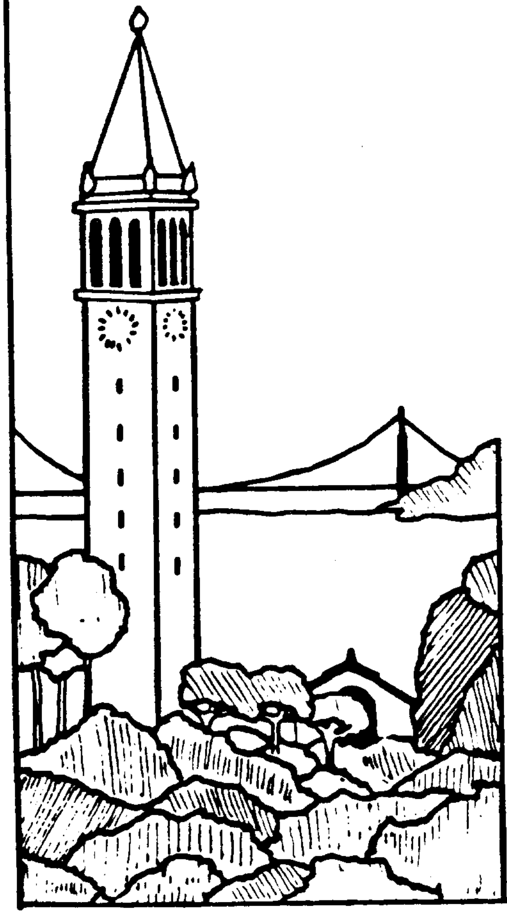
Randomized Rounding And Discrete Ham-Sandwich Theorems: Provably Good Algorithms For Routing And Packing Problems

Prabhakar Raghavan



Report No. UCB/CSD 87/312

July 1986

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Randomized Rounding And Discrete Ham-Sandwich Theorems: Provably Good Algorithms For Routing And Packing Problems

Prabhakar Raghavan

Report No. UCB/CSD 87/312

July 1986

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Randomized Rounding and Discrete Ham-Sandwich Theorems: Provably Good Algorithms for Routing and Packing Problems

Prabhakar Raghavan

ABSTRACT

This thesis deals with the approximate solution of a class of zero-one integer programs arising in the design of integrated circuits, in operations research, and in some combinatorial problems. Our approach consists of first relaxing the integer program to a linear program, which can be solved by efficient algorithms. The linear program solution may assign fractional values to some of the variables, and these values are 'rounded' to obtain a provably good approximation to the original integer program.

We first consider the problem of global routing in gate-arrays. This problem has important applications in the design of integrated circuits, and can be formulated as a zero-one integer program. We introduce a technique we call randomized rounding for producing a provably good approximation to this integer program from the solution to its relaxation. In order to prove the quality of this approximation, we make use of some new bounds on the tail of the binomial distribution. We present the results of experiments conducted on industrial gate-arrays using our methods; these are encouraging and call for further work.

We then show that our randomized rounding technique can be applied to some problems in combinatorial optimization and operations research. We also describe the relation between the problems we study and a class of combinatorial results known as "discrete ham-sandwich theorems". This leads to the problem of rounding linear program solutions deterministically in polynomial time. We invoke an interesting "method of conditional probabilities" for this purpose. An extension of this method shows that it is possible to deterministically mimic the randomized algorithm in a certain precise sense. This leads us to the development of a deterministic polynomial time rounding algorithm that yields the same performance guarantees as the randomized method.

Table of Contents

Chapter 1
Introduction

1.1 Overview

Integer programming is a fundamental combinatorial problem with a multitude of practical applications. The general $0-1$ integer programming problem is known to be NP-Complete [17]. Although there is no efficient algorithm known for solving an arbitrary integer programming problem, a number of special cases can be solved either approximately or exactly in polynomial time. This thesis deals with the approximate solution of a class of integer programs arising in the design of integrated circuits, operations research, and some combinatorial problems.

In an integer programming problem Π_I we are given a convex polytope P_I in $\mathbb{R}^d$; P_I is specified as the intersection of several half-spaces defined by linear inequalities. We are to find a point in the polytope that minimizes the value of a linear objective function. Furthermore, the point we select must be a *lattice point*, i.e. it must have all co-ordinates integral.

If the restriction of integrality were removed, we would obtain the *relaxation linear programming* problem Π_R. We know of polynomial-time algorithms for the linear programming problem [16,19]. It is natural to ask what information about Π_I can be derived from the solution to Π_R. In particular, it is interesting to see whether the solution to Π_R can be efficiently used to find a lattice point in P_I. If we could thus find a lattice point with an objective function value close to the optimal solution to Π_I, we have found an *approximation* to the optimal solution.

In general, the solution to Π_R is not known to give enough information to construct an approximate solution to Π_I. Indeed, the value of the objective function at the optimum of Π_R can be arbitrarily far from the corresponding value for Π_I (see, for example, page

309 of reference [31]). However, in a number of specific cases, it is easy to make use of the solution to Π_R to approximately solve Π_I.

In this thesis we study some integer programs through the relaxation approach. We show in these cases that the solution to the relaxation linear program leads to an approximate solution to the original integer program.

All the integer programs we consider will be $0-1$ integer programs, in which all coordinates of admissible lattice-points (solutions) are constrained to be 0 or 1. These coordinates are known as the *variables* of the integer program. In the solution to the relaxation Π_R, all variables will thus assume values in the interval [0,1]. Our problem then is to "round" each of these fractional variables to 0 or 1 so as to obtain an integer solution with an objective function value close to the optimum of Π_I.

Our emphasis in this thesis will be on the quality of these approximations, rather than on the exact running time of our approximation algorithms. For our purposes, we will be satisfied with showing that the procedures we develop have running times that are polynomial in the size of the input. To do so, we will implicitly use the fact that the linear programming algorithm of Karmarkar [16] is polynomial-time. We then need only show that our algorithms for rounding the linear program solutions run in polynomial-time. This will usually be obvious from the descriptions of our algorithms.

1.2 Organization of the thesis

In chapter 2, we introduce the problem of global routing in gate arrays. This problem has important applications in the design of integrated circuits, and is therefore a problem of considerable practical interest. The global routing problem is proved NP-Complete at the end of the chapter, paving the way for the search for approximation algorithms.

Chapter 3 begins with a formulation of global routing as a $0-1$ integer linear program. We then discuss strategies for rounding the solution of the relaxation linear program, and introduce a procedure we call *randomized rounding* as a means of producing a provably good approximation to the integer program. In order to prove the quality of the solution proved by randomized rounding, we make use of some new bounds on the tail of the binomial distribution.

In the latter half of chapter 3, we describe experiments conducted on gate arrays using our methods. We present the results of these experiments, which are encouraging and call for further work.

In chapter 4 we show that our randomized rounding technique can be applied to some problems in combinatorial optimization and operations research. Chapter 5 surveys some combinatorial results related to our rounding methods, and leads to the study of removing the randomness from our rounding methods.

In chapter 6 we study the problem of rounding linear program solutions deterministically. We invoke an interesting "method of conditional probabilities" for this purpose. This leads us to the development of a deterministic polynomial time rounding algorithm that achieves precisely the same performance guarantees as the randomized method.

We conclude by summarizing our main results and noting directions for further work, in chapter 7.

1.3 Notation

Throughout this thesis, $\ln x$ will denote the natural logarithm of x, while $\log x$ will denote the logarithm of x to the base 2. Aside from this, we use the standard notation used in algorithmic analysis [20].

Chapter 2
Global Routing in Gate Arrays

2.1 A Description of Gate Arrays

Gate arrays are a popular vehicle for the semi-custom design of integrated circuits. A gate array is a two-dimensional array of gates (figure 2.1) arranged in rectilinear fashion. A large number of such chips are fabricated by a manufacturer. A customer who wishes to build a logic circuit decides on a mapping of the gates in her circuit onto the gates in the array. The customer specifies the interconnections to be made between the gates in the array to realize the circuit she has in mind. In figure 2.2, for instance, the gates numbered 1 are to be connected together, as are the gates numbered 2, and so on. The manufacturer then makes the necessary connections by running wires over the gates on the chip, producing a routed chip. Figure 2.3 shows a possible routing for the connections required in figure 2.2.

