Global Routing in Gate Arrays: Algorithms and Complexity ‡

Teofilo F. Gonzalez
Department of Computer Science
University of California
Santa Barbara, CA 93106

ABSTRACT

In this paper we study the computational aspects of the gate array global routing problem. We develop an algorithm that generates a routing with congestion which is bounded by 2*OPT, where OPT is the congestion of an optimal solution. Our algorithm reduces the global routing problem to the solution of a set of linear programming problems which can be solved efficiently. Our algorithm is the only known algorithm for which one can show that the solution value is within a fixed percentage of the optimal solution value.

‡ This research was supported in part by the National Science Foundation under Grant DCR - 8503163.

INTRODUCTION

Gate arrays have been successfully used as a design tool to generate, with a fast turnaround time, computer chips. In this paper we study the computational aspects of the gate array global routing problem. A typical gate array is given in figure 1. A gate array consists of r-1 rows of active areas, i.e., where components (transistors or high level function blocks) have been placed. In each of the active areas there is only one layer for routing. All the routing wires that one may introduce in these areas wires that run along the vertical tracks. Between every pair of adjacent active areas, there is a nonactive area or (horizontal) channel in which two layers are available for routing. The wires in one layer must run along the vertical tracks and the wires in the other layer must run along the horizontal tracks.

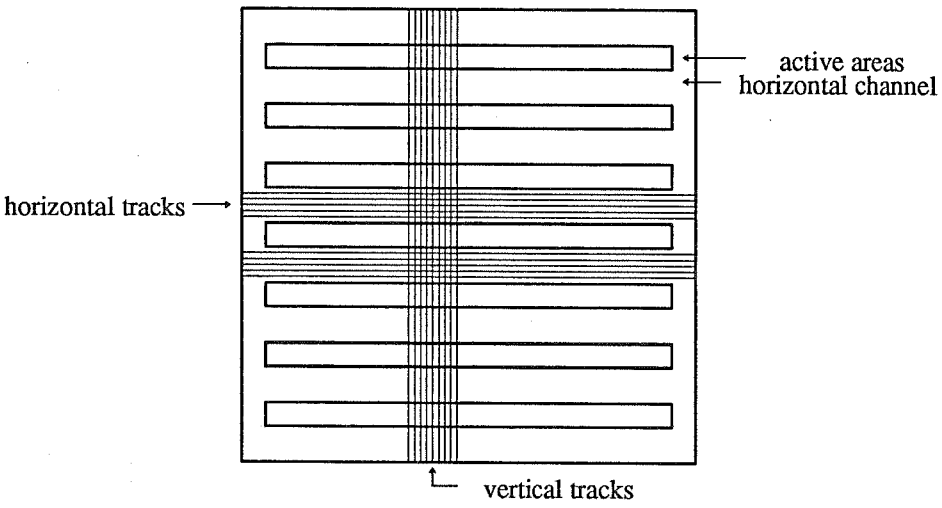

Figure 1: Typical gate array.

In the gate array design methodology, we initially place the components (transistors, high level function blocks, etc.) on the active areas and then we try to interconnect all points that need to be made electrically common by using the tracks available for wiring. This procedure is repeated until a successful wiring is obtained, or until we have enough confidence that such wiring does not exist. In the latter case we partition our circuit into two or more computer chips.

In this paper we present an algorithm to solve the gate array global routing problem. The typical approach for solving this routing problem begins by dividing the gate array into r by c *cells* as illustrated in figure 2. The cells are referred to as $C_{i,j}$ for $1 \le i \le r$ and $1 \le j \le c$. Cell $C_{i,j}$ contains the set of points $S_{i,j}$ that need to be made electrically common. The number of elements in each of these sets is at most p. The set of points $S = \cup S_{i,j}$ in all cells has been partitioned into $N = \{ N_1, N_2, ..., N_n \}$. Each set N_i is called a *net*. All the points in each net need to be made electrically common by interconnecting them by wires. Each wire must follow a path in the routing regions specified above, and since there are two layers the path consists of alternating vertical and horizontal line segments. Note that in this design methodology the number of tracks available for wiring is an upper bound on the maximum number of wires that cross from one cell to any of its adjacent cells. The typical approach for solving the gate array routing problem consists of the following two steps:

(1) *Global Routing:* Find the global routing R, which specifies for each net the set of cell boundaries that the wire connecting all the points in this net cross. In a global wiring R the number of wires that cross the boundary between adjacent cells must not exceed the capacity of the cell, i.e., the number of tracks that cross the edge.

