Segmented Channel Routing

Jonathan Greene†, Vwani Roychowdhury‡, Sinan Kaptanoglu†, Abbas El Gamal‡

Actel Corporation, Sunnyvale, CA 94086 †
Information Systems Lab, Stanford University, Stanford, CA 94305 ‡

Abstract: Routing channels in a field-programmable gate array contain predefined wiring segments of various lengths. These may be connected to the pins of the gates or joined end-to-end to form longer segments by programmable switches. The segmented channel routing problem is formulated, and polynomial time algorithms are given for certain special cases. The general problem is NP-complete, but it can be adequately solved in practice. Experiments indicate that a segmented channel with judiciously chosen segment lengths may near the efficiency of a conventional channel.

1. Introduction

In the classic channel routing problem (see [1]), the channel may be freely customized (by the appropriate mask) to route the desired connections. This paper considers the more restricted case where the routing is constrained to use fixed wiring segments of predetermined lengths and locations within the channel. The segments may be interconnected by programmable *switches*, which provide the only means of customization. This *segmented channel routing* problem arises in the context of field-programmable gate arrays (FPGAs) [2] [3], and possibly in other contexts as well. (E.g., the "channel" may be a bus connecting processors.)

In particular, our model is motivated by the Actel FPGA [3], which has an architecture similar to conventional (mask-programmed) channeled gate arrays: rows of cells (logic modules) separated by routing channels (Fig. 1). The tracks in the routing channels contain wiring segments of various lengths. The inputs and outputs of the modules each connect to a dedicated vertical segment. Programmable switches are implemented by "anti-fuses" [4] located at each crossing of vertical and

The work of A. El Gamal and V. Roychowdhury was partially supported under contract J-FBI-89-101.

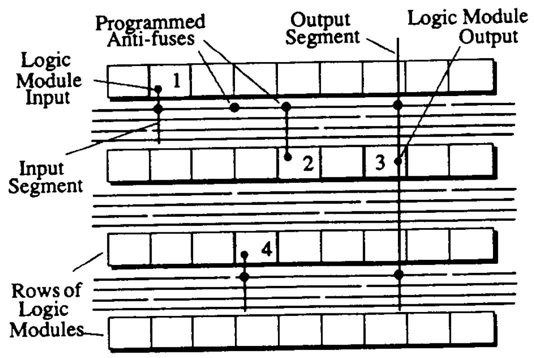

Fig. 1: FPGA Routing Architecture. ● denotes a programmed anti-fuse. Unprogrammed anti-fuses are omitted for clarity.

horizontal segments and also between pairs of adjacent horizontal segments in the same track. By programming an anti-fuse, a low resistance path is created between the two crossing or adjoining segments. A typical net routing is shown in Fig. 1. The vertical segment connected to the output of module 3 is connected by a programmed anti-fuse to a horizontal segment, which in turn is connected to the input of module 4 through another programmed anti-fuse. In order to reach the inputs of modules 1 and 2, two adjacent horizontal segments are connected to form a longer one.

The process of automatically mapping a design into such an FPGA is similar to that for a gate array: placement, global routing to subdivide each net into connections to be routed in each channel, and finally channel routing.

The choice of the wiring segment lengths in a segmented channel is driven by tradeoffs involving the number of tracks, the resistance of the switches, and the capacitances of the segments. Fig. 2 illustrates this.

Fig. 2A shows a set of connections to be routed. With the complete freedom to configure the wiring afforded by mask programming, the "left edge algorithm" [5] will always find a routing using a number of tracks equal to the density of the connections (Fig. 2B). This occurs since there are no "vertical constraints" [1]

