MDPS containing $MC(DL_{\nu_i}|j)$ and no other elements of DL. Then, for $l_{\nu_i} \leq j \leq r_{\nu_i}$,*

$$|C_{i,j}| = |MC(DL_{\nu_i}|j)| + \max_{l_\nu \leq k \leq r_{\nu_i}} \{|MC(ST_{\nu_i}|k)| + |MC(SB_\nu|k)|\} ,$$

where l_ν is the rightmost left endpoint of $MC(DL_{\nu_i}|j)$, and $|P_i| = \max_{l_{\nu_i} \leq j \leq r_{\nu_i}}\{|C_{i,j}|\}$.

Proof. By the construction of DL_{ν_i}, the right endpoint of ν_i is the leftmost right endpoint of all elements in DL_{ν_i}, and the left endpoint of ν_i is the leftmost left endpoint of all elements in DL_{ν_i}. Thus, ν_i is included in every $MC(DL_{\nu_i}|j)$ for $l_{\nu_i} \leq j \leq r_{\nu_i}$. (See Figure 3.) From Remark 4, only elements of ST_{ν_i} and SB_ν can be included with $MC(DL_{\nu_i}|j)$ in $C_{i,j}$. Since any two nets from different sets among DL_{ν_i}, ST_{ν_i}, and SB_ν are coplanar, we can find an MDPS at column j by finding maximum cliques of the three sets separately, which yields the expression for $|C_{i,j}|$. Finally, the expression for P_i follows because it must contain one of the $MC(DL_{\nu_i}|j)$ sets. ■

From Lemma 1 and 2, we can compute (the size of) an MDPS, using the algorithm shown in Figure 4. After we find an MDPS of nets containing only *SB*, *ST*, and *DL*, an MDPS of nets containing only *SB*, *ST*, and *DR* can be computed similarly. The larger one is an MDPS of all the nets.

For X being any of the four sets *SB*, *ST*, *DL*, or *DR*, let n_X, m_X, and w_X denote the number of nets in X, the number of edges in $G_i(X)$, and the maximum clique size for $G_i(X)$. Also let n, m, and w be the maximum over the choices for X of the corresponding parameters.

Lines 1–2 can be completed in time $O(n + \min\{m_{ST}, n_{ST}w_{ST}\} + \min\{m_{SB}, n_{SB}w_{SB}\})$, and lines 3–6 can be completed in that amount of time multiplied by n_{DL}. In the remainder of the algorithm, the running time of most steps is bounded by nn_{DL}. The n_{DL} executions of line 9 requires an additional $O(n_{DL} \min\{m_{DL}, n_{DL}w_{DL}\})$ time, but this is dominated by the time required

```
1   find |MC(ST|1)|, ..., |MC(ST|2n)|, |MC(SB|1)|, ..., |MC(SB|2n)|
2   P_0 ← max_{1≤c≤2n}{|MC(ST|c) + |MC(SB|c)|}
3   for i ← 1 to n_DL do
4       find ST_{ν_i}, SB_{ν_i}
5       find |MC(ST_{ν_i}|1)|, ..., |MC(ST_{ν_i}|2n)||MC(SB_{ν_i}|1)|, ..., |MC(SB_{ν_i}|2n)|
6   endfor
7   for i ← 1 to n_DL do
8       find DL_{ν_i}
9       find |MC(DL_{ν_i}|1)|, ..., |MC(DL_{ν_i}|2n)|
10      |C_{i,l_{ν_i}}| ← 1 + max_{l_{ν_i}≤k≤r_{ν_i}}{|MC(ST_{ν_i}|k)| + |MC(SB_{ν_i}|k)|}}
11      for j ← l_{ν_i} + 1 to r_{ν_i} do
12          if |MC(DL_{ν_i}|j)| > |MC(DL_{ν_i}|j − 1)|
13          then
14              Let ν ∈ DL be such that l_ν = j.
15              |C_{i,j}| ← max{|C_{i,j−1}|, |MC(DL_{ν_i}|j)| + max_{l_ν≤k≤r_{ν_i}}{|MC(ST_{ν_i}|k)| + |MC(SB_ν|k)|}}
16          else
17              |C_{i,j}| ← |C_{i,j−1}|
18          endif
19      endfor
20      P_i ← |C_{i,j}|
21  endfor
22  |MDPS| ← max{P_0, max_{1≤i≤n_DL}{P_i}}
```

Figure 4: This algorithm uses interval containment graphs and interval overlap graphs to compute an MDPS.

by line 15. The test in line 12 ensures that line 15 is executed at most $\min\{m_{DL}, n_{DL}w_{DL}\}$ times; each execution requires $O(n)$ time. Putting all the terms together, the running time of the procedure in Figure 4 is

$$O(n(n_{DL} + \min\{m_{DL}, n_{DL}w_{DL}\}) + (n_{DL} + 1)(\min\{m_{SB}, n_{SB}w_{SB}\} + \min\{m_{ST}, n_{ST}w_{ST}\})) .$$

Finally, the overall time to find an MDPS can be obtained by incorporating terms based on DR and the $n \lg n$ sorting time performed up front.

Under a variety of conditions in which the sets of the four types of nets have small numbers of edges in the corresponding interval graph or small clique sizes, the running time is substantially superior to the naive $O(n^3)$ approach. Even if we simplify the bound greatly, we can see substantial opportunity for the running time to be $o(n^3)$. In particular, we can express the overall running time to find an MDPS as $O(n(\lg n + t + \min\{m, tw\}) + \min\{m, nw\})$, where t is the number of two-sided nets in the channel. If we simplify further by removing the minimization with respect to m and note that w is always at least 1, we obtain a bound of $O(n(\lg n + tw + w)) = O(n \lg n + n(t+1)w)$.

4 Conclusion

In addition to providing a naive $O(n^3)$ method for finding a maximum-density planar subset of nets in a channel, we have presented a method that runs in time $o(n^3)$ for many types of input problems. For example, superior running time is obtained if the number of two-sided nets or the number of nets in the output is $o(n)$. The running time can also be kept to $o(n^3)$ under more limited but different conditions by applying Buckingham's maximum planar subset algorithm column by column. These two algorithms actually could be interleaved to obtain $o(n^3)$ running time in the union of the favorable cases for the two algorithms. An open area for investigation is to obtain a simpler algorithm that runs in $o(n^3)$ time in all of these cases or to obtain an algorithm that has $o(n^3)$ running time on all n-net problems.

References

[1] Mark A. Buckingham. Circle graphs. Technical Report NSO-21, Courant Institute of Mathematical Sciences, Computer Science Department, New York University, October 1980.

[2] Ronald I. Greenberg, Alexander T. Ishii, and Alberto L. Sangiovanni-Vincentelli. MulCh: A multi-layer channel router using one, two, and three layer partitions. In *IEEE International Conference on Computer-Aided Design*, pages 88–91. IEEE Computer Society Press, 1988.

[3] S. Masuda and K. Nakajima. Optimal algorithms for interval containment graphs. In *Proc. 29th Midwest Symposium on Circuits and Systems*, pages 431–434, Lincoln, NE, August 1986.

[4] S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Efficient algorithms for finding maximum cliques of an overlap graph. Technical Report UMIACS-TR-86-31.1, University of Maryland Institute for Advanced Computer Studies, College Park, January 1988.

[5] K. J. Supowit. Finding a maximum planar subset of a set of nets in a channel. *IEEE Trans. Computer-Aided Design*, 6:93–94, January 1987.