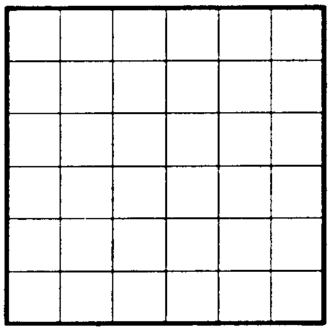


Figure 2.1

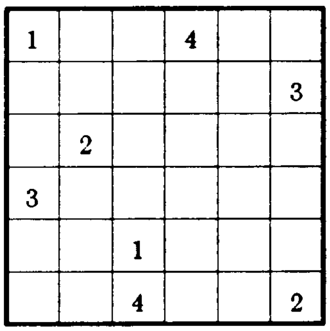


Figure 2.2

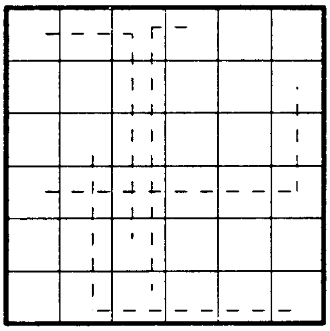


Figure 2.3

We now give an informal description of the routing problem. A formal model of the routing process will be stated in section 2.3. A convenient algorithmic abstraction is to view each gate in the array as a square, as in figure 2.1. The array may thus be thought of as being composed of abutting squares. This assumption is fairly close to practical gate arrays; the reader interested in physical and technological aspects is referred to [3].

Wires between gates run parallel to the axes, and pass over the boundaries between gates. These boundaries - known as *channels* - have fixed widths determined by the sizes of the gates; each channel can thus permit no more than a certain number of wires to be routed through it. Since the manufacturer builds the arrays with no *a priori* knowledge of the customer's circuit, these channel widths - or *capacities* - are fixed in advance. Let us suppose for the moment that the gates of the customer's circuit have been mapped on to gate locations on the array. Such a mapping - known as the *placement* of the circuit onto the array - defines sets of gates to be interconnected by routing wires through channels. A set of gates to be interconnected in this fashion is known as a *net*.

The routing problem has traditionally been viewed as consisting of two phases. The first is *global routing*, where for each net a pattern of routes through channels is selected, subject to the constraint that the number of routes passing through any channel does not exceed its capacity. Each channel physically consists of a number of parallel *tracks* in which wires are actually laid down; the number of tracks in a channel is equal to its capacity. Note that a global routing does not specify the manner in which routes are arranged amongst the tracks in a channel. Given a global routing, the *detailed routing* is an assignment of a track to each route passing through a channel. This assignment is specified for all channels, and must satisfy certain consistency conditions at the junctions of channels.

In this thesis, we will concern ourselves only with the global routing problem. We will thus assume that we are given a placement of the logic circuit; and that detailed routing follows any global routing we might generate. In reality, the placement process and the two routing phases are closely related – a bad placement might make it impossible to produce a feasible global routing. Likewise, (although this happens somewhat less frequently in practice), a bad global routing might make it impossible to produce a detailed routing.

2.2 Related Work

2.2.1 The Placement Problem

The placement problem is closely related to graph partitioning and other similar problems widely believed to be intractable [13]. Furthermore, it is difficult to characterize precisely a "good" placement - a good placement must permit easy routing, keep highly-connected portions of the circuit in physical proximity (to avoid many long interconnection wires on the chip), and avoid excessive congestion in any particular region of the chip. A number of heuristics are employed in practice [8,32]. The actual heuristic used depends on the application at hand and is governed by such factors as the type of circuits implemented by customers, the size of the array, and the technology used.

2.2.2 Routing Algorithms and Heuristics

An early effort in the area of routing is due to Lee [24]. Lee's algorithm was the prototype of the "maze" style of routing algorithm. Here nets are handled one at a time; after a net is routed the prevailing congestion in the channels of the array is taken into account before the next net is routed. Each net executes a walk through the "maze" created by the nets already routed.

The success of the Lee algorithm is very sensitive to the order in which the nets are processed. In general, it could happen that the Lee algorithm encounters a bottleneck before all the nets are routed, even though the problem instance may have a feasible solution. At this point it is necessary to enter a "rip-up-and-reroute" or backtrack phase in which some of the nets routed so far are removed and the routing process restarted using a different ordering of the nets.

Two aspects of Lee's algorithm are unsatisfactory from a theoretical standpoint. The first is that it is a backtrack search procedure with a running time that can only be bounded by a function that is exponential in the size of the input. Secondly, it is not known whether the Lee algorithm will find in polynomial time a solution close to the optimal one. The algorithms we develop in the next chapter will address these issues.

So far, we have viewed the global routing problem as a feasibility problem. It is possible to view the problem instead as an optimization problem, as follows. Suppose that every gate in the array were a square of side C, i.e. each channel has capacity C. (The assumption that a gate is a square is not critical - we could, for instance, speak of a rectangle of a given aspect ratio). Given an instance of the routing problem, we could ask for the minimum value of C for which we can find a feasible routing. Note that this optimization problem is at least as hard as the feasibility problem.

Burstein and Pelavin [5] used a dynamic programming approach to present an approximation algorithm for the optimization problem. Subsequently, Karp *et al.* [18] used a linear-programming approach together with a divide-and-conquer process to develop an algorithm that provided a provably good approximate solution. In particular, they showed that if $C^{(I)}$ were the optimal channel capacity for a problem instance, their method would find a routing that required channels of capacity $O\,(C^{(I)} \cdot \log \frac{n}{C^{(I)}})$. Their work motivated much of the work in this thesis, and the integer programming formulation we develop is section 2.4 bears a strong resemblance to theirs. Their model of channels and channel capacity is however considerably different from ours, which we define below in section 2.3.

Hu and Shing [15] also proposed a linear program formulation of the global routing problem. Their formulation uses column generation techniques in the simplex method [7]. The algorithm they propose does not address an important feature of linear program solutions which we will describe in detail in section 2.4. More recently, the *simulated annealing* technique has been applied to the problem by Vecchi and Kirkpatrick [38].

2.3 A Model for Routing in Regular Arrays

In this section we introduce a formal model for routing in two dimensional regular arrays, which are an abstraction of gate arrays.

(M1) A *regular array* is a two-dimensional $n \times n$ lattice $L(V,E)$ where the lattice nodes represent gates, and the edges between nodes represent channel through which nets can be routed. A *net* is a set of nodes that are to be connected.

(M2) An *instance of the routing problem* is a set R of nets.

(M3) A *connection* between two nodes v_a and v_b, for $v_a, v_b \in E$, is a simple path $\{e_1, e_2, \ldots, e_k\}$, $e_i \in E$ *for* $1 \leq i \leq k$ where e_1 is incident on v_a and e_k is incident on v_b. Such a connection is said to be of *length* k. 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.

(M4) A *solution* to the routing problem is a set S of trees such that every net is routed.

(M5) The *width of an edge* is the number of trees in S that use the edge. The *width of a solution* is the maximum width of an edge taken over all edges in the lattice.

(M6) An *optimal solution* is one whose width is the least among all solutions.

Assumption M1 contracts each gate in an array to a node in a lattice. We are only interested in the boundaries of each gate and their capacity to accommodate routes. The internal structure of the gates is thus unimportant, permitting us to abstract gates by nodes. The rectangular shape of gates now translates naturally to four edges in the lattice L each of which represents a channel. The capacity of a channel is abstracted in M5 as the width of the corresponding edge in E. The width of a solution identifies the "bottleneck" - the channel(s) on an array most congested by the solution. For example, the solution in figure 2.3 is of width 2.

Clearly the model stated above is not specific to a lattice - we could define a routing problem in any graph $G(V,E)$ rather than just the lattice. Indeed, the algorithms we will develop in chapter 3 are applicable to any graph.

2.4 The Complexity of Global Routing

We introduced the global routing problem as a feasibility problem, or a *decision problem*. This decision problem is interesting for complexity-theoretic classification. We now state the decision problem formally, using the terminology of the model in the previous section.

GLOBAL ROUTING

INSTANCE: Given a lattice $L(V,E)$, a set R of nets, and a positive integer C.

QUESTION: Does there exist a set S of trees in L such that every net is routed and no edge of E has width exceeding C?

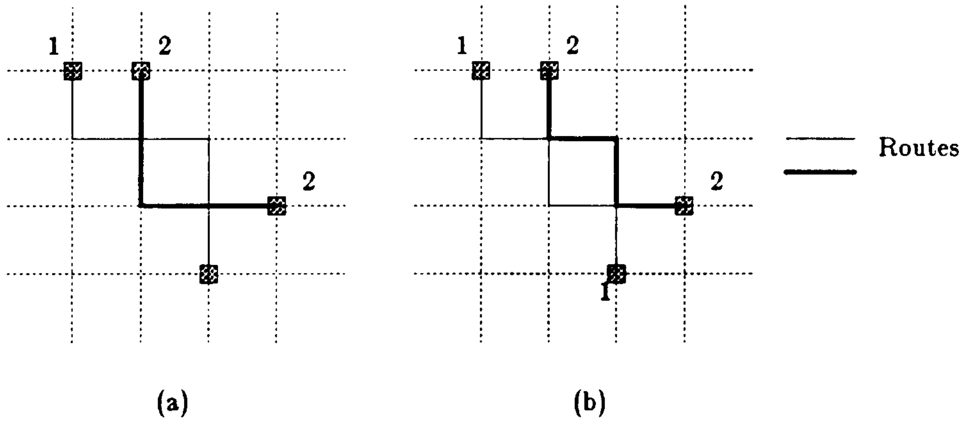


Figure 2.4: Differences in routing models.