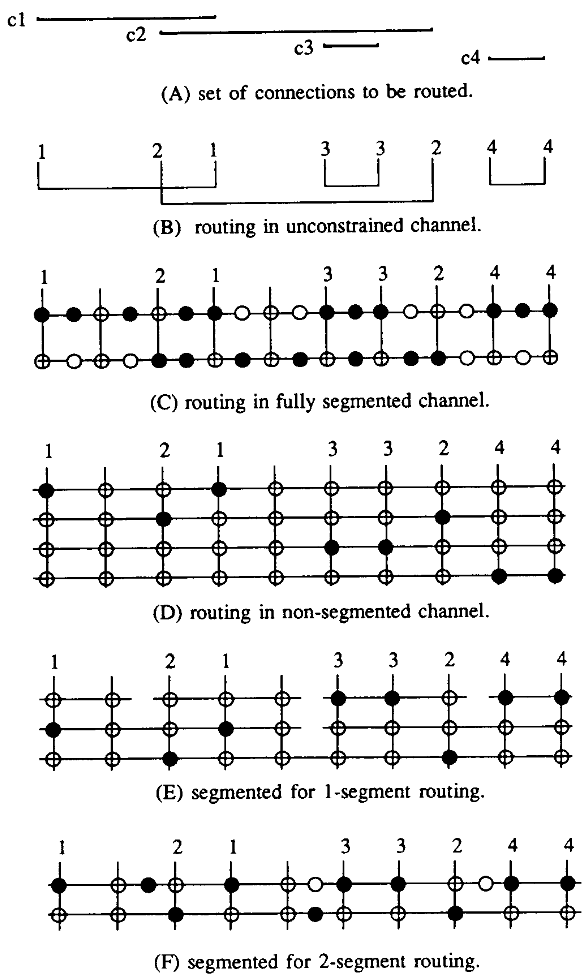
c1
c2
c3
c4
(A) set of connections to be routed.
1 2 1 3 3 2 4 4
(B) routing in unconstrained channel.
1 2 1 3 3 2 4 4
(C) routing in fully segmented channel.
1 2 1 3 3 2 4 4
(D) routing in non-segmented channel.
1 2 1 3 3 2 4 4
(E) segmented for 1-segment routing.
1 2 1 3 3 2 4 4
(F) segmented for 2-segment routing.

Fig. 2: examples of channel routing.
○ denotes open switch, ●closed switch.

in the problems we consider.

In an FPGA, achieving this complete freedom would require switches at every cross point. Furthermore, switches would be needed between each two cross points along a wiring track so that the track could be subdivided into segments of arbitrary length (Fig. 2C). Since all present technologies offer switches with significant resistance and capacitance, this would cause unacceptable delays through the routing. Another alternative would be to provide a number of continuous tracks large enough to accommodate all nets (Fig. 2D). Though the resistance is limited, the capacitance problem is only compounded, and the area is excessive.

A segmented routing channel offers an intermediate approach. The tracks are divided into segments of varying lengths (Fig. 2E), allowing each connection to be routed using a single segment of the appropriate size. Greater routing flexibility is obtained by allowing limited numbers of adjacent segments in the same track to be joined end-to-end by switches (Fig. 2F). Enforcement of simple limits on the number of segments joined or their total length guarantees that the delay will not be unduly increased.

Succeeding sections of the paper discuss the segmented routing problem for the models of Figs. 2E and 2F. The problem is defined precisely in Section 2. Polynomial time routing algorithms are given for two special cases: (a) where each connection occupies only one segment (Fig. 2E); and (b) where all tracks are segmented identically. The general segmented routing problem is NP-complete. Section 3 describes a general algorithm that is linear in the number of connections, though exponential in the number of tracks. If there are many tracks, but they are segmented in a fixed number of ways, the algorithm becomes polynomial in the number of tracks as well. Section 4 gives experimental results. Surprisingly, we find that the considerable restrictions of segmented routing do not greatly increase the number of tracks required.

2. Definitions and Special Cases

A segmented channel routing problem, as depicted in Fig. 3, consists of a set of M connections and a set of T tracks. The tracks are numbered from 1 to T. Each track extends from column 1 to column N, and is divided into a set of contiguous segments separated by switches. The switches are placed between two consecutive columns.

For each segment s, we define $left(s)$ and $right(s)$ to be the leftmost and rightmost column in which the segment is present. $1 \le left(s) \le right(s) \le N$. Each connection c_i, $1 \le i \le M$, is characterized by its leftmost and rightmost column: $left(c_i)$ and $right(c_i)$. Without loss of generality, we assume throughout that the connections have been sorted so $left(c_i) \le left(c_j)$ for $i<j$.