(2) *Detail Routing:* For each net specify the exact position for the wires.

Of course not all global routings R have a detailed routing. However, the complexity of our problem is greatly reduced by dividing our routing problem into two subproblems.

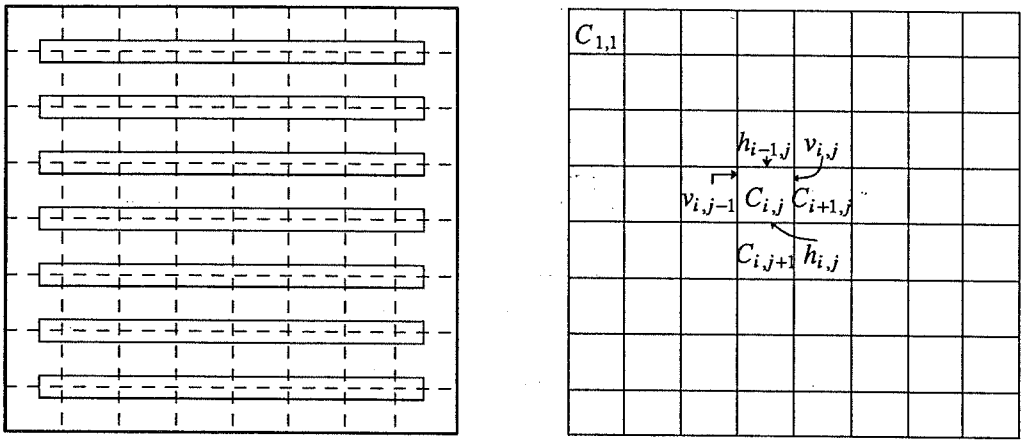

Figure 2: Partition of the gate array.

In this paper we assume that all nets are of size two. An approach similar to ours can be used to solve the more general problem. Figure 3 shows different global wires that connect a net. A wire is called a *t-turn wire* if it makes t turns, i.e., it bends t times (see figure 3). A global wiring R is called a *t-turn global wiring* if every wire that connects a pair of points contains at most t turns or bends. The capacity of the edges adjacent to $C_{i,j}$ is given by $v_{i,j-1}$, $v_{i,j}$, $h_{i-1,j}$ and $h_{i,j}$ (see figure 2). Of course $v_{i,0} = v_{i,c} = 0$ for $1 \le i \le r$; and $h_{0,j} = h_{r,j} = 0$ for $1 \le j \le c$. For a global routing R, let $x_{i,j}$ ($y_{i,j}$) be the number of wires that cross the boundary between $C_{i,j}$ and $C_{i,j+1}$ ($C_{i,j}$ and $C_{i+1,j}$). The objective of the global routing problem is to find a global routing R such that the *congestion* (defined by $B = \mathbf{max}\{x_{i,j}, y_{i,j}\}$) is least possible. Karp *et. all.* [KL] showed that the global routing problem is NP-complete even when p = 1 and all the nets are of size two. Shing and Hu [SH] developed a polynomial time algorithm to generate suboptimal solutions to the gate array global routing problem. Raghavan and Thompson [RT] developed algorithms that guarantee provable good solution with high probability. These two algorithms generate a solution by reducing the global routing problem to a linear programming problem. Other algorithms for our problem appear in [GAM], [NH], [TT] and [VK]. The techniques behind these algorithms ranges from simple heuristics to simulated annealing. The underlying characteristic of all the previous algorithms for global routing is that none of them have been shown to generate solutions within a fixed percentage of the optimal solution value. *That is the main difference between our algorithms and the previous algorithms for global wiring.*

Let OPT_k be the congestion of an optimal k-turn routing R and let OPT be the congestion of an optimal routing R (there is no limit on the number of bends in any wire). Clearly $OPT_{k+1} \le OPT_k$ and $OPT_k \le$ OPT. Karp *et. all.* [KL] also showed that for some problem instances $OPT_1 \ge ((r+c)/2)$OPT. The algorithm developed by Thomson and Raghavan generates a routing with congestion at most $2*OPT_1$. However, this does not imply that such routing has a congestion bounded by 2*OPT. A wire is said to be a *t-turn falling wire* if it is a t-turn wire and every horizontal line intersects at most once the vertical sections of the wire. All the wires in in figure 3, except for the 6-turn wire, are falling wires. A global routing R is said to be a *t-turn falling routing* if all the wires are t-turn falling wires. Let OPT_k' be the congestion of an optimal k-turn falling routing R and let OPT be the congestion of an optimal falling routing R (there is no limit on the number of bends in each wire). Clearly, $OPT_k \le OPT_k'$ and OPT $\le$ OPT'. However, OPT and OPT' are in general very close to each other. In this paper we develop an algorithm that generates a routing with congestion which is bounded by 2*OPT'. It is interesting to note that the wirings generated by our algorithm have the same provable good behavior as the ones generated by the algorithm in [RT]. Our algorithm reduces the global routing problem to the solution of a set of linear programming problems which can be solved by the classical algorithms [GA] or the newer algorithms [Ka], [K], and [Va].