Kramer and van Leeuwen [21] showed that the related problem of WIRE ROUTING is NP-Complete. The WIRE ROUTING problem differs from GLOBAL ROUTING only in the wiring model. In their model two distinct connections can pass through a node only in orthogonal directions, i.e. one connection must pass through the node vertically and the other horizontally. In our model, we also allow "knock-knees" at a node: figure 2.4 illustrates the difference. Both parts of figure 2.4 constitute legal routings in our lattice model. In the Kramer-van Leeuwen model, however, the routing in part (b) is not legal, since it makes use of knock-knees.

The manner in which Kramer and van Leeuwen prove the WIRE ROUTING problem NP-Complete is by reducing 3-SAT to an intermediate problem called OBSTACLE

ROUTING, and then showing that OBSTACLE ROUTING reduces to WIRE ROUTING.

DEFINITION: An *obstacle* in a lattice is a rectangular sub-array of lattice nodes.

We now define the LATTICE OBSTACLE ROUTING problem, which differs from the OBSTACLE ROUTING problem of Kramer and van Leeuwen only in that we allow knock-knees in our routing model.

LATTICE OBSTACLE ROUTING

INSTANCE: Given a lattice $L(V,E)$, a set R of nets, a set of L_O of obstacles in L, and a positive integer C.

QUESTION: Under the routing model of section 2.3, does there exist a solution-set S of trees in L containing no node of L_O, such that every net is routed and no edge of L has width exceeding C?

LEMMA 2.1: LATTICE OBSTACLE ROUTING is NP-Complete.

PROOF: From the reduction of Kramer and van Leeuwen [21]. Their reduction holds even when knock-knees are allowed.

Suppose that in our model for routing in a two-dimensional lattice (section 2.3), we had the ability to create "obstacles" in the lattice, i.e. create rectangular regions in the lattice through which no wire can be routed. We then have an instance of OBSTACLE ROUTING. Thus if we could show that in our model it is possible to create rectangular obstacles in the lattice, we could embed an arbitrary instance of OBSTACLE ROUTING in an instance of GLOBAL ROUTING.

Suppose the input R contains C nets between a node v in the lattice and each of its four neighbors. It is clear that all the edges incident on v are "blocked" in that no other net can use these edges for its routing. Now consider a rectangular array of nodes in the lattice L; this induces a sub-graph L_s. If the input R contained C nets joining each node in L_s to each of its neighbors, it is clear that no other net in the global routing problem can use any edge of L_s. Rectangular regions of the form of L_s create the necessary obstacles to embed an instance of OBSTACLE ROUTING in an instance of GLOBAL ROUTING.

THEOREM 2.2 : GLOBAL ROUTING is NP-Complete.

PROOF : By reduction from OBSTACLE ROUTING.

Theorem 2.2 tells us that unless P=NP, there is little hope of finding a polynomial-time algorithm that solves the optimization problem exactly. Indeed, the algorithms we will develop in the next chapter will be approximation algorithms that can only guarantee finding a solution close to the optimum.

Chapter 3
Randomized Rounding

3.1 Overview

We begin this chapter by formulating the global routing problem as an integer linear programming problem. We then introduce a probabilistic technique for approximately solving such integer programs. The Chernoff bound on the tail of the sum of independent Bernoulli trials will be used to prove the quality of this approximation algorithm.

The problem of integer multicommodity flow will then be shown to be closely related to a special case of the routing problem. We show how our algorithm may be adapted to this case as well, using a random walk procedure. We conclude the chapter by presenting the results of some experiments on gate-arrays obtained from industrial sources.

3.2 Formulation as an Integer Program

In this section we show that the global routing problem reduces to solving an integer linear program in which the variables assume only the values 0 or 1. We begin by considering a restricted version of the global routing problem, in which each net must be routed using one of a small set of trees (or *configurations*). Such a restriction is often desirable from a practical point of view, in order to control the quality of the routing produced. For instance, a restricted set of configurations may be used to preclude circuitous routes that result in unduly long connections between nodes. Long connections degrade both the speed and the reliability of the finished chip, and are thus to be avoided.

Thus for each net $r_i \in R$ we are given a set $T(r_i)$ of configurations that may be used to route r_i. Let t_{ij} be the j^{th} configuration in $T(r_i)$. For each t_{ij}, let x_{ij} be an indicator $(0-1)$ variable that denotes the presence or absence of t_{ij} in the solution set S. We write constraints to ensure that every net in R is routed:

$$\sum_j x_{ij} = 1 , \quad \forall \ i \tag{3.1}$$

Next, we ensure that no more than C routes pass through any edge in E:

$$\sum_{e \in t_{ij}}^{\prime} x_{ij} \leq C \;, \quad \forall \; e \in E \tag{3.2}$$

Subject to these constraints, our integer program objective is to

$$\text{Minimize} \quad C \;, \quad x_{ij} \in \{0,1\} \tag{3.3}$$

3.2.1 The Relaxation Linear Program

In view of theorem 2.1, this integer programming problem is NP-Hard. Our approach consists of first solving a linear program relaxation in which the variables x_{ij} are allowed to assume fractional values:

$$\text{Minimize} \quad C \;, \quad x_{ij} \in [0,1] \tag{3.4}$$

This is a linear programming problem, for which a number of algorithms are known [7,16]. In particular, the algorithms of Karmarkar [16] and Khachian [19] are polynomial-time algorithms. Solving the above linear programming problem assigns to each variable a value x^*_{ij}, and an optimum width C^* for the objective function.

Because we have solved a relaxation linear program, the values x^*_{ij} and C^* may be fractional, and do not thus correspond to a physically meaningful solution to the global routing problem. It must be noted that C^* is a lower bound on the width $C^{(I)}$ of the optimum solution to the original integer program. The linear program solution routes each net by using 'fractional pieces' of several configurations. The significance of x^*_{ij} is that it represents the fraction of net r_i routed by configuration t_{ij} in the "fractional routing" generated by the linear program. In order to obtain a solution to the integer program (and thus a routing), we wish to 'round' the linear program results x^*_{ij} to $0-1$ values.

3.2.2 A Simple Rounding Strategy

We now examine a simple heuristic for rounding the fractional values x^*_{ij} to integer $(0-1)$ values. For each net r_i, we choose $k(i)$ so as to satisfy:

$$x^*_{ik(i)} \geq x^*_{ij} \ , \quad \forall \ j \tag{3.5}$$

Ties are broken arbitrarily in the choice of $k(i)$. In this heuristic, which we call MAXIMUM, net r_i is routed using configuration $t_{ik(i)}$. Thus, for each net r_i we are picking a configuration $t_{ik(i)}$ that has contributed the most to its fractional routing. We need one further condition to prove a performance guarantee for the MAXIMUM heuristic for rounding. Let the sets of allowed routes $T(r_i)$ be bounded in size:

$$| \ T(r_i) \ | \ \leq \ B \ , \quad \forall \ i \tag{3.6}$$

THEOREM 3.1: Let $C^{(I)}$ be the width of the optimum solution. The routing produced by MAXIMUM has width no more than $B \cdot C^{(I)}$.

PROOF: By (3.6),

$$x^*_{ik(i)} \geq \frac{1}{B} \ , \quad \forall \ i \tag{3.7}$$

By constraint (3.2),

$$\sum_{e \in t_{ij}} x^*_{ij} \leq C^* \ , \quad \forall \ e \in E \tag{3.8}$$

From these it follows that for each $e \in E$,

$$| \ \{ \ i : e \in t_{ik(i)} \ \} \ | \ \leq \ \frac{C^*}{1/B} \ \leq \ B \cdot C^* \tag{3.9}$$

The theorem follows from our earlier observation that C^* is a lower bound $C^{(I)}$. □

3.3 Randomized Rounding

The bound in theorem 3.1 is not particularly good, especially when B is large. We now introduce a more sophisticated strategy for rounding the x^*_{ij}, which we call "randomized rounding". Randomized rounding will be the main algorithmic technique we will use in this chapter and the next. The method is probabilistic, and is applied independently to each net r_i.