A connection c may be *assigned* to a track t, in which case the segments in track t that are present in the columns spanned by the connection are considered *occupied.* More precisely, a segment s in track t is occupied by the connection c if $right(s) \ge left(c)$ and $left(s) \le right(c)$. In Fig. 3 for example, connection $c1$ would occupy segments $s11$ and $s12$ in track 1 or segment $s21$ in track 2. A *routing* of a set of connections consists of an assignment of each connection to a track such that no segment is occupied by more than one connection. A *K-segment routing* is a routing that satisfies the additional requirement that each connection occupies at most K segments. Define the following problems:

Problem 1: Unlimited Segment Routing. Given a set of connections and a segmented channel, find a routing.

Problem 2: K-Segment Routing. Given a set of connections and a segmented channel, find a K-segment routing.

It is often desirable to determine a routing that is optimal with respect to some criterion such as delay. We may thus specify a weight $w(c, t)$ for the assignment of connection c to track t, and define:

Problem 3: Routing Optimization. Given a set of connections and a segmented channel, find a routing which assigns each connection c_i to a track t_i such that $\sum_{i=1}^{M} w(c_i, t_i)$ is minimized.

For example, a reasonable choice for $w(c, t)$ would be the sum of the lengths of the segments occupied when connection c is assigned to track t. Note also that with appropriate choice of $w(c, t)$, Problem 3 subsumes Problem 2.

Special Case 1: 1-Segment Routing

If we restrict consideration to 1-segment routings, Problem 2 may be solved by the following greedy algorithm. The connections are assigned in order of increasing left ends as follows. For each connection, find the set of tracks in which the connection would occupy one segment. Eliminate any tracks where this segment is already occupied. From among the remaining tracks, choose one where the occupied segment's right end is farthest to the left, and assign the connection to it. In the example of Fig. 3, the algorithm assigns $c1$ to $s21$, $c2$ to $s31$, $c3$ to $s12$, $c4$ to $s13$, and $c5$ to $s23$. It is easily shown that if some connection cannot be assigned to any track, then no complete routing is possible. The time required is $O(MT)$.

Problem 3 may be solved efficiently by bipartite matching for 1-segment routing. Fig. 4 shows the graph corresponding to the routing problem in Fig. 3. The left side has a node for each connection and the right side a node for each segment. An edge is present between a connection and a segment if the connection can be assigned to the segment's track. The weight $w(c, t)$ is assigned to the edge between connection c and a segment in track t. A minimum-weight matching indicates an optimal routing. The time required using the best known matching algorithm (see [6]) is $O(V^3)$, where $V \le M+NT$ is the number of nodes.

Special Case 2: All Tracks Identically Segmented

If all tracks are identically segmented (i.e., the locations of the switches are the same in every track), then Problems 1 and 2 can be solved by the left edge algorithm in time $O(MT)$. Assign the connections in order of increasing left ends as follows. For each connection, assign the connection to the first track in which none of the segments it would occupy are yet occupied.

Note that the density of the connections does not provide an upper bound on the number of tracks required for routing (as is the case for conventional routing when the left edge algorithm is used in the absence of vertical constraints). However if, prior to computing the density, each connection is extended at both ends until a column adjacent to a switch is reached, then the density will be a valid upper bound.

3. An Algorithm for Segmented Routing

Unfortunately, the general segmented routing problem is computationally quite difficult.

Theorem 1: Problem 1, and Problem 2 for K>1, are NP-complete.

Segmented routing can be shown to be equivalent to the NP-complete problem of "numerical matching with targeted sums." (See [7] for a description of this problem). Details of the proof are omitted; they will appear in a subsequent paper.