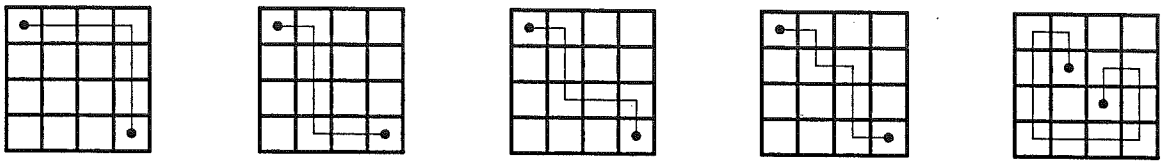

Figure 3: t-turn global wires.

II. TRANSFORMATION TO THE VIA PLACEMENT PROBLEM

Let us now outline our approach. First we transform the problem to an equivalent problem whose solution can be used to solve the original problem. In the new problem we also have r*c cells denoted by $C_{i,j}$. Associated with cell $C_{i,j}$ there is a region for placing via columns referred to as *via slot* $V_{i,j}$. Let us assume for the moment that net N_i consist of points located in cells $C_{1,1}$ and $C_{4,4}$. In this case we introduce three via columns. Each via column has to be placed in one of the *via column slots* formed by the via slots located in the same column. The ith via column has a pin in row i and a pin in row i+1. In figure 5 (1-turn) all the via columns are placed in via column slot 4; in figure 5 (2-turn) all the via columns are placed in via column slot 2; in figure 5 (3-turn) the first two via columns are placed in via column slot 2 and the last one in via column slot 4; in figure 5 (4-turn) the first via column is placed in via column slot 2 and the other two in via column slot 3; and in figure 5 (5-turn) the ith via column is placed in the i+1st via column slot. In the new problem we only care to count the wires that cross the thick lines (see figure 5). Note that any t-turn falling wire can be obtained by placing the via columns in the appropriate via column slots. The global routing for the original problem is obtained by ignoring the via points and considering only the "thick" boundaries crossed by the wire.

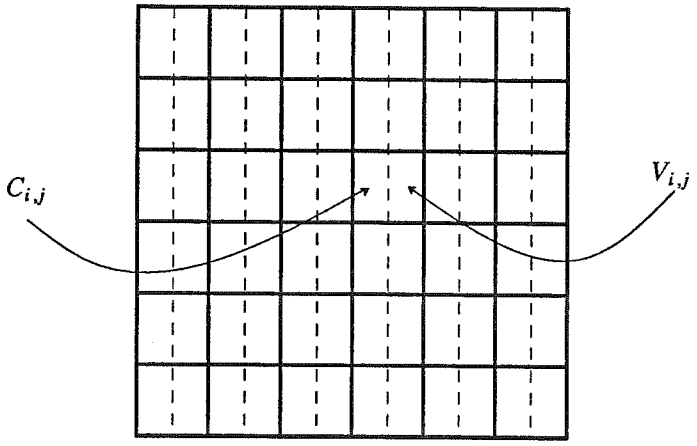

Figure 4: Component cells and via slots.

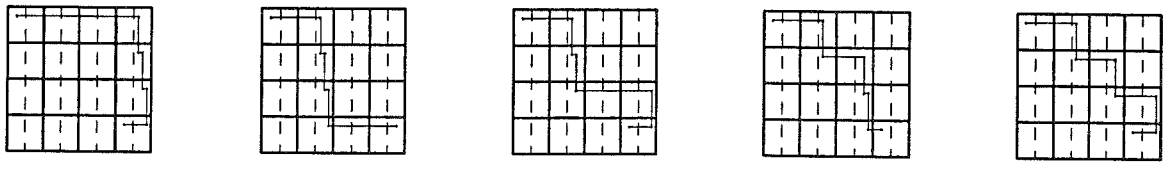

Figure 5: t-turn global wires after placement of the via columns.