The technique is as follows. For each i, independently, we use t_{ij} to route r_i with probability x^*_{ij}. In other words, we interpret the linear program values for the indicator variables as probabilities for using the corresponding configurations. For each net r_i, exactly one configuration is chosen by making the probabilistic choice mutually exclusively among the t_{ij} (conceptually, we are casting a biased $|T(r_i)|$-faceted die with face probabilities x^*_{ij}).

Randomized rounding has some intuitively appealing properties. For instance, a configuration with a high contribution x^*_{ij} to the fractional routing is more likely to be chosen to route r_i. From a more rigorous standpoint, we can prove that randomized rounding achieves a routing of width very close to the optimum width. The theorem on the performance guarantee of randomized rounding involves some notions from probability theory which we develop in the next section. The theorem and its proof will be given in section 3.5.

3.4 The Chernoff Bound

We now derive certain forms of the Chernoff bound on the tail of the binomial distribution. These bounds will be used to prove the performance of randomized rounding. In addition, the general principles used in its derivation will prove useful in chapters 4 and 5. The reader is referred to [4] for a general treatment of the theory of moment-generating functions and Chernoff-type bounds. The material below is entirely self-contained.

Let $X_1, X_2, \ldots, X_r$ be independent, identically distributed (or *i.i.d.*) $0-1$ random variables. Each random variable assumes the value 1 with probability p, and the value 0 with probability $1-p$. Let

$$S = \sum_{i=1}^{r} X_i \tag{3.10}$$

A simple combinatorial argument shows that

$$\Pr[\, S = k \,] = \binom{r}{k} p^k (1-p)^{r-k} \;, \quad 1 \le k \le r \tag{3.11}$$

It is also clear that S has expectation rp. Chernoff's bound applies to deviations of S from its expectation. We first examine deviations of S above its expectation.

THEOREM 3.2: Let $\xi > 1$. Then

$$\Pr[\, S > \xi \cdot rp \,] < \left[\frac{e^{\xi - 1}}{\xi^{\xi}} \right]^{rp} \tag{3.12}$$

PROOF: The proof uses Chernoff's general technique [6] involving moment generating functions and the Markov inequality, and then bounds the specific form obtained for the binomial distribution. For any positive real number t,

$$\Pr[\, S > \xi \cdot rp \,] = \Pr[\, e^{tS} > e^{t \cdot \xi \cdot rp} \,] \le \frac{\mathrm{E}[\, e^{tS} \,]}{e^{t \cdot \xi \cdot rp}} \tag{3.13}$$

The last inequality is the Markov inequality. Since the X_i and thus e^{tX_i} are independent, the right hand side of the above is

$$\frac{\mathrm{E}[\, e^{tX_i} \,]^r}{e^{t \cdot \xi \cdot rp}} = e^{-t \cdot \xi \cdot rp} \cdot [\, p \cdot e^t + 1 - p \,]^r < e^{-t \cdot \xi \cdot rp} \cdot e^{rp(e^t - 1)} \tag{3.14}$$

The last inequality is obtained using $1 + x \le e^x$; it is strict since we assume $p > 0$ and will use $t > 0$. Let $t = \ln \xi$. From the above we have

$$\Pr[\, S > \xi \cdot rp \,] < \exp[\, rp\,(\, \xi - 1 - \xi \cdot \ln \xi \,)\,] = \left[\frac{e^{\xi - 1}}{\xi^{\xi}} \right]^{rp} \quad \square \tag{3.15}$$

REMARK: The bound in theorem 3.2 is *universal*, in that it holds for all $\xi > 1$. However, it is often desirable to have forms of (3.15) that are easy to invert, especially in the design of algorithms. We now examine some simplified versions of (3.15) that have this property. The following well-known version of Chernoff's bound follows immediately from (3.15).

COROLLARY 3.2.1:

$$\Pr[\, S > \xi \cdot rp \,] < \left[\frac{e}{\xi} \right]^{\xi rp} \tag{3.16}$$

REMARK: Obviously, corollary 3.2.1 is useful only for large deviations; indeed, it gives useful information only when $\xi > e$.

The following lemma will enable us to derive a simple upper bound on (3.15).

LEMMA 3.3: For all $\delta > 0$, the function

$$f(\delta) = \frac{(1+\delta)\cdot\ln(1+\delta) - \delta}{\delta^2} \tag{3.17}$$

is monotonically decreasing.

PROOF: We will show that the derivative of $f(\delta)$ is always negative for positive δ. To do so, we must prove that

$$\frac{\delta^2\cdot\ln(1+\delta) - 2\delta((1+\delta)\cdot\ln(1+\delta) - \delta)}{\delta^4} < 0\ , \quad \delta > 0 \tag{3.18}$$

or

$$\delta\cdot\ln(1+\delta) < 2(1+\delta)\cdot\ln(1+\delta) - 2\delta\ , \quad \delta > 0 \tag{3.19}$$

or

$$\frac{2\delta}{2+\delta} < \ln(1+\delta)\ , \quad \delta > 0 \tag{3.20}$$

At $\delta = 0$ the two functions above are equal. We show that for all $\delta > 0$ the latter function grows strictly faster. Taking derivatives, we are reduced to having to show that

$$\frac{4}{(2+\delta)^2} < \frac{1}{1+\delta}\ , \quad \delta > 0 \tag{3.21}$$

which follows from the fact that

$$4 + 4\delta < 4 + 4\delta + \delta^2\ , \quad \delta > 0 \quad \square \tag{3.22}$$

Returning to our bound of theorem 3.2, we re-write it with $\xi = 1+\delta$; thus δrp represents the deviation (above) of S relative to its mean. For $\delta > 0$, (3.15) becomes

$$\Pr[S - rp > \delta\cdot rp] < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{rp} \tag{3.23}$$

$$= \exp\left[-rp\delta^2\left(\frac{(1+\delta)\cdot\ln(1+\delta) - \delta}{\delta^2}\right)\right] \tag{3.24}$$

COROLLARY 3.2.2: For any $\Delta > 0$

$$\Pr[S - rp > \delta\cdot rp] < \exp[-f(\Delta)\cdot rp\delta^2]\ , \quad \delta\in(0,\Delta] \tag{3.25}$$

PROOF: Follows from lemma 3.3 and (3.24), since the right hand side of (3.23) is bounded above by the right hand side of (3.24). $\square$

REMARK: This form of theorem 3.2 is easy to invert. It gives us the Chernoff bound for the binomial distribution for deviations in any fixed range. Note that Δ does not have to be $O(1)$. It generalizes and improves on a result due to Angluin and Valiant [1], who proved a bound of $\exp[\frac{-rp\delta^2}{3}]$ for $\Delta = 1$. Indeed, for this case corollary 3.2.2 gives a bound of $\exp[-f(1) \cdot rp\delta^2]$ where $f(1) = 2\ln 2 - 1 \approx 0.3863$.

Using an approach very similar to that used in the proof of theorem 3.2, we have the following theorem on the deviations of S *below* its expectation rp.

THEOREM 3.4: For $\gamma \in (0,1]$,

$$\Pr[\, S - rp < -\gamma rp \,] < e^{-\frac{\gamma^2 rp}{2}} \tag{3.26}$$

COROLLARY 3.4.1: For $\gamma > 0$,

$$\Pr[\, S - rp < -\gamma \cdot rp \,] < \left[\frac{e^{\gamma}}{(1+\gamma)^{(1+\gamma)}}\right]^{rp} \tag{3.27}$$

PROOF: The proof follows from the fact that (3.27) is an upper bound on $e^{-\frac{\gamma^2 rp}{2}}$. □

REMARK: This shows that the Chernoff bound in the form shown in (3.23) holds for deviations below the mean as well. These bounds hold even when γ exceeds 1, although this case is not very interesting. Also, it is worth noting that by means of theorems 3.2 and 3.4, we have bounded the binomial distribution from above by means of a function that is symmetric about the mean. In general, the tail probability is not symmetric about the mean.

We require one more fact related to the Chernoff bound on the tail of the binomial distribution. Let $X_1, X_2, \ldots, X_r$ be independent $0-1$ random variables. Random variable X_j assumes the value 1 with probability p_j, and the value 0 with probability $1-p_j$. As before, let

$$S = \sum_{i=1}^{r} X_j \tag{3.28}$$

and let

$$m = \sum_{i=1}^{r} p_j \tag{3.29}$$

THEOREM 3.5: Let $\xi > 1$. Then

$$\Pr[\, S > \xi \cdot m \,] < \left[\frac{e^{\xi - 1}}{\xi^{\xi}} \right]^m \tag{3.30}$$

PROOF: Proceeding along the lines of (3.13), we have

$$\Pr[\, S > \xi \cdot m \,] < \frac{\prod_{j=1}^{r} \mathrm{E}[e^{tX_j}]}{e^{t \cdot \xi \cdot m}} = e^{-t \cdot \xi \cdot m} \prod_{j=1}^{r} [p_j \cdot e^t + 1 - p_j] \tag{3.31}$$

For $t = \ln \xi$, this is bounded above by

$$e^{-t \cdot \xi \cdot m} \cdot \prod_{j=1}^{r} e^{(\xi - 1) p_j} = \left[\frac{e^{\xi - 1}}{\xi^{\xi}} \right]^m \quad \square \tag{3.32}$$

An analogous result can be proved for deviations below the mean.