Although segmented routing is NP-complete, we describe below an algorithm that finds a routing in time linear in M (the number of connections) when T (the number of tracks) is fixed. This is of interest since T is often substantially less than M. The algorithm may also be quite efficient when there are many tracks, but they are segmented in a limited number of ways (see Theorem 4 below). The algorithm first constructs a data structure called an *assignment tree* and then reads a valid routing from it. The same algorithm applies to both Problem 1 and 2, though with different time and memory bounds. It can also be extended to Problem 3.

3.1. Frontiers and the Assignment Tree

Given a valid routing for connections c_1 through c_i, it is possible to define a *frontier* which constitutes sufficient information to determine how the routing of $c_1 \cdots c_i$ may be extended to include an assignment of c_{i+1} to a track such that no segment occupied by any of c_1 through c_i will also be occupied by c_{i+1}. Figure 5 shows an example of a frontier. It will be apparent that c_{i+1} may be assigned to any track t in which the frontier has not advanced past the left end of c_{i+1}. This is the case for $c3$ in the figure.

More precisely, given a valid routing of $c_1 \cdots c_i$, $1 \le i < M$, define the frontier $\mathbf{x}$ to be a T-tuple $(\mathbf{x}[1], \mathbf{x}[2], ..., \mathbf{x}[T])$ where $\mathbf{x}[t]$ is the leftmost unoccupied column in track t at or to the right of column $left(c_{i+1})$. (A column in track t is considered unoccupied if the segment present in the column is not occupied.) The frontier is thus a function $\mathbf{x} = F_i(t_1, \ldots, t_i)$ of the tracks $t_1 \cdots t_i$ to which $c_1 \cdots c_i$ are respectively assigned. For i=0, let $\mathbf{x} = F_0$, where $F_0[t] = left(c_1)$ for all t. For $i=M$, let $\mathbf{x}$ =

F_M, where $F_M[t] = N+1$ for all t.

Next, we describe a graph called the assignment tree which is used to keep track of partial routings and the corresponding frontiers. A node at level i, $1 \le i < M$, of the assignment tree corresponds to a frontier resulting from some valid routing of c_1 through c_i. Level 0 of the tree contains the root node, which corresponds to F_0. If a complete valid routing for $c_1 \cdots c_M$ exists, then level M of the tree contains a single node corresponding to F_M. Otherwise, level M is empty.

The assignment tree is constructed inductively. Given level $i \ge 0$ of the tree, construct level $i+1$ as follows. (For convenience, we identify the node by the corresponding frontier.)

```
For each node $\mathbf{x}_i$ in level $i$ {
    For each track $t$, $1 \le t \le T$ {
        If $\mathbf{x}_i[t]=left(c_{i+1})$ {
            /* $c_{i+1}$ can be assigned to track $t$. */
            Let $\mathbf{x}_{i+1}$ be the new frontier after
              $c_{i+1}$ is assigned to track $t$.
            If $\mathbf{x}_{i+1}$ is not yet in level $i+1$ {
                Add node $\mathbf{x}_{i+1}$ to level $i+1$.
                Add an edge from node $\mathbf{x}_i$ to
                  node $\mathbf{x}_{i+1}$. Label it with $t$.
            }
        }
        Else {
            /* $\mathbf{x}_i[t]>left(c_{i+1})$ so $c_{i+1}$ cannot be
              assigned to track t. */
            Continue to next track $t$.
        }
    }
}
```

If there are no nodes added at level $i+1$, then there is no valid assignment of c_1 through c_{i+1}.

Searching for the node $\mathbf{x}_{i+1}$ in level $i+1$ can be done in $O(T)$ time using a hash table. Insertion of a new node in the table likewise requires time $O(T)$.

If there are a maximum of L nodes at each level, then construction of the entire assignment tree requires time $O(MLT^2)$. Once the assignment tree has been constructed, a valid routing may be found by tracing a path from the node at level M back to the root, reading the track assignment from the edge labels. (If there is no node at level M, then no complete valid assignment exists.) This takes only $O(M)$ time, so the overall time for the algorithm is $O(MLT^2)$. The memory required to store the assignment tree is $O(MLT)$.

A minor change allows us to solve the optimization problem as well. Each edge is labeled with the weight $w(c, t)$ of the corresponding assignment. Each node is labeled with the weight of its parent node plus the weight of the incoming edge. The algorithm is modified as follows. If a search in level $i+1$ finds that the new node $\mathbf{x}_{i+1}$ already exists, we examine its weight relative to the weight of node $\mathbf{x}_i$ plus $w(c_{i+1}, t)$. If the latter is smaller, we replace the edge entering $\mathbf{x}_{i+1}$ with one from $\mathbf{x}_i$ and update the weights accordingly. Thus the path traced back from the node at level M will correspond to a minimal weight routing. The order of growth of the algorithm's time remains the same, as does that of its memory.

3.2. Analysis for K-Segment Routing

The following theorem shows that for K-segment routing, $L \le (K+1)^T$, so that the time to construct the assignment tree and find an optimal routing is $O(MT^2(K+1)^T)$ and the memory required is $O(MT(K+1)^T)$.

Theorem 2: for K-segment routing, the number of distinct frontiers that may occur for some valid routing of c_1 through c_i is at most $(K+1)^T$.

Proof: Let $l=left(c_{i+1})$ and consider track t. Since the connections are sorted by increasing left edge, at most one connection from among $c_1 \ldots c_i$ may occupy track t in columns at or to the right of column l. Such a connection may occupy track t rightward through the segment appearing in column l, or through that segment plus the next one, or possibly as far as the K^{th} segment at or to the right of column l. Of course it is also possible that no connection from among c_1 through c_i occupies the segment in column l of track t. Thus there are only $K+1$ possible locations for the frontier $\mathbf{x}[t]$ in track t, and at most $(K+1)^T$ possible values for the frontier $\mathbf{x}$ overall. □

3.3. Analysis for Unlimited Segment Routing

The following theorem shows that for unlimited segment routing, $L \le 2\,T!$, so that the time to construct the assignment tree and find an optimal routing is $O(M\;T^2\;T!)$ and the memory required is $O(M\;T\;T!)$.

Theorem 3: for unlimited segment routing, the number of distinct frontiers that may occur for some valid assignment of c_1 through c_i is at most $2\,T!$.

Proof: Let $l=left(c_{i+1})$. Let d be the number of connections among c_1 through c_i that are present in column l. Since the assignment of c_1 through c_i determining the frontier must be a valid one, we know that $d \le T$. The d connections can be assigned to d of the T tracks in $T!/(T-d)!$ ways. Once we have assigned a connection to a track t, the value of $\mathbf{x}[t]$ in that track is determined. For each of the remaining $(T-d)$ tracks, there are only two alternatives:

- The track t may be unoccupied in column l, in which case $\mathbf{x}[t]=l$.
- The track may be occupied in column l by some connection c with $right(c) < l$, up to the first switch to the right of column l. In this case, $\mathbf{x}[t]$ is the

column just to the right of this switch, regardless of which such connection c is involved.

Thus the number of possible frontiers is at most $2^{(T-d)}\, T!/(T-d)! \le 2\, T!$. □

3.4. Case of Many Tracks of a Few Types

Suppose the T tracks fall into two types, with all tracks of each type segmented identically. Then two frontiers that differ only by a permutation among the tracks of each type may be considered equivalent for our purposes in that one frontier can be a precursor of a complete routing if and only if the other can. Thus we can restrict consideration to only one of each set of equivalent frontiers, and strengthen the result of Theorem 2 as follows.

Theorem 4: Suppose there are T_1 tracks segmented in one way, and $T_2 = T - T_1$ segmented another way. The number of distinct frontiers $\mathbf{x}$ that may occur for some valid K-segment routing of c_1 through c_i, and that satisfy $\mathbf{x}[t] \le \mathbf{x}[t']$ for all $t < t'$ with tracks t and t' of the same type, is $O((T_1 T_2)^K)$.