At this point the original problem has been transformed into the via placement problem. A similar version of this problem, the via placement problem for MPCBs, has been studied in [GK]. Gonzalez and Kurki-Gowdara [GK] present a polynomial time algorithm that generates solutions that are within twice the optimal solution value. The main difference between these problems is that in the via placement problem for MPCBs we only care to minimize the row congestion. The column congestion is not important. The main problem now is how to achieve also a small column congestion.

A solution to this more complex problem can be obtained by modifying the previous transformation. The new problem has a solution in which every vertical or horizontal wire crosses no more than one of the thick boundaries. For problems of this type we have developed an algorithm similar to the one in [GK] that generates a solution with objective function value within twice the value of an optimal solution. The technique is similar to the one in [GK], but the analysis is much more complex. The idea of "short wires" is abstracted in the following two figures where the via slots have been granulated into a set of smaller slots.

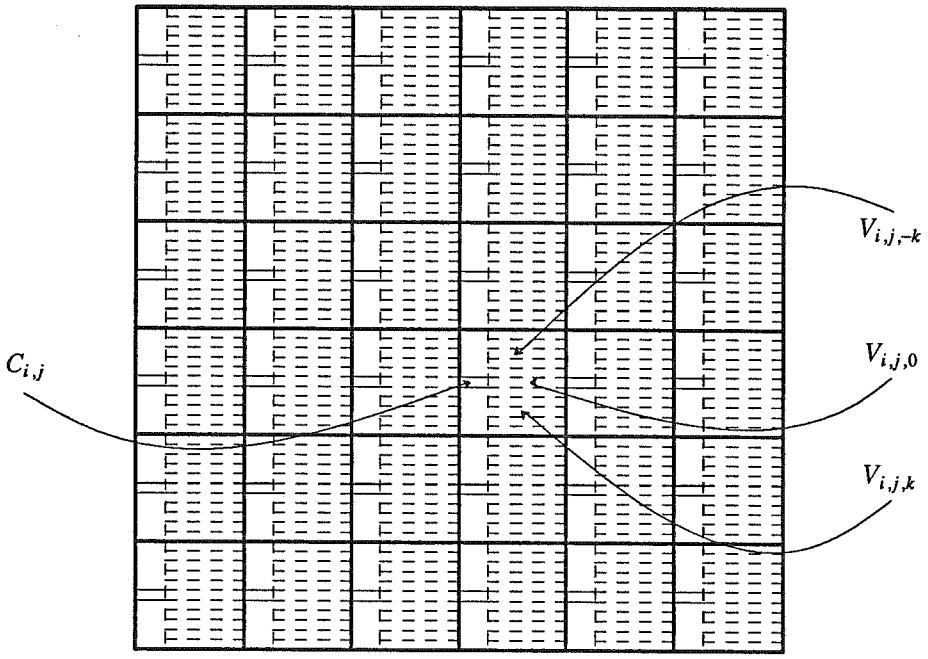

Figure 6: Component cells and via slots that allow "short" wires.

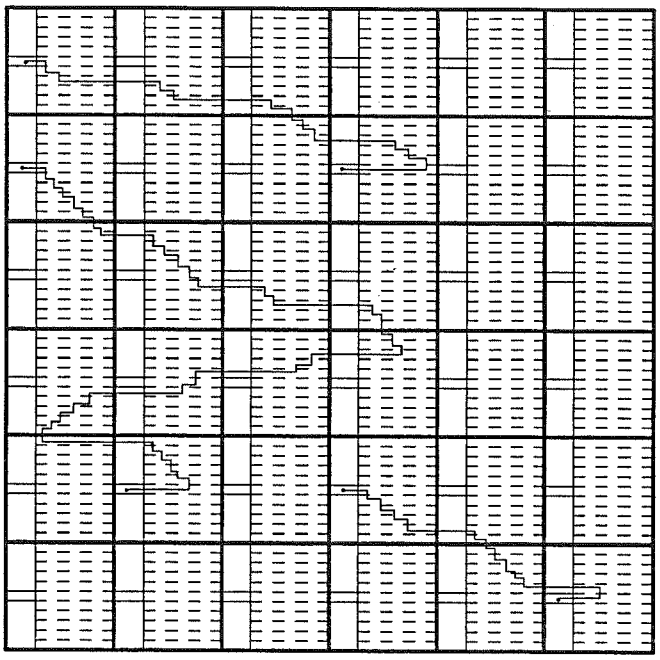

Figure 7: Global wirings using via column slots that allow short wires.