Theorem 3.5 has the following interesting interpretation. In bounding the tail of the sum of Bernoulli trials by means of our bounds, the only information we need is the expected number m of successes, rather than the probabilities of the individual trials. Accordingly, we introduce the following notation. We denote by $B(m,\delta)$ the Chernoff bound on the probability that the sum of Bernoulli trials with expectation m exceeds $(1+\delta)m$, for positive δ.

$$B(m,\delta) = \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} \right]^m \tag{3.33}$$

We denote by $D(m,x)$ the deviation above the mean that results in the Chernoff bound on the tail probability being x:

$$B(m, D(m,x)) = x \tag{3.34}$$

3.5 The Performance of Randomized Rounding

We now prove a guarantee on the performance of randomized rounding for the global routing problem. Let the number of edges in the lattice L be $N = 2n\cdot(n-1)$.

THEOREM 3.6: Let ε be a fixed positive constant. With probability at least $1-\varepsilon$, randomized rounding produces a solution of width

$$\le C^{(I)}\left[1 + D(C^{(I)}, \frac{\varepsilon}{N})\right] \tag{3.35}$$

PROOF: Consider constraints of the form (3.2); there are N of these. After the linear program has been solved, the values x_{ij}^* satisfy

$$\sum_{e\in t_{ij}} x_{ij}^* \le C^* \tag{3.36}$$

for any edge in E. Rounding the x_{ij} randomly results in a constraint becoming the sum of independent Bernoulli trials. The expected number of routes in the corresponding edge is no more than C^*. We now apply theorem 3.5 with deviation $D(C^*, \frac{\varepsilon}{N})$.

The probability that the number of routes in an edge (its width) exceeds

$$C^*\left[1 + D(C^*, \frac{\varepsilon}{N})\right] \tag{3.37}$$

is (by the definition of $D(m,x)$) at most ε/N. It follows that the probability that any one of the N edges has width exceeding (3.31) is no more than ε. The theorem follows from the usual observation that $C^* \le C^{(I)}$. □

The function $D(m,x)$ in theorem 3.6 suggests that the performance guarantee delivered by randomized rounding is dependent on the relative values of $C^{(I)}$ and N. Let us first consider the case $C^{(I)} \ge \ln \frac{N}{\varepsilon}$. From the point of view of gate-arrays, this is an interesting case; even if each node of L were a member of at most one net, it can be shown [18] that width $\Omega(N^{1/2})$ may be necessary in the worst case. Although this worst case may never arise in practice, stochastic analyses based on empirical studies of gate-arrays [12,33] have shown that the optimum width $C^{(I)}$ grows as a fractional power of N.

Applying corollary 3.2.2 we see that when $C^{(I)} > \ln \frac{N}{\varepsilon}$, randomized rounding yields a solution of width at most

$$C^{(I)} + (e-1)\cdot\left[C^{(I)} \ln \frac{N}{\varepsilon}\right]^{1/2} \tag{3.38}$$

with probability at least $1-\varepsilon$. A performance bound similar to this was reported in [34]; the more general version of theorem 3.6 is due to appear in [35].

On the other hand, if $C^{(I)} < \ln \frac{N}{\varepsilon}$, it is easy to apply theorem 3.2 to show that the solution produced by randomized rounding has width at most

$$\frac{e \ln N/\varepsilon}{\ln\left[\frac{\ln N/\varepsilon}{C^{(I)}}\right]} \tag{3.39}$$

Thus theorem 3.5 is particularly good when $C^{(I)}$ is large; as we have remarked earlier, this appears to be the case in practice. It is worth noting that the proof of theorem 3.5 is "loose" in at least three places - (i) we use the Chernoff bound (which is tightest when all the x^*_{ij} in a constraint are equal, an unlikely event); (ii) we add up the probabilities of too many routes in an edge to bound the probability of failure (tightest when the constraints are uncorrelated - which is never the case); (iii) the expected width of an edge after rounding may be less than C^* if a width constraint has some 'slack' in the linear program solution.

3.6 Multicommodity Flow

The algorithm in section 3.3 relied on the assumption that we were given a set $T(r_i)$ of possible configurations for each net. We noted that restricting the set of choices might be of practical value. It is nevertheless interesting to ask whether there is an approximation algorithm that explores every possible configuration for routing each net r_i. We show in this section and the next that this is possible provided each net has a small number of nodes. As before, we are not restricted to regular arrays; this algorithm can be applied to any graph.

In this section we consider the special case where each net has exactly two nodes which we shall call *terminals*. A configuration that routes such a net is then a simple path in L joining the terminals of the net. We can then restate the global routing problem as follows. We are given a regular array L and a set R of pairs of terminals. Each pair of terminals is to be joined by a path in L. We are to find the smallest integer $C^{(I)}$ so that no more than $C^{(I)}$ paths pass through any edge.

This is a version of the *integral multicommodity flow* problem [17]. In this problem, we are given a graph together with k pairs of nodes; one node in each pair is marked s_j and one marked t_j. Each node s_j is a source of a commodity labeled j, and we are to convey d_j units of commodity j from s_j to t_j. The d_j, which are integral, are known as *demands*. For each edge e of the graph we are given a capacity $c(e)$. The flows are to be realized as integers, and the total flow in an edge (measured as the sum of the flows of all commodities in that edge) must not exceed its capacity.

Clearly, if we could solve the integral multicommodity flow problem, we could solve the global routing problem for two-node nets. The general integral problem is known to be NP-Complete [10], although the non-integral version can be solved using linear programming methods [17] in polynomial time. As in the algorithm of section 3.3, the difficulty with the linear program solution is that we have flows that take on fractional values in the interval [0,1]. As before we denote the linear program optimum by C^*, and a fractional flow $f_i(e)$ for commodity (net) i in edge e. We are therefore confronted again with the problem of rounding fractional flows to integral flows. In this case there is no direct interpretation of the fractional flows as probabilities. We now present two techniques for regarding the fractional flows as a probability space.

3.6.1 Path Stripping and Coin Flipping

The main idea of this phase is to convert the edge flows for each net r_i into a set Γ_i of possible paths which could be used to route the net. Initially, Γ_i is empty.

For each i:

(1) Form a directed subgraph $L_i(V,E_i) \subseteq L(V,E)$ where E_i is a set of directed edges derived from E as follows: for each $e \in E$, assign a direction to e which is the direction of positive flow in e. If $f_i(e) = 0$, e is excluded from E_i.

(2) Discover a directed path $\{e_1, \ldots, e_m\}$ in L_i from s_i to t_i using a depth-first search, discarding loops. Let

$$f_m = \min \{f_i(e_j),\ 1 \le j \le m\} \tag{3.40}$$

For $1 \le i \le m$, replace $f_i(e_j)$ by $f_i(e_j) - f_m$.

Add the path $\{e_1, \ldots, e_m\}$ to Γ_i along with its *weight* f_m.

(3) Remove any edges with zero flow from E_i. While there is non-zero flow leaving s_i, repeat step (2).

Otherwise, next i.

It is clear that the above process terminates, since at each execution of step (2), at least one edge (the one with minimum flow in the path) is deleted from E_i. Thus the number of times it is executed is bounded above by $|E|$. It is also evident that on termination, the sum of the weights of the paths in Γ_i is one.

Once we complete the path-stripping process for each net i, we can regard the sets Γ_i of paths as the sample spaces for the randomization. The randomization step is similar to that in section 3.3, except that for net r_i we now have $|\Gamma_i|$ choices rather than $|T(r_i)|$.

For each i independently **do**:

Cast a $|\Gamma_i|$-faced die with face-probabilities equal to the weights of the paths in Γ_i. Assign to net r_i the path whose face comes up.

Let $C^{(I)}$ be the width of the best integer solution. We can then prove a theorem similar to theorem 3.6; the proof is essentially the same.

