RC 12806 (#57605) 6/1/87
Computer Science 11 pages

Multiterminal Global Routing: A Deterministic Approximation Scheme[1]

Prabhakar Raghavan
IBM T. J. Watson Research Center, P. O. Box 218,
Yorktown Heights, NY 10598.

Clark D. Thompson
Department of Computer Science,
University of Minnesota, Duluth, MN 55812.

ABSTRACT

We consider the problem of routing multiterminal nets in a two-dimensional gate-array. Given a gate-array and a set of nets to be routed, we wish to find a routing that uses as little channel space as possible. We present a deterministic approximation algorithm that uses close to the minimum possible channel space. We cast the routing problem as a new form of zero-one multicommodity flow, an integer programming problem. We solve this integer program approximately by first solving its linear program relaxation and then rounding any fractions that appear in the solution to the linear program. The running time of the rounding algorithm is exponential in the number of terminals in a net; the algorithm is thus best suited to cases where the number of terminals on each net is small.

LIMITED DISTRIBUTION NOTICE

IBM Research Division
Almaden • Yorktown • Zurich

1. Problem Statement

A gate-array is a two-dimensional array of gates. A logic circuit is realized by connecting together some of these gates. In an instance of the routing problem, we are given a collection of sets of gates that are to be connected. Each such set of gates is called a *net*. The space over the array is used for routing the wires interconnecting the gates. Wires follow rectilinear or "Manhattan" paths between gates. The number of wires that can pass between adjacent gates is limited by the width of the boundary between the gates. This boundary is called the *channel*

This work was done while the authors were with the Computer Science Division, University of California at Berkeley. The work of Prabhakar Raghavan was supported by an IBM Doctoral Fellowship, and the work of Clark Thompson was supported by a California State MICRO grant (AT&T Foundation).

between the gates, and the number of wires that can pass through it is called the *capacity* of that channel. In solving the *global routing problem*, we are to connect the gates in each net without violating the capacity constraints of any channel. These ideas are made precise in the formal model below.

1.1. Our Approach

We show that the global routing problem for multiterminal nets can be cast as a special form of zero-one multicommodity flow. This is an zero-one linear program that is difficult to solve efficiently; indeed, the problem is NP-complete even for two-terminal nets [10]. We therefore try to obtain an approximate solution to this integer program whose solution is close to the optimum. We first solve a linear program relaxation of the integer program; this can be done in polynomial time [3]. The solution to the linear program could contain fractional values for some of the variables, and is therefore not an admissible solution to the global routing problem. We "round" these fractional values in deterministic polynomial time in such a manner that the resulting integer solution (global routing) is close to the best possible in terms of channel space.

The main techniques developed in this paper are: (1) the formulation of multiterminal routing as a new form of multicommodity flow problem; and (2) a simple deterministic algorithm for rounding linear program solutions that is easy to analyze. Using these methods, we develop an algorithm for global routing that is *provably good* in that it finds a solution close to the best possible. Our performance guarantee will be in terms of the best possible routing for the instance. The notions of "best possible routing" and "nearly optimal" will be apparent from the model below.

1.2. The Global Routing Model

1. A *regular array* is a two-dimensional $m \times n$ lattice $L(V, E)$ where the lattice nodes represent gates, and the edges between nodes represent channels through which nets can be routed.
2. A *net* is a set of nodes that are to be connected.
3. An instance of the routing problem is a set R of nets.
4. A *connection* between two nodes v_a and v_b is a simple path in L between v_a and v_b. A net is said to be routed if there exist connections between every pair of nodes in the net. A net is thus routed by specifying a tree in L that spans the nodes of the net.
5. A solution to the routing problem is a set S of trees such that every net is routed.
6. The *width* of an edge given S is the number of trees of S that use that edge. The *width of the solution* S is the maximum edge width among all the edges.
7. An *optimal solution* is one whose width is the minimum, among all possible solutions.

A number of features of the model are noteworthy. We are interested only in the number of routes passing through an edge - corresponding to the "flux" through a channel - and not in the manner in which wires are physically arranged within the channel. We are thus interested only in the global routing problem and not in the channel routing problem. Furthermore, we are solving a "bottleneck" optimization problem, one in which we consider the edge with the maximum width - the worst case occurring in the array. This is reasonable in practical situations, where we cannot afford to congest even one channel in the array. Finally, it should be observed that the global routing problem in practice is a *feasibility* problem. We are solving an equivalent *optimization* problem, one that is at least as hard as the feasibility problem.