We proposed an algorithm to solve the global routing problem in gate arrays. Our algorithm consists of transforming the problem into a set of linear programming problems, which can be solved efficiently. The main advantage of our algorithm over previously known algorithms is that our algorithm generates solutions that are provably good. The difference with respect to the time complexity of our algorithm and the ones in [RT] and [SH] is minimal since in both cases we transform the problem into a linear programming problem. From the practical point of view, our algorithms are important because they provide us with an estimate of the optimal solution value. In practical situations our algorithm should be run concurrently with the other algorithms and then select the best of the solutions. With our bound we can estimate the quality of our solutions.

REFERENCES

[GJ2] Garey M. and D. Johnson, *"Computers and Intractability: A Guide to the Theory of NP-Completeness"*, Freeman, 1979.

[GAM] Gamal, A El, "Two-Dimensional Stochastic Model for Interconnections in Master-Slice Integrated Circuits," *IEEE Transactions on Circuits and Systems*, Vol. CAS-28, pp. 127 - 137, Feb. 1981.

[GA] Gass, S.,"Linear Programming," Mc-Graw Hill, 1969.

[G] Gonzalez T., "An Approximation Algorithm for the Via Assignment Problem," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-3, No 4, Oct. 1984, pp. 257-264.

[GK] Gonzalez, T. and S. Kurki-Gowdara, "An Approximation Algorithm for the Via Placement and Related Problems", Proceedings of the Twenty-Fifth Annual Allerton Conference on Communications, Control and Computing, Oct. 1987.

[Ka] Karmarkar, N., "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, Vol. 4, 1984, pp. 373-395.

[KL] Karp R. M., F. T. Leighton, R. L. Rivest, C. D. Thompson, U. Vazirani and V Vazirani, "Global Wire Routing in Two-Dimensional Arrays," Proceedings of the Twenty-fourth Annual Symposium on Foundations of Computer Science, Oct. 1986, pp. 453 - 459.

[K] Khachiyan, L. G., "A Polynomial Algorithm in Linear Programming," *Doklady Akademiia Nauk SSSR*, 244:S (1979), pp. 1093-1096, translated in *Soviet Mathematics Doklady*, 20:1 (1979), pp. 1-12.

[NH] Nair, R., S. J. Hong, S. Liles, and R. Villani, "Global Routing On a Wire Routing Machine," Proceedings of the Nineteenth Design Automation Conference, June 1982, pp. 224 - 231.

[RT] Raghavan P. and C. D. Thompson, "Provably Good Routing in Graphs: Regular Arrays," Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computation, May 1985, pp. 79 - 87.

[S] Sahni, S., "Computationally Related Problems," *SIAM Journal on Computing*, 3, 4 (Dec. 1974), pp. 262-279.

[SH] Shing M. T. and T. C. Hu, "Computational Complexity of Layout Problems," *Layout Design and Verification*, T. Ohtsuki (Editor), Elsevier Science Publishers, 1986, pp. 267 - 294.

[TT] Ting, B. S. and B. N. Tien, "Routing Techniques for Gate Array," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-2, No 4, Oct. 1983, pp. 301 - 312.

[Va] Vaidya, P. M., "An Algorithm for Linear Programming which Requires $O(((m+n)n^2 + (m+n)^{1.5} n))$ Arithmetic Operations," Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computation, May 1987, pp. 29 - 38.

[VK] Vecchi M. R. and S. Kirkpatrick, "Global Wiring by Simulated Annealing," IEEE Transactions on Computer-Aided Design, Vol. CAD-2, No. 4, Oct. 1983, pp. 215 - 222.

WORKING PAPER:
The Weighted Variable Power Problem

J. Barr and T. Gonzalez
UC Santa Barbara

May 15, 1996

1 Variable Power (Hu and Engel)

A *traffic matrix* D is an n by n matrix with non-negative entries. Entry $d_{i,j}$ represents the traffic from site i to site j. To simplify our notation we assume that $0 \leq i < n$, and $0 \leq j < n$. The total amount of *power* available is given by a positive number p The number of *links* that can be active simultaneously is a positive integer c (channels). A (c,p) *switching matrix* (cpSM) is an n by n matrix of nonnegative numbers with at most one nonzero entry in each row or column and at most c nonzero entries altogether. The sum of the entries of a cpSM must not exceed p, i.e., $\sum d_{i,j} \leq p$. A cpSM switching matrix represents the traffic that can be carried during one time slot. The number of *time slots* is a positive integer t.