THEOREM 3.7: For any $\varepsilon > 0$, with probability at least $1-\varepsilon$ the width of the solution produced by path stripping and randomized rounding does not exceed

$$C^{(I)}\left[1 + D(C^{(I)}, \frac{\varepsilon}{N})\right] \tag{3.41}$$

3.6.2 A Solution Using Random Walks

The idea of path stripping was to convert the fractional flows generated by the linear program into a sample space for the randomized rounding phase. We now present an alternative method for rounding that uses random walks through the regular array, instead of path stripping. Each net then chooses a route independently of all other nets as follows. For each net r_i we construct once again the subgraph L_i, as before. We route the net from s_i to t_i by means of a random walk based on the flow values in L_i. At a typical step, the net proceeds from a node to its successor, chosen in the following manner. Let the edges directed out of the node be $e_1, \ldots, e_d$, and the associated fractional flows be $f_i(e_1), \ldots, f_i(e_d)$. We choose to continue the walk on e_j with probability

$$\frac{f_i(e_j)}{f_i(e_1) + \ldots + f_i(e_d)}, \quad 1 \le j \le d \tag{3.42}$$

Using the following lemma, theorem 3.7 can be shown to hold in this case as well.

LEMMA 3.8: Let $P_i(e)$ be the probability that the routing of net r_i passes through edge e.

$$P_i(e) = f_i(e) \tag{3.43}$$

3.7 Multiterminal Multicommodity Flows

In this section we show that some of the ideas involving multicommodity flow extend to the case where each net has a small number of nodes. We describe only the case when each net r_i has at most three nodes. We show that in this case we can in fact explore every possible configuration for routing r_i, just as we did for two-node nets in the previous section. We describe the integer program and the randomization below.

Consider a net r_i consisting of three nodes v_{i1}, v_{i2} and v_{i3}. In general, any configuration for routing r_i must consist of a Steiner point in L to which we connect each of the nodes v_{i1}, v_{i2} and v_{i3} (in a 'degenerate' case, the Steiner point could be one of v_{i1}, v_{i2} or v_{i3}). For each node v_k of L, we assign a 0−1 indicator variable y_{ik} to indicate whether or not v_k is the Steiner point used to connect net r_i. We write a constraint to ensure that r_i has a Steiner point:

$$\sum_k y_{ik} = 1 \ , \quad \forall \ i \tag{3.44}$$

We now set up a multicommodity flow network in which each node v_{i1}, v_{i2} and v_{i3} must send y_{ik} units of flow to node v_k. As before, we can try to minimize the common capacity $C^{(I)}$ of all the edges of L. It is clear that an integer solution for the y_{ik} and the flows corresponds to a routing.

The solution to the relaxation linear program will now contain both fractional flows and fractional Steiner points (i.e. the y_{ik} will assume fractional values y_{ik}^*). The rounding process now consists of two phases, each of which is conducted independently for all the nets. In the first phase, the Steiner point for net r_i is chosen as follows: node v_k is chosen with probability y_{ik}^*. In the second phase, the fractional flows from v_{i1}, v_{i2} and v_{i3} to the chosen Steiner point are rounded using one of the techniques of the previous section. This results in v_{i1}, v_{i2} and v_{i3} being connected to the Steiner point; net r_i is thus connected.

It is easy to show that theorem 3.7 holds in this case as well.

3.8 Experimental Results for Global Routing

In this section we present the results of some preliminary experiments with global routing in gate arrays using randomized rounding. The experiments described here involve a very special implementation of the algorithm in section 3.3. We present results on two gate arrays drawn from industrial sources. Both are relatively small gate arrays compared to the current state of the art.

These results are described in greater detail in [27]. In interpreting these results, it is important to realize that only two small arrays have been studied. The practical feasibility of the method will depend on more experiments involving larger arrays.

3.8.1 The Decomposition Style of Routing

In routing practice, a popular technique for dealing with a multi-node net is to decompose it first into simpler *connections* and route each of the smaller connections individually. For instance, a t-node net can be decomposed into $t-1$ two-node connections. Our initial round of experiments used such a decomposition process. For each decomposed connection, we consider a set of routes that can be used to route it - this corresponds to the set $T(r_i)$ of section 3.2. One of the routes that we consider will be used to route the connection. We describe below the details of implementation - decomposition, linear program generation and coding.

3.8.1.1 Decomposition of Nets

Each net is decomposed into connections of different types using the following heuristics due to Hanan [14].

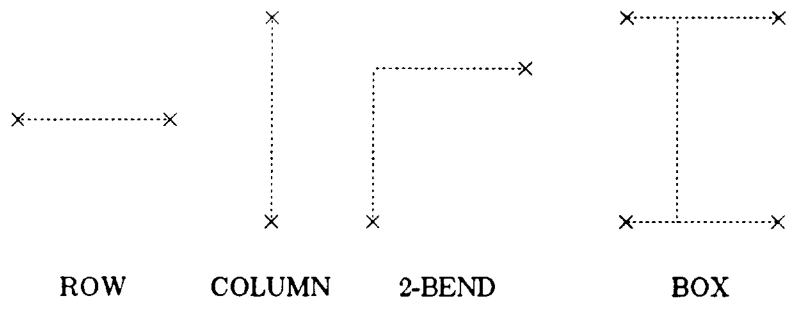


Figure 3.1: Types of connections.

There are four types of connections, depending on the relative position of the nodes to be connected. If the nodes lie on the same row of the lattice, they are connected with a *row* connection. Similarly, if they lie on the same column, they are connected with a *column* connection. If the nodes do not share a common row or column, they are connected by a *two-bend* connection - the nodes are connected using any minimum-length route with at most two bends. In the special event that there are four nodes on the corners of a rectangle in the lattice, a *box* connection is used. Figure 3.1 shows the types of connections used.

3.8.1.2 Decomposition Heuristics

The decomposition phase reduces all nets to connections of the types listed above. Nets with two nodes are trivially decomposed, since a two-node net is either a row, a column, or a two-bend connection.

Nets with three nodes are treated under two cases (figure 3.2). The *median-point* is defined as the point whose coordinates are the median values of the three row and three column coordinates. If the net has a node at the median-point, then the median-point is routed to the other two nodes by two two-node connections. Otherwise, a Steiner point is introduced at the median-point and the three nodes are connected to the Steiner point by two-node connections.

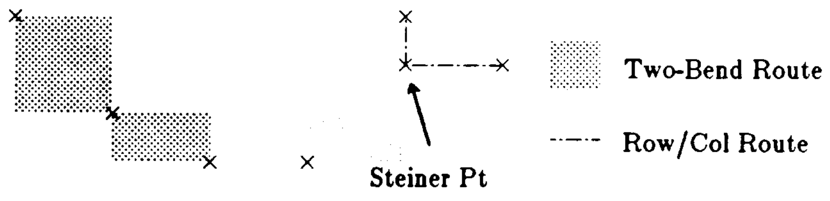


Figure 3.2: Decompositions of three-node nets.

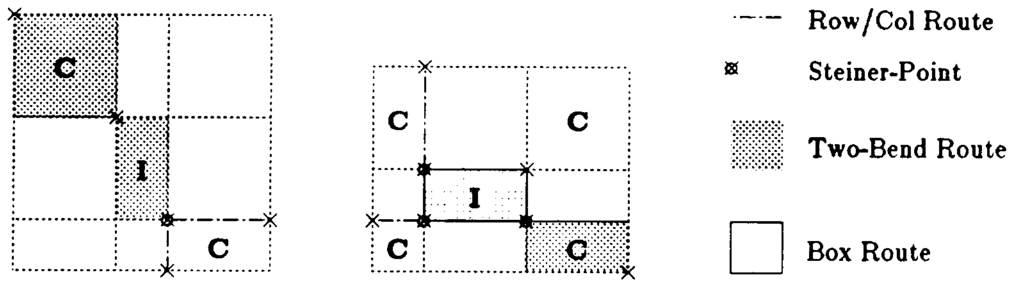


Figure 3.3: Decompositions of four-node nets.

Four-node nets are decomposed by considering the four corner rectangular areas (labeled *C*). Each corner rectangular area (see figure 3.3) is decomposed by connecting all nodes in the area to the corner adjacent to the inner rectangular area. The inner rectangular area (labeled *I*) is then decomposed as either a two-bend or a box connection.

Nets with five or more nodes are decomposed using a minimum spanning tree algorithm [22]. The edges in the (rectilinear) minimum spanning tree define connections in a natural manner. Each connection consists of two nodes and is trivially a row, column, or two-bend connection.

3.8.1.3 Linear Program Generation

The linear program is generated from the list of connections produced by decomposing the nets. Each connection can be realized using one of a set of configurations. A parameter 'SPAN' controls the number of configurations that are generated for row and column connections. For row and column connections, the route can be displaced on either side from the straight route by up to SPAN gates. This results in (2 * SPAN + 1) configurations for row and column connections.