1.3. Related Work

A number of algorithms and heuristics have been proposed for the global routing problem. Lee popularized the "maze-running" style of global routing [6] . The maze algorithm gives no bound on the width of the routing (compared with the optimal solution) within polynomial running time. Burstein and Pelavin [2] used dynamic programming to develop an approximation algorithm for the global routing problem. Subsequently, Karp *et al.* [4] used a linear programming approach to develop a provably good approximate solution. In particular, they showed that if C were the optimal width for a problem instance, their method would find a routing of width at most $O(\ C\ \log \frac{n}{C}\ )$. Their algorithm can handle nets containing arbitrarily many nodes, whereas the one we develop below is best suited to small nets. More recently, the heuristic of simulated annealing has been applied to the problem by Vecchi and Kirkpatrick [5].

2. Multiterminal Multicommodity Flows

For the remainder of the paper, we only describe problem instances in which every net has exactly three nodes. Generalizations to larger nets are straightforward, but are practical only when the nets are not too large. We now describe the formulation of the routing problem as a zero-one multicommodity flow problem.

Let t_{i1} , t_{i2} and t_{i3} be the three nodes of net i. Clearly, the tree in the solution that is used to route this net must have a Steiner point - a node in L connected by simple paths to t_{i1} , t_{i2} and t_{i3}. An indicator variable $s_{ij} \in \{0, 1\}$ in the integer program denotes whether or not vertex $v_j \in L$ is the Steiner point for net i. The global routing problem is then a multicommodity flow problem in which s_{ij} units of flow are to be shipped from each of t_{i1} , t_{i2} and t_{i3} to node v_j. We denote by $f_i(e)$ the total flux of net i (from all its three nodes) through edge e. The net flux through each edge is not to exceed W, where W is the objective function (width) to be minimized in the integer program. All flows are to be integral, and conservation laws are to be respected at each node for every commodity.

3. The Relaxation Linear Program

Consider a linear program relaxation of the integer multiterminal multicommodity flow problem, in which all the variables (flows as well as Steiner point indicators) are allowed to take on real values in the interval [0, 1]. Let us call this relaxation linear program "LP". We can solve LP in polynomial time. The resultant solution could contain fractional values for the variables $f_i(e)$, s_{ij}, and W of LP. We denote by F a feasible solution to LP, i.e. a set of non-negative real values for the variables $f_i(e)$ and s_{ij}. Thus, the set of real values that constitute F must satisfy conservation laws, and furthermore $\sum_j s_{ij} = 1 \ \forall i$. The last equality is a relaxation of the requirement that we have a Steiner point for each net; instead, we now have "fractional" Steiner points for each net that together sum up to one Steiner point. Note that the definition of the feasible flow F has nothing to do with optimality.

3.1. The Potential Function Algorithm

Let $\hat{W}$ be the optimum width obtained from the solution to LP. Let $\hat{f}_i(e)$ and $\hat{s}_{ij}$ be the corresponding values for the various flows and Steiner point "indicators". Let $r = |R|$ denote the number of nets in the problem instance, and $N = m(n-1) + n(m-1)$ be the number of edges in L (i.e. the number of channels in the array). The following are evident:

$$\sum_{j=1}^{mn} \hat{s}_{ij} = 1, \quad 1 \le i \le r \tag{1}$$

$$\sum_{i=1}^{r} \hat{f}_i(e) \le \hat{W}, \quad e \in E$$

Here all the $\hat{s}_{ij}$, $\hat{f}_{ij}(e) \in [0, 1]$. Let us define the *initial feasible flow* F_0 to be the set of real values given by the linear program solution. To be precise,

$$F_0 = \{ \hat{W}, \hat{f}_i(e), \hat{s}_{ij}, \ 1 \le i \le r, 1 \le j \le mn, e \in E \}$$

The initial feasible flow F_0 corresponds to a fractional routing for all the nets.

We define the following potential function, which is a mapping from feasible flows to positive real numbers. This potential function will play a key role in our algorithm.

$$\Phi(F) = \sum_{e \in E} \prod_{i=1}^{r} [f_i(e) \Delta + 1 - f_i(e)] \ , \ f_i(e) \in F$$

In section 3.2 we will specify a value for $\Delta > 1$; then $\Phi(F)$ is guaranteed to be positive since the values of the flows are non-negative.