The *Variable Power* (VP) problem is defined as follows, given p, c, t, and D, to determine whether or not there exist cpSM switching matrices $S_1, S_2, \ldots S_t$ such that $D = \sum_{k=1}^{t} S_k$.
Hu and Engel showed that the VP problem is NP-Complete, by reducing 3-partition to it.

3-partition
INSTANCE: A finite set A of $3m$ elements, a bound $B \in Z^+$, and a size $s(a) \in Z^+$ for each $a \in A$, such that, $B/4 < s(a) < B/2$, and $\sum s(a) = mB$
QUESTION: Can A be partitioned into m disjoint sets $A_i, A_2, \ldots, A_m$ such that $\sum_{a \in A_i} s(a) = B > 2$ for $1 \leq i \leq m$? Note that each of the sets in a 3-partition must have exactly there elements.

Theorem 1.1 *The VP problem is NP-complete.*

Proof: It is simple to show that VP is in NP. We now give Hu and Engel's polynomial reduction from 3-partition to VP. Given an instance of 3-partition I3P we construct the instance IVP of VP as follows.

Let D be a $3m$ by $3m$ diagonal matrix with diagonal entries $s(a_1), s(a_2), \ldots, s(a_{3m})$, let $p = B$ be the power constraint, let $t = m$ be the time constant, and let $c = 3$ be the channel constraint. The correspondence of these two problems is obvious.

□

Definition 1.1 *A star network is a network configuration in which a single site (the hub) communicates with all other sites, and each site communicates with the hub.*

A *star variable power* (SVP) problem is the VP problem restricted to all entries in the D matrix being zero except of the ones in row 0 and in column 0.

Theorem 1.2 *The star variable power problem is NP-complete.*

Proof: It is simple to show that SVP is in NP. We now give Hu and Engel's polynomial reduction from 3-partition to SVP.

Given an instance of 3-partition I3P we construct the instance ISVP of SVP as follows.

Let D be a $3m+1$ by $3m+1$ matrix whose entries $d_{i,j}$ are all zero except:

$$d_{0,j} = B \text{ for } 1 \leq j \leq m$$

$$d_{i,0} = B - s(a_i) \text{for } 1 \leq i \leq 3m$$

So, row 0 of D has exactly m nonzero entries, all equal to B. Column 0 of D has exactly $3m$ nonzero entries. All other entries are zeroes, so D is the traffic matrix of a star network. Let $p = B$ be the power constraint. Let $t = 3m$ be the time constant. Let $c = 2$ be the channel constraint.

We now show that ISVP can be scheduled in t time slots iff I3P has a 3-partition. The proof is in two parts.

(a) If I3P has a 3-partition then ISVP has a schedule.

Let $A_1, A_2, \ldots, A_m$ be a 3-partition for I3P, i.e., $A_1, A_2, \ldots, A_m$ is a partition of A into three elements subsets such that the sum of the sizes of the element in each set sums up to exactly B. For each set A_i we construct three switching matrices. For $A_i = \{a, b, c\}$ the first switching matrix has nonzero entries $d_{a,0} = B - s(a)$ and $d_{0,i} = s(a)$, the second switching matrix has nonzero entries $d_{b,0} = B - s(b)$ and $d_{0,i} = s(b)$, and the third switching matrix has nonzero entries $d_{c,0} = B - s(c)$ and $d_{0,i} = s(c)$, Clearly each of the three switching matrices satisfies the power constraint, and the channel constraint. Therefore the concatenation of the m switching matrices constructed from the A_is also satisfies the time constraint and forms a schedule for ISVP.

(b) If ISVP has a schedule, then the I3P has a 3-partition.

Clearly, the sum of all the entries in D is $3mB$. Since $t = 3m$ and $p = B$ it must be that each switching matrix must sum up to B. Since there are $3m$ positive entries in column zero and one can use at most one such entry in each switching matrix, it follows that each entry in column zero must be in exactly one switching matrix. Reorder the switching matrices so that the one for that uses entry $d_{i,0}$ is the i^{th} one. We say that the i^{th} switching matrix represents element a_i. Since $p = B$, each switching matrix matrix must sum to B and one can only use one entry in row zero in each switching matrix, it then follows that exactly one entry in row 0 in the i_{th} switching matrix must be nonzero and has the value $s(a_i)$. Since $D = \sum_{k=1}^{t} S_k$, it must be that the switching matrices that contribute to $d_{0,j}$ must contribute exactly B units, and and the elements a_i these switching matrices represent sum to B. Therefore, A has a 3-partition.