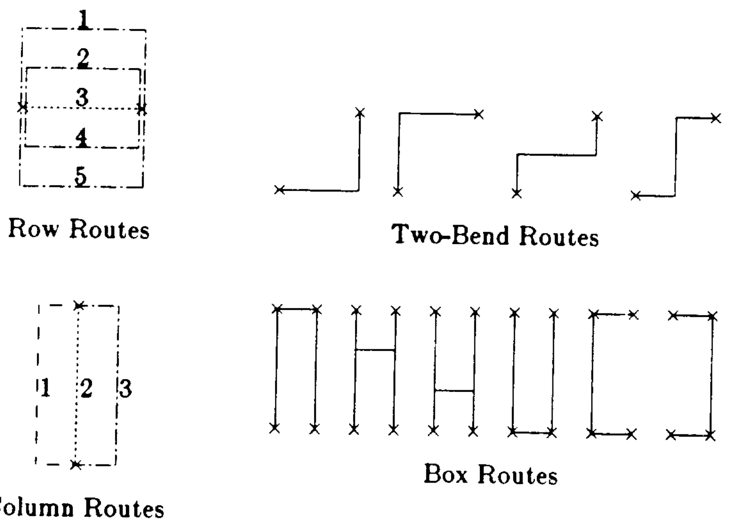


Figure 3.4:Configurations for various connections.

Two-bend connections are realized by considering all the possible minimum distance two-bend routings. Box connections are realized using the 'H'-shaped configurations shown in figure 3.4 (a 'U'-shaped connection is a degenerate case).

3.8.1.4 Coding

The collection of programs for decomposition, linear program generation, and rounding were written in the C programming language and run on a VAX 11/785 running Berkeley 4.3 BSD UNIX. Processing time on the VAX is short enough to allow interactive usage of the various modules. In particular once the linear program is solved, one instance of the rounding problem can be solved in about 1 VAX CPU second. The linear program is run using MPSX, an implementation of the simplex algorithm for linear programming, on an IBM 3081 running IBM/CMS.

3.8.1.5 Problems arising from Net Decomposition

Two problems arise from the decomposition style of routing. Both problems occur because it is not possible to specify within the context of the linear program which net the connections belong to. In both cases the effect on the objective function is to raise it artificially.

When two connections pass through the same channel, they occupy two tracks. If they belong to the same net, the electrical route will only require one track. We call this phenomenon *track sharing*.

A *cycle* occurs when, in a physical realization of a net, there is more than one distinct path from a node on the net to some other node in the lattice. This can happen when non minimum-distance routes overlap. Again this means counting unnecessary tracks and may artificially raise the objective function value.

3.8.2 The Experiments

We now describe the performance of randomized rounding on two gate arrays from industrial sources; we call them example A and example B. Since these were actual arrays from industrial practice, they had specifications of channel-capacity which had to be respected in order to generate a feasible routing. In our linear programs, we defined a new quantity, the *excess*, which is the number of routes through an edge (channel) less its capacity. Let E denote the maximum excess (taken over the edges in the array). This quantity E was the linear program objective in these experiments. Let E^* denote the optimum value of the linear program objective function.

Table 3.1 gives basic data on the two examples, such as array size, number of nets and statistics on the net sizes.

Array	Size (Nodes)	No. of Nets	No. of 2 nd. Nets	No. of 3 nd. Nets	No. of 4 nd. Nets	No. of >4 nd. Nets	Nds. in Largest Net	Avg. no. of nds. in Net
A	15 x 12	285	111	152	19	3	5	2.6
B	17 x 23	449	257	88	29	75	21	3.64

Table 3.1: Input data for the two examples.

Array	No. of Connec-tions	No. of Row Connections	No. of Column Connections	No. of 2-bend Connections	No. of Box Connections
A	506	154	238	109	5
B	1266	481	440	341	4

Table 3.2: Decomposition Statistics.

Example B has over twice as many gates as example A. While B does not have twice as many nets as A, it contains several very large nets (including a 21-node net) and thus a larger average net size. Larger nets lead to more 2-node connections, non-minimal distance routings and the problems of track-sharing and cycles mentioned in the previous section.

Table 3.2 contains information about the examples after their nets have been decomposed into 2-node connections as described in the previous section. Notice that although B did not have twice as many nets as A, it has over twice as many connections - this is, as we noted above, due to the more complex nets in B. For each decomposed example, four linear programs were created for values of SPAN from one through four (corresponding to increasing degrees of freedom for the routes).

Table 3.3 gives information on the linear programs - number of constraints (rows), variables (columns), the number of simplex iterations to feasibility and optimality, the IBM 3081 runtime in seconds and the optimal value of the objective function E^*. The last column shows a surprising phenomenon; it was found that many of the variables in the linear program solution had already been assigned integer values (0 or 1), and thus did not have to be rounded.

LP	SPAN	No. of LP rows	No. of LP columns	Iter. to feas.	Iter. to optim.	Runtime (sec.)	Optimal value of E^*	%age of integer solutions
A1	1	840	1774	497	514	25.10	0.0	77 %
A2	2	840	2435	402	474	32.93	-0.1667	75 %
A3	3	840	3050	0	145	15.63	-0.1667	92 %
A4	4	840	3600	0	146	19.74	-0.1667	92 %
B1	1	2009	5570	126	203	38.23	0.00	100 %
B2	2	2009	7270	133	187	42.98	0.00	100 %

Table 3.3: Linear program statistics for each example.

Integral-valued variables correspond to routed connections, and the last column in table 3.3 indicates the percentage of such connections in each of the cases. The most interesting result here is that in example B, for the cases B1 and B2, all 1266 connections were routed deterministically (100 % integer solutions). Furthermore, this was with an objective function $E^* = 0$; in other words, the linear program had found a perfect routing! For example B, increasing SPAN to 3 produced no reduction in the objective function, although some of the variables now took on fractional solutions.

Unlike example B the linear programs for example A did not directly yield a routing (all the solutions integral), so that randomized rounding was necessary. However, E^* did attain negative values for A2, A3 and A4. Interestingly, the versions with SPAN=3 and 4 (more freedom allowed) reached and optimum with fewer simplex iterations and thus decreased runtime. Thus, increased flexibility in the routes means an increase in the size of the linear program but not necessarily an increase in runtime.

Where necessary, the fractional variables from the linear program were converted to integer (0-1) solutions using randomized rounding, to produce physically meaningful routings. For each of the cases A1, A2, A3 and A4, fifty-one independent randomized roundings were performed (no rounding was necessary for B1 and B2 since the linear program solution in these cases was perfectly integral). Table 3.4 summarizes the results of rounding.

Example	No. of Violations		Frequency of
	Average	Best	best of solution
A1	27.61	19	2 %
A2	22.92	14	4 %
A3	2.73	0	4 %
A4	3.53	0	8 %
B1	0	0	100 %
B2	0	0	100 %

Table 3.4: Results of 51 roundings of linear program solutions.

In each case, we list the number of channel capacities violated by the rounded solution (the "routing"); a value of zero corresponds to a feasible routing to the problem. Note that feasible solutions were found in the cases A3, A4, B1 and B2 (the column "Best" under "No. of violations" gives the minimum number of capacity violations among the 51 roundings). Notice also that the trend for the cases A1 through A4 under 'average number of violated channels' shows that as the freedom (SPAN) is increased, we proceed closer to a feasible routing. This is also confirmed by the frequency with which the best solution occurs - A3 produced a feasible routing only 2 times out of 51 roundings, while A4 yielded a routing 4 times out of 51. Our results suggest that SPAN = 3 is necessary to successfully route example A (SPAN = 2 could not even come close to a routing).

3.8.3 Further Experimental Work

In this section we summarize our ongoing experimental research. Two major goals can be identified in this respect. One is to eliminate the decomposition style of routing, for the following reasons. The quality of the routing produced by the decomposition process is sensitive to the particular decomposition heuristic used. Also, in piecing together the routing of a net from the routings of its constituent connections, we encounter the problems of track-sharing and cycles discussed in section 3.8.1.5. To this end, we are working on directly generating a set $T(r_i)$ of possible configurations for each net. In doing so, we would like to maintain some control over the set of trees generated. For this purpose, we are developing a language for describing tree configurations connecting a set of nodes in a lattice [28].

Our other effort is in experimenting with more gate arrays drawn from industrial practice. As mentioned before, the two examples we have studied so far are small compared with the current state of the art. Larger arrays mean larger linear programs, and thus greater running time. Another factor which governs the size of the linear program is the number of configurations considered for routing each net. While a richer set of configurations should lead to better routings, it would also increase the linear program size. The tradeoffs between running time and routing quality (or even routability) should be interesting to study.

Chapter 4
Packing Integer Programs

4.1 Overview

In chapter 3 we saw that the integer programming problem of global routing could be approximated by solving the relaxation linear program and then using randomized rounding. It is natural to ask what other integer programs can be tackled using a similar approach. In this chapter we show that the global routing problem we have considered is related to a class of integer programs known as "packing" problems. Direct application of randomized rounding does not yield a feasible approximate solution to such integer programs. We introduce a device we call *scaling* to extend randomized rounding to these packing integer programs.