We now outline the algorithm. The idea is to progress in stages from F_0 to a sequence of feasible flows $F_1, F_2, \ldots, F_r$ with the following properties.

1. There are r stages in all. During stage i, $1 \leq i \leq r$, we progress from feasible flow F_{i-1} to feasible flow F_i.
2. At the end of stage i, the feasible flow F_i that we discover during that stage will have integral values for all flows $f_k(e)$, $1 \leq k \leq i$ and for all Steiner point indicators s_{kj}, $1 \leq k \leq i$. In other words, we would have found integral flows (routings) for nets 1 through i inclusive.
3. Once a net is routed, it is not disturbed again. Thus the flow of net i is changed only during stage i and at no other time.
4. Each of the r rounding stages consists of two phases, which we describe in the following subsections. In the first phase of stage i, a Steiner point is chosen for net i. In the second phase of the stage, we find paths connecting the nodes of net i to the chosen Steiner point. The rounding algorithm will make use of the potential function at each phase of each stage.

3.1.1. Choosing the Steiner Point

We will now describe how a Steiner point is chosen for net i during stage i. This corresponds to rounding s_{ij} to 1 for exactly one value of j, and rounding the rest to 0.

Let us fix our attention on one potential Steiner point (for net i), node v_p in the lattice. Recall that in our linear programming formulation, we ensured that s_{ip} units of flow were conveyed from each node of net i to node v_p. Thus, in the solution to LP, we will have $\hat{s}_{ip}$ units of flow from each node of net i to v_p. If we actually chose v_p to be the Steiner point for net i, we would be setting s_{ip} to 1 and all s_{ij}, $j \neq p$ to 0. As a consequence, the flows from the nodes of net i to v_{ip} should all be increased to 1 while flows to all other potential Steiner points should be nullified. To facilitate this discussion, we denote by $f_{ij}(e)$ the flow from the nodes of net i to node v_j through edge e, and by $\hat{f}_{ij}(e)$ the value assigned to $f_{ij}(e)$ by the solution to LP.

$$\sum_{j=1}^{nin} \hat{f}_{ij}(e) = \hat{f}_i(e) \tag{2}$$

Thus, if we select v_p as the Steiner point for net i, we replace $\hat{f}_{ip}(e)$ by $\frac{1}{\hat{s}_{ip}} \times \hat{f}_{ip}(e)$, and $\hat{f}_{ij}(e)$ by 0 for $j \neq p$.

Let us denote by $f'_{ij}(e)$ and $s'_{ij}(e)$ the new values of the variables (after a Steiner point is chosen and flow values adjusted accordingly). It is clear that these values constitute a feasible flow: they

are scaled versions of the original flows (scaled by either $1/\hat{s}_{ip}$ or 0) and thus satisfy conservation laws; 1 unit of flow leaves each node of the net; and 1 unit of flow arrives at the chosen Steiner point from each node of the net. Of course, some of the edges may now have width exceeding $\hat{W}$, but we still have a feasible flow. Let us denote by F'_i the feasible flow obtained by choosing the Steiner point. The feasible flow F'_i is an intermediate flow between F_{i-1} and F_i.

We will now describe how the Steiner point is picked. The algorithm is simple: we choose that vertex v_j as the Steiner point which minimizes $\Phi(F'_i)$. We denote the (already computed) "rounded" flows of nets 1 through $i-1$ by $\bar{f}_i(e)$.

Lemma 1 :

$$\Phi(F'_i) \leq \Phi(F_{i-1})$$

Proof : Let us denote by F'_{ij} the feasible flow resulting from the choice of v_j as the Steiner point for net i. We will show that $\min_j \Phi(F'_{ij}) \leq \Phi(F_{i-1})$, the minimum being taken over all possible Steiner points v_j. Let us define

$$K_i(e) = \prod_{k=1}^{i-1} [\bar{f}_k(e)\Delta + 1 - \bar{f}_k(e)] \times \prod_{k=i+1}^{r} [\hat{f}_k(e)\Delta + 1 - \hat{f}_k(e)]$$