□

2 Weighted Variable Power (New Result)

The Weighted Variable Power (WVP) Problem is a generalization of the VP problem. A *traffic matrix* D is an n by n matrix with non-negative entries. Entry $d_{i,j}$ represents the traffic

from site i to site j. To simplify our notation we assume that $0 \leq i < n$, and $0 \leq j < n$. A *weight matrix* W is an n by n matrix with positive entries. Entry $w_{i,j}$ represents the amount of power needed to transfer one unit of traffic the from site i to site j. To simplify our notation we assume that $0 \leq i < n$, and $0 \leq j < n$. The total amount of *power* is given by a positive number p The number of *links* that can be active simultaneously is a positive integer c (channels). A (c, p) *switching matrix* (cpSM) is an n by n matrix of nonnegative numbers with at most FOUR nonzero entry in each row or column and at most c nonzero entries altogether. The total (weighted) amount of power used must not exceed p, i.e., $\sum w_{i,j} \cdot d_{i,j} \leq p$. A cpSM switching matrix represents the traffic that can be carried during one time slot. The number of time *slots* is a positive integer t.

The *Weighted Variable Power* (WVP) problem is, given p, c, t, W, and D, to determine whether or not there exist integer valued cpSM switching matrices $S_1, S_2, \ldots S_t$ such that $D = \sum_{k=1}^{t} S_k$.

The differences between the VP and the WVP problem are: the power constraint is weighted ($\sum w_{i,j} \cdot d_{i,j} \leq p$ rather than just $\sum d_{i,j} \leq p$), the switching matrices are integer valued (rather than real values), and there can be four nonzero entries in each row or column in each switching matrix rather than just one. We show that the WVP problem is NP-Complete. We reduce 3-partition to WVP.

Theorem 2.1 *The WVP problem is NP-complete.*

Proof: It is simple to show that WVP is in NP. We now give polynomial reduction (identical to the one by Hu and Engel) from 3-partition to WVP.

Given an instance of 3-partition I3P we construct the instance IWVP of WVP as follows. Let D be a $3m$ by $3m$ diagonal matrix with diagonal entries $s(a_1), s(a_2), \ldots, s(a_{3m})$, and all the entries in the W matrix have value 1. Let $p = B$ be the power constraint, let $t = m$ be the time constant, and let $c = 3$ be the channel constraint. The correspondence of these two problems is obvious.

□

We now show that the problem remains NP-complete even when the network configuration is a star network.

Theorem 2.2 *The star weighted variable power problem is NP-complete.*

Proof: It is simple to show that SWVP is in NP. We now give a polynomial reduction from 3-partition to SWVP similar to the one by Hu and Engel for the SVP.

Given an instance of 3-partition I3P we construct the instance ISWVP of SWVP as follows. Let D be a $12m + 1$ by $12m + 1$ matrix whose entries $d_{i,j}$ are all zero except:

$$d_{0,j} = B \text{ for } 1 \leq j \leq 10m$$

$$d_{i,0} = B - s(a_i) \text{ for } 1 \leq i \leq 3m$$

$$d_{i,0} = B \text{ for } 3m + 1 \leq i \leq 12m$$

Column zero and row zero for the weight matrix W have the following entry values and the rest are not important.

$$d_{0,j} = 1 \text{ for } 1 \leq j \leq m$$

$$d_{0,j} = B^2 \text{ for } m+1 \leq j \leq 10m$$

$$d_{i,0} = 1 \text{ for } 1 \leq i \leq 3m$$

$$d_{i,0} = B^6 \text{ for } 3m+1 \leq i \leq 12m$$

So, row 0 of D has exactly $10m$ nonzero entries, all equal to B. The first m entries have weight 1, and the remaining ones have weight B^2. Column 0 of D has exactly $12m$ nonzero entries. The first $3m$ entries have weight 1, and the remaining ones have weight B^6. All other entries are zeroes, so D is the traffic matrix of a star network. Let $p = 3B^7 + 3B^3 + B$ be the power constraint. Let $t = 3m$ be the time constant. Let $c = 8$ be the channel constraint (this will always be satisfied because there is only one row and column in D with nonzero values). We now show that ISWVP can be scheduled in t time slots iff the I3P has a 3-partition. The proof is in two parts.

(a) If I3P has a 3-partition than ISWVP has a schedule.

Let $A_1, A_2, \ldots, A_m$ be a 3-partition for I3P, i.e., $A_1, A_2, \ldots, A_m$ is a partition of A into three elements subsets such that the sum of the sizes of the element in each set sums up to exactly B. For each set A_i we construct three switching matrices. For $A_i = \{a, b, c\}$ the first switching matrix has the following nonzero entries, and the corresponding weights are given below.