4.2 Packing Problems

Let A be an $n \times r$ matrix in which each element a_{ij} is either a 0 or a 1 (in section 4.6 we consider more general matrix entries). Consider the following integer linear program:

$$\begin{aligned} &\text{Max. } \sum_{j=1}^{r} a_{1j}\, x_j \\ &\text{s.t. } \sum_{j=1}^{r} a_{ij}\, x_j \le k \ , \quad 2 \le i \le n \\ &x_j \in \{0,1\} \end{aligned} \tag{4.1}$$

This is a packing integer program in the following sense: we are trying to pack as many of the column-vectors of A as possible into an n-dimensional cube of side k. The constraints stipulate that the vector sum of the chosen vectors should fit in the cube. The reader may wonder at this stage why any of the entries a_{1j} should be 0, since the corresponding column of A would then be redundant. The answer is that in section 4.5 we will study a generalization of the packing program (4.1) in which the coefficients a_{ij} can be reals in the interval [0,1]. In the interest of uniformity of notation, we present the integer program in the form (4.1).

Lovász [25] calls this problem *simple k-matching in a hypergraph*. The reason for this terminology is that A can be thought of as the incidence matrix of a hypergraph, with the rows representing the vertices and the columns the edges. The element a_{ij} is a 1 if edge j is incident on vertex i. The integer program of (4.1) then seeks the largest set of edges such that no more than k are incident on any vertex. The problem can also be phrased in terms of set systems.

Aside from these combinatorial applications, there is also a scheduling interpretation to the integer program (4.1). Each of the variables x_j may be associated with a task. Each row of the matrix represents a machine or facility. The entry a_{ij} is 1 if the execution of task j requires a unit of time on machine i. We wish to maximize the number of tasks that can be scheduled for execution within a finishing time k. The assumption here is that the processing of a task by various machines does not have to follow any sequence, and the tasks are unrelated.

In discussing the problem below, we will speak of it as k-matching; the terminology is, however, purely a matter of convenience.

As in chapter 3, our approach consists of solving the relaxation linear program with $x_j \in [0,1]$. Let the linear program yield a value x_j^* for the variable x_j. Let the corresponding value of the objective function be M^*; various authors [11,25] have referred to M^* as the *fractional k-matching number* of the hypergraph.

When $r > n$, elementary linear algebra can be used [2] to show that we can round $r-n$ of the fractional variables to integer values without changing the values of $\sum_{j=1}^{r} a_{ij} x_j$ for $1 \le i \le n$. The following remarks therefore apply when we are faced with rounding $r \le n$ variables x_j.

For all j, independently we set x_j to 1 with probability x_j^*. Let the resultant rounded value of variable x_j be $x_j^{(r)}$. The difficulty now is that after rounding, $\sum_{j=1}^{r} a_{ij}\, x_j^{(r)}$ may exceed k for some i, thus violating a constraint. Indeed, it can happen that the probability that *some* constraint is violated is very high. We thus have to diminish the chance that more than k variables that contribute to a constraint will get rounded. To this end, we introduce a device which we call *scaling*.

4.3 Scaling

We begin by illustrating the simplest version of this technique. Let ε_1 and ε_2 be fixed reals in the interval (0,1) such that $\varepsilon_1 + \varepsilon_2 = \varepsilon < 1$. Let $\nu \in (0,1)$ be a number such that

$$B(\nu k\ ,\ \frac{1-\nu}{\nu}) < \frac{\varepsilon_1}{n} \tag{4.2}$$

We have not as yet established under what conditions such a ν must exist; let us for the moment continue under the assumption that we do have such a value of ν. The idea is to multiply each x_j^* by ν before rounding; this can be thought of as "scaling" down the probability that $x_j^{(r)}$ is 1.

$$x_j^s = \nu \cdot x_j^* \tag{4.3}$$

The superscript s indicates a fractional value that has been scaled. As a result, the fractional value of the objective function is also scaled down by the factor ν; we let M^s denote $\nu\, M^*$. The randomized rounding process now consists of rounding variable x_j to 1 with probability x_j^s. We will now show that after rounding, with high probability no constraint is violated *and* the rounded value of the objective function does not fall "too far" below M^s.

THEOREM 4.1: With probability at least $1-\varepsilon$, scaling followed by randomized rounding finds an integer k-matching of cardinality at least

$$M^s \cdot [\ 1\ -\ D(M^s, \varepsilon_2)\ ] \tag{4.4}$$

PROOF: After rounding, the expected value of each constraint is no more than νk. By our choice of ν (equation 4.2), the probability that a constraint is violated (i.e. its value exceeds k) is thus less than $\frac{\varepsilon_1}{n}$. Thus the probability that *any* constraint is violated is less than ε_1.

The expected value of the objective function is M^s. The probability that it falls below (4.4), by corollary 3.4.1, is less than ε_2. Thus, with probability at least $1-(\varepsilon_1+\varepsilon_2)$, we will have a k-matching of the cardinality guaranteed by the theorem. □

How does the guarantee of theorem 4.1 compare with the optimum k-matching? We know that the relaxation linear program optimum M^* is an upper bound on the integer optimum. The value M^s in theorem 4.1 is smaller than M^* by the multiplicative factor ν. Theorem 4.1 assures us of finding a k-matching that is smaller than νM^* by a subtractive factor.

It remains to determine for what values of k there exists a positive value of ν satisfying (4.2). The performance guarantee of theorem 4.1 is stated somewhat abstractly in terms of the function D, because we wished to give a bound that was correct for all values of k. Examination of (3.23) reveals that if k exceeds $\ln n/\varepsilon$, ν is a positive constant. For the remainder of this chapter, we will be interested only in values of $k > \ln n/\varepsilon$. We do so in order to avoid having to deal with case studied in (3.39). With this assumption on k, we can now state more concrete bounds. To do so, we require the following fact.

PROPOSITION 4.2: For a packing problem of the form (4.1),

$$M^* \geq k \tag{4.5}$$

PROOF: Any k columns of A (corresponding to any k variables x_j) constitute a feasible solution to the packing program (4.1). Thus the integer optimum $M^{(I)}$ is bounded below by k, and above by M^*. □

It follows from our assumption about k that M^* is at least $\ln n/\varepsilon$. Using corollary 3.2.2 with $\Delta = e-1$, we find that $\nu \geq 1/e$. The following special case of theorem 4.1 results.

COROLLARY 4.1.1: Let $M^{(I)}$ be the size of the optimum k-matching. When $k \geq \ln n/\varepsilon$, with probability at least $1-\varepsilon$ scaling followed by randomized rounding finds a matching of cardinality

$$\geq \frac{M^{(I)}}{e} - \left[\frac{2M^{(I)}\ln n/\varepsilon}{e}\right]^{1/2} \tag{4.6}$$

Proof: We let ε_2 of theorem 4.1 be ε/n, and ε_1 be $\varepsilon-\varepsilon/n$. Application of corollary 3.2.2 and theorem 3.4 yields the result. □

We thus have what Papadimitriou and Steiglitz [31] call a *fully polynomial-time approximation scheme (FPTAS)* for the packing problem (4.1). As we observed in connection with the global routing problem, the performance guarantee is best when the integer optimum is large.

4.4 Maximum Multicommodity Flow

In section 3.6 we used a version of the integer multicommodity flow problem to solve a special case of the global routing problem. The following version of the problem, known as *maximum 0-1 multicommodity flow*, is an important problem in operations research [23]. We are given a directed graph $G(V,E)$, and k source-sink pairs as described in section 3.6. Each edge $e \in E$ has a positive capacity $c(e)$. For $1 \leq j \leq k$, the flow of commodity j is said to be realized if we convey one unit of flow from source s_j to the corresponding sink t_j. The flow must be *integral*, i.e. we must specify a path in G from s_j to t_j. We wish to maximize the number of commodities whose flow is realized, with the constraint that the total flow in any edge e does not exceed $c(e)$. (In some problems of practical interest, constraints are also placed on the flux through each node in$V$; our methods could be adapted to this case as well).

This problem can be formulated as a $0-1$ integer linear program. We know that optimizing this integer linear program is NP-Hard [10,17]; but the relaxation linear program can be solved efficiently. We will show that randomized rounding will find an approximate solution to the maximum multicommodity flow problem provided no edge capacity is very small. The idea is to use the techniques of path-stripping and randomization developed in section 3.6, together with appropriate scaling as in section 4.3.

Let F^* be the optimal value of the (maximum) fractional flow. We denote by $F^{(I)}$ be the best integer optimum. Let $N = |E|$