Then

$$\Phi(F_{i-1}) = \sum_{e \in E} K_i(e) [\hat{f}_i(e)\Delta + 1 - \hat{f}_i(e)]$$

while

$$\Phi(F'_{ij}) = \sum_{e \in E} K_i(e) \left[\frac{\hat{f}_{ij}(e)}{\hat{s}_{ij}} \Delta + 1 - \frac{\hat{f}_{ij}(e)}{\hat{s}_{ij}} \right]$$

Using (1) and (2), we can now write

$$\Phi(F_{i-1}) = \sum_{j=1}^{mn} \hat{s}_{ij} \times \Phi(F'_{ij})$$

But this means that $\Phi(F_{i-1})$ is a convex combination of all the $\Phi(F'_{ij})$, and therefore one (and in particular the minimum) of the $\Phi(F'_{ij})$ must be at most $\Phi(F_{i-1})$. ■

Thus our algorithm chooses a Steiner point without allowing the potential function to rise in value.

3.1.2. Choosing routes for net i

We have just described the procedure for picking a Steiner point (let us call it v_p) for net i. In the process, we rounded all the variables s_{ij} , $j \neq p$ to $s'_{ij} = 0$ and s_{ip} to $s'_{ip} = 1$. We now pick paths from the nodes t_{i1}, t_{i2} and t_{i3} to the chosen Steiner point v_p. In doing so we will round all the flows $f'_{ip}(e)$ to 0 or 1 (recall that all flows $f'_{ij}(e)$, $j \neq p$, have already been set to 0). The feasible flow thus obtained will be F_i .

We now describe how node t_{i1} of net i is connected to the chosen Steiner point v_p. Identical procedures are followed for nodes t_{i2} and t_{i3}. Let $f'_{ip1}(e)$ denote that component of $f_{ip}(e)$ that flows from node t_{i1} to v_p; thus

$$f'_{ip}(e) = f'_{ip1}(e) + f'_{ip2}(e) + f'_{ip3}(e)$$

To begin with, we convert the flows $f'_{ip1}(e)$ from t_{i1} to v_p into a set of paths $\Gamma_{i1} = \{ P_1, P_2, \ldots, P_k \}$ that can be used for connecting t_{i1} to v_p. To do this, we use a procedure known as "path stripping" which has been used before for global routing [11] and is similar in spirit to a process used in the network flow algorithm of Malhotra, Kumar and Maheshwari [7].

Path stripping consists of the following three phases:

(1) Form a directed subgraph $L_{i1}(V,E_{i1})$ where E_{i1} is the set of directed edges derived from E as follows: for each $e \in E$, assign a direction to e which is the direction of the flow $f'_{ip1}(e)$. If $f'_{ip1}(e) = 0$, e is excluded from E_{i1}.

(2) Discover a directed path $\{ e_1, \ldots, e_L \}$ in L_{i1} from t_{i1} to v_p using a depth-first search. Let

$$f_m = \min_{1 \leq g \leq L} \{ f'_{ip1}(e_g) \}$$

For $1 \leq g \leq L$, replace $f'_{ip1}(e_g)$ by $f'_{ip1}(e_g) - f_m$. Add the path $\{ e_1, \ldots, e_L \}$ to Γ_{i1} along with its *weight* f_m.

(3) Remove any edges with zero flow from E_{i1}. While there is non-zero flow leaving t_{i1}, repeat (2).

We remark that since at each execution of step (2) we remove at least one edge of E_{i1} (that edge with the minimum flow f_m), the above process takes at most $O(mn)$ time. Let w_l be the weight of path P_l , $1 \leq l \leq k$. It is clear that

$$\sum_{l=1}^{k} w_l = 1 \qquad (3)$$

and

$$\sum_{e \in P_l} w_l = f'_{ip1}(e) \qquad (4)$$

We can think of the feasible fractional flows from t_{i1} to v_p as being the superposition of the flows w_l in the paths P_l. We will now use one of the paths P_l to connect t_{i1} to v_p. When we do so, the flow in all edges of P_l is rounded up to 1, and the flow in all other edges of E_{i1} is set to 0; a phenomenon similar to the choice of the Steiner point.