$d_{a,0} = B - s(a)$	$d_{m+9(i-1)+1,0} = B$	$d_{m+9(i-1)+2,0} = B$	$d_{m+9(i-1)+3,0} = B$
$w_{a,0} = 1$	$w_{m+9(i-1)+1,0} = B^6$	$w_{m+9(i-1)+2,0} = B^6$	$w_{m+9(i-1)+3,0} = B^6$
$d_{0,i} = s(a)$	$d_{0,3m+9(i-1)+1} = B$	$d_{0,3m+9(i-1)+2} = B$	$d_{0,3m+9(i-1)+3} = B$
$w_{0,i} = 1$	$w_{0,3m+9(i-1)+1} = B^2$	$w_{0,3m+9(i-1)+2} = B^2$	$w_{0,3m+9(i-1)+3} = B^2$

the second switching matrix has nonzero entries

$d_{b,0} = B - s(b)$	$d_{m+9(i-1)+4,0} = B$	$d_{m+9(i-1)+5,0} = B$	$d_{m+9(i-1)+6,0} = B$
$w_{b,0} = 1$	$w_{m+9(i-1)+4,0} = B^6$	$w_{m+9(i-1)+5,0} = B^6$	$w_{m+9(i-1)+6,0} = B^6$
$d_{0,i} = s(b)$	$d_{0,3m+9(i-1)+4} = B$	$d_{0,3m+9(i-1)+5} = B$	$d_{0,3m+9(i-1)+6} = B$
$w_{0,i} = 1$	$w_{0,3m+9(i-1)+4} = B^2$	$w_{0,3m+9(i-1)+5} = B^2$	$w_{0,3m+9(i-1)+6} = B^2$

the third switching matrix has nonzero entries

$d_{c,0} = B - s(c)$	$d_{m+9(i-1)+7,0} = B$	$d_{m+9(i-1)+8,0} = B$	$d_{m+9(i-1)+9,0} = B$
$w_{c,0} = 1$	$w_{m+9(i-1)+7,0} = B^6$	$w_{m+9(i-1)+8,0} = B^6$	$w_{m+9(i-1)+9,0} = B^6$
$d_{0,i} = s(c)$	$d_{0,3m+9(i-1)+7} = B$	$d_{0,3m+9(i-1)+8} = B$	$d_{0,3m+9(i-1)+9} = B$
$w_{0,i} = 1$	$w_{0,3m+9(i-1)+7} = B^2$	$w_{0,3m+9(i-1)+8} = B^2$	$w_{0,3m+9(i-1)+9} = B^2$

Multiplying the weights by the transmission we know that the power used is $B+3B^7+3B^3$ which is equal to p. So, each of the three switching matrices satisfies the power constraint, and the channel constraint. Therefore the concatenation of the $3m$ switching matrices also satisfies the time constraint and is a schedule for ISWVP.

(b) If ISWVP has a schedule, then the I3P has a 3-partition.

We now claim that a feasible schedule must have each of its switching matrices with at most $3B$ transmissions with weight B^6. Suppose not, suppose that one such switching matrix has more than $3B$ of such transmissions. Then the power consumed by those entries is at least $3B^7 + B^6$ but that is greater than $p = B + 3B^7 + 3B^3$ because B is at least 2. Therefore every switching matrix in a feasible schedule must have at most $3B$ entries with weight B^6. Since the sum of all the entries in D with weight B^6 is $9mB$, and $t = 3m$, it must be that all the switching matrices in a feasible schedule have exactly $3B$ transmissions with weights B^6.

At this point one can use similar arguments to show that all the switching matrices in a feasible schedule have exactly $3B$ transmissions with weights B^2. Similarly, one can show that all the switching matrices in a feasible schedule have exactly B transmissions with weights 1.

Since there can be at most four different transmissions in each row and in each column, it must then be that each switching matrix in a feasible schedule has exactly three transmissions of B units each with weight B^6, exactly three transmissions of B units each with weight B^2, and two transmissions (one in column zero and one in row zero) of a total of B units with weight 1. Reorder the switching matrices so that the one for that uses entry $d_{i,0}$ is the i^{th} one. We say that the i^{th} switching matrix represents element a_i. Since $D = \sum_{k=1}^{t} S_k$, it must be that the switching matrices that contribute to $d_{0,j}$ must contribute exactly B units, and the elements a_i that these switching matrices represent sum to B. Therefore, A has a 3-partition.

□