Proof: As in theorem 2, there are at most $K+1$ possible values for $\mathbf{x}[t]$. Due to the inequality restriction (which eliminates all but one member of each set of equivalent frontiers), the number of possible frontiers is at most:

$$\begin{bmatrix} T_1+K \\ T_1 \end{bmatrix} \times \begin{bmatrix} T_2+K \\ T_2 \end{bmatrix}$$

which, for large T_1 and T_2, is $O((T_1 T_2)^K)$. □

It follows that a K-segment routing may be found in time $O(M(T_1T_2)^K T^2)$, and memory $O(M(T_1T_2)^K T)$.

The result of Theorem 4 may easily be generalized to the case of j types of tracks, in which case the time is $O(M(T_1 \cdots T_j)^K T^2)$, and the memory is $O(M(T_1 \cdots T_j)^K T)$.

4. Experimental Results

In this section we report the results of some experiments with segmented channel routing.

Channel segmentations were designed by a combination of trial-and-error and human judgement. (We have no analytic procedure at this time). Two segmentations, each with 32 tracks and 40 columns, were created. "Segmentation 1" and "Segmentation 2" are intended for one- and two-segment routing, respectively. They were each tuned to achieve the greatest likelihood of complete routing of randomly chosen sets of connections under the corresponding segment limitation. The distribution of connections was derived from actual placements of 510 channels from 34 designs, and gives the probability of occurrence as a function of the length and starting point of the connection.

The one-segment routing problem was solved by the algorithm described for Special Case 1. For the two-segment case, various heuristics were used to prune the assignment tree. In either case, computation of a routing for one channel takes under three seconds on a three-MIPs workstation. Although the design of the segmentations and the two-segment routing methods are probably not optimal, they do provide valid lower bounds on what can be achieved with segmented routing.

For each segmentation, a series of trials were made as follows. First, a set of connections was chosen from the distribution, and the density of the set was determined. (The number of connections in each set is varied so as to obtain a range of densities.) Then an attempt was made to route the connections in Segmentation 1 or Segmentation 2. We also attempted one-segment routings in Segmentation 2. The results are shown in Fig. 6.

In a conventional channel, any set of connections can be routed in a number of tracks equal to the density by the left edge algorithm. Thus problems with densities not exceeding the 32 tracks provided would always route. For the one-segment case (using Segmentation 1), a high probability of successful routing is achieved when the density is about 12 less than the number of tracks. However, allowing 2-segment routing (using Segmentation 2) approaches within 3-4 tracks of the conventional case.

The rather poor results for one-segment routing using Segmentation 2 as opposed to Segmentation 1 demonstrate the importance of tuning the segmentation for the routing algorithm. Our experience shows the importance of tuning for the statistics of the connection lengths, as well.

5. Conclusions

The experiments of Section 4 show that a segmented channel can generally accommodate connections using only a few more tracks than a conventional channel would. Polynomial time segmented routing algorithms have been presented for certain special cases. The general problem is NP-complete, but it can be adequately solved in practice with reasonable CPU time by heuristics. Segmented channel routing has thus found practical use in FPGAs, and it is possible that it will play as important a role there as conventional channel routing has in mask-programmable arrays.

Our results were obtained using ad-hoc designs for the channel segmentation. More analytic solutions to the design problem would be of interest.

References

[1] M. Lorenzetti and D. S. Baeder. "Routing." Chapter 5 in *Physical Design Automation of VLSI Systems*, B. Preas and M. Lorenzetti, eds. Benjamin/Cummings, 1988.

[2] H. Hsieh, et al. "A Second Generation User Programmable Gate Array." *Proc. Custom Integrated Circuits Conf.*, May 1987, pp. 515-521.

[3] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-Ayat, and A. Mohsen. "An Architecture for Electrically Configurable Gate Arrays." *IEEE J. Solid-State Circuits*, Vol. 24, No. 2, April, 1989, pp. 394-398.

[4] E. Hamdy, et al. Dielectric Based Antifuse for Logic and Memory ICs. *IEDM Tech. Digest*, San Francisco, CA, 1988, pp. 786-789.

[5] A. Hashimoto, J. Stevens. "Wire Routing by Optimizing Channel Assignment Within Large Apertures." *Proc. 8th IEEE Design Automation Workshop*, 1971.

[6] C. H. Papadimitrou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Inc., New Jersey, 1982.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability--A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

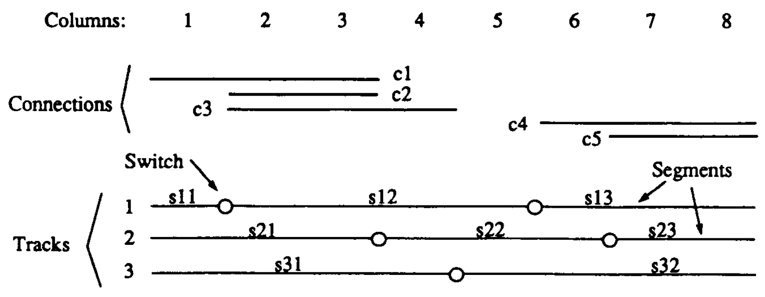

Fig. 3: An instance of the segmented routing problem. M=5, T=3, N=8. Connections: c1, c2, c3, c4, c5. Segments: s11, s12, s13, s21, s22, s23, s31, s32.

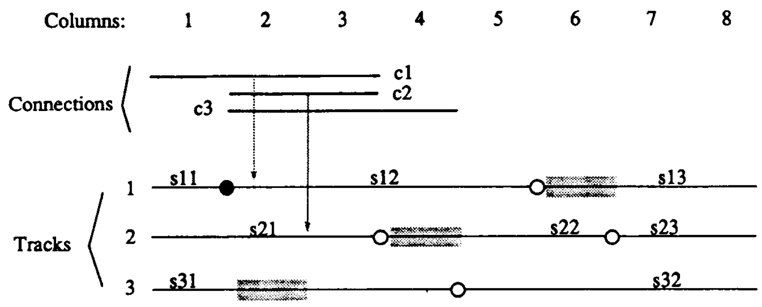

Fig. 5: a frontier for the example of Fig. 3. Shaded area marks the position of the frontier in each track after assignment of c1 to s11 and s12, and of c2 to s21. The frontier is x=[6,4,2]. The switch between s11 and s12 is considered programmed.

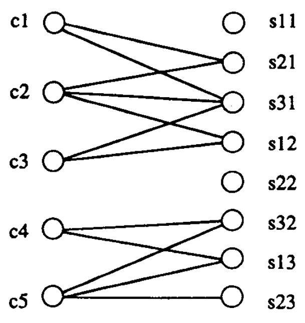

Fig. 4: Bipartite graph for 1-segment routing of the problem in Fig. 3.

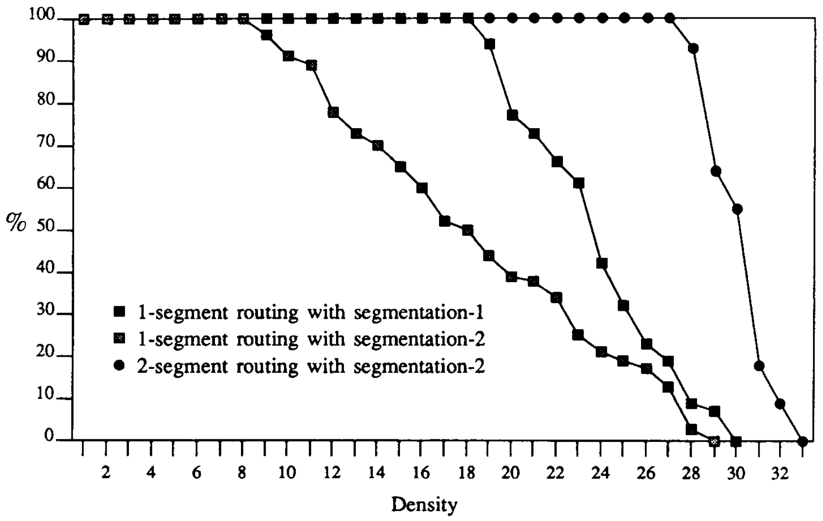

Fig. 6: Probability of Routing Success vs. Density