The algorithm for choosing the path is similar to the one used for the choice of the Steiner point: we choose that path which minimizes the new value of the potential function. Let $\Phi\ (\ F'(P_l)\ )$ denote the new value of the potential function if P_l is used to connect t_{i1} to v_p. It is easy to prove that

$$\Phi\ (F'_{ip}) \quad = \quad \sum_{l=1}^{k} w_l \ \times \ \ \Phi\ (\ F'(P_l)\ )$$

The proof is similar to lemma 1, with the role of equations (1) and (2) being replaced by (3) and (4). We can now assert, as in lemma 1, that in our choice of the path connecting t_{i1} to v_p, we have not caused the potential function to rise. In a similar manner, nodes t_{i2} and t_{i3} are connected to v_p. Throughout, we ensure that the potential function does not rise. We can thus establish the following:

Lemma 2 :

$$\Phi\ (F_i) \ \leq \ \Phi\ (F_{i-1})$$

We have thus described how net i is routed. The process is repeated for all the nets $1 \leq i \leq r$, and we can then assert the following about the final value of the potential function:

Corollary 3 :

$$\Phi\ (F_r) \ \leq \ \Phi\ (F_0)$$

3.2. The Quality of the Approximation

In the preceding sections, we described the rounding algorithm which converted the solution to LP to a routing. In our discussion of feasible flows and potential functions during the algorithm, we did not explicitly consider the value of the width of the edges at any stage of the algorithm (except at the beginning, where we knew the width to be $\hat{W}$). We now show that the width of the solution produced by our rounding algorithm is close to $\hat{W}$. Since $\hat{W}$ is the optimum for the relaxation linear program, it is a lower bound on the best possible width of any integer solution (routing).

We now provide the analysis for the performance guarantee. We first specify the value used for Δ in the computation of the potential function. We choose Δ to satisfy the equation

$$\left[\frac{\exp(\Delta - 1)}{\Delta^{\Delta}} \right]^{\hat{W}} = \frac{1}{N} \tag{5}$$

Recall that N is the number of edges in the array. Using this value of Δ, we now compute the value of $\Phi(F_0)$.

$$\Phi(F_0) = \sum_{e \in E} \prod_{i=1}^{r} [\hat{f}_i(e)\,\Delta + 1 - \hat{f}_i(e)] \leq \sum_{e \in E} \prod_{i=1}^{r} \exp\{\hat{f}_i(e)[\Delta - 1]\}$$

$$= \sum_{e \in E} \exp\{\sum_{i=1}^{r} \hat{f}_i(e)[\Delta - 1]\} \leq \sum_{e \in E} \exp\{\hat{W}[\Delta - 1]\} \leq \Delta^{\Delta \hat{W}}$$

The last inequality follows from (3), and the fact that there are N edges in all. Using corollary 3, we can now conclude that

$$\Phi(F_r) \leq \Delta^{\Delta \hat{W}}$$

We now note that either $\bar{f}_i(e) = 1$ (if some path of the routing of net i passes through edge e), or $\bar{f}_i(e) = 0$. This is not entirely obvious, since it can happen that more than one of the three paths of net i may be routed through e. However, in this case only one unit of flow (one wire) need physically pass through that edge (channel), since all the paths of the net are electrically the same. We can thus ignore multiple passages of a net through an edge, counting only one unit of flow in such cases. Let $\bar{f}(e)$ be the total flow through edge e - the width of the edge - at the end of the rounding. Then, it is clear that

$$\Phi(F_r) = \sum_{e \in E} \prod_{i=1}^{r} [\bar{f}_i(e)\,\Delta + 1 - \bar{f}_i(e)]$$

$$= \sum_{e \in E} \Delta^{\bar{f}(e)} \leq \Delta^{\Delta \hat{W}}$$

Since all the terms of the summation are nonnegative, we can conclude that for *any* edge e,

$$\Delta^{\bar{f}(e)} \leq \Delta^{\Delta \hat{W}}$$

This implies the following theorem.

Theorem : The width of the solution produced by the rounding algorithm is at most $\Delta \hat{W}$.

3.2.1. A Discussion of the Bound

Owing to the choice of Δ by the somewhat complicated equation (3), our performance guarantee may be difficult to interpret. We now give some simplified versions of the guarantee. Using some algebraic manipulation, it can be shown that provided $\hat{W} > \ln N$, our performance guarantee becomes

$$Width \leq \Delta \hat{W} \leq \hat{W} + (e-1)\sqrt{\hat{W} \ln N}$$

Note that the e referred to here is the base of the natural logarithm. For such values of $\hat{W}$, our performance guarantee is especially good; $\hat{W}$ is the best possible width we can possibly achieve, and we are off by an additive factor that is small as $\hat{W}$ becomes large and N remains fixed. Our bound is not as tight when $\hat{W}$ becomes small. More algebraic manipulation shows that when $\hat{W} < \ln N$, our guarantee becomes

$$Width \leq \frac{e \ln N}{\ln \left[\frac{e \ln N}{\hat{W}} \right]}$$

4. Concluding Remarks

Our algorithm can be modified to handle more than three nodes per net; the number of Steiner points grows, and as a consequence the number of variables in the linear program. Note that the number of Steiner point indicator variables is exponential in the number of nodes in a net, since we have to consider all pairs of Steiner points for four-node nets, all triples for five-node nets, and so on. It remains open whether this difficulty can be circumvented by means of a better integer program formulation than ours. Note that our formulation is not exponential in the size of the gate-array or the number of nets.

We have not included a careful analysis of the running time of the rounding algorithm because it is dominated by the running time of the linear program phase (both in theory and in practice). We do note that the entire rounding algorithm is polynomial-time. The practical significance of our method remains to be seen; the cost of solving the linear program is considerable. We are initiating experimental work on a related randomized algorithm [8].

Our algorithm is derivable from probabilistic methods [12] , but we have chosen to present a simple combinatorial description here to highlight the practical flavor of the problem. The potential function approach is similar in spirit to algorithms for set-balancing and related combinatorial problems due to Olsen and Spencer [9] and also Beck and Fiala [1].

4.1. Acknowledgements

The authors would like to thank Tom Leighton and Charles Leiserson for helping develop the multicommodity flow formulation, and Ravi Boppana and Joel Spencer for interesting discussions on potential functions.

References

[1] Jozsef Beck and Tibor Fiala, ""Integer-Making" Theorems", *Discrete Applied Mathematics* , pp. 1-8, 1981.

[2] Michael Burstein and Richard Pelavin, "Hierarchical Wire Routing", *IEEE Transactions on Computer-Aided Design* , vol. CAD-2, no. 4, pp. 223-234, October 1983.

[3] Narendra Karmarkar, "A new polynomial-time algorithm for linear programming", *Combinatorica* , vol. 4, no. 4, pp. 373-396, 1984

[4] R.M. Karp, *et al.* , "Global Wire Routing in Two-Dimensional Arrays", *Proceedings of the 24th Annual Symposium on FOCS* , pp. 453-459, October 1983. Also submitted to *Algorithmica* .

[5] Scott Kirkpatrick and Mario P. Vecchi, "Global Wiring by Simulated Annealing", *IEEE Transactions on Computer-Aided Design* , vol. CAD-2, no. 4, pp. 215-222, October 1983.

[6] C. Y. Lee, "An algorithm for path connections and its applications", *IRE Transactions on Electronic Computers* , vol. EC-10, pp. 346-365, September 1961.

[7] V.M. Malhotra, M.P. Kumar and S.N. Maheshwari, "An $O(|V|^3)$ algorithm for finding maximum flows in networks", *Information Processing Letters* , vol. 7, pp. 277-278, 1978.

[8] Antony P-C Ng, Prabhakar Raghavan and Clark D. Thompson, "Experimental Results with a Linear Program Global Router", to appear in *Computers and Artificial Intelligence* , 1987.

[9] John E. Olson and Joel Spencer, "Balancing Families of Sets", *Journal of Combinatorial Theory, Series A* , vol. 25, no. 1, pp. 29-37, July 1978.

[10] Prabhakar Raghavan, "Randomized Rounding and Discrete Ham-Sandwich Theorems: Provably Good Algorithms for Routing and Packing Problems", *PhD Thesis, UC Berkeley* , August 1986.

[11] Prabhakar Raghavan and Clark D. Thompson, "Provably Good Routing in Graphs: Regular Arrays", *Proceedings of the 24th ACM STOC* , pp. 79-87, May 1985.

[12] Prabhakar Raghavan, "Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs", to appear in *Proceedings of the 27th Annual Symposium on FOCS* , October 1986.

Copies may be requested from:

IBM Thomas J. Watson Research Center
Distribution Services 73-F11
Post Office Box 218
Yorktown Heights, New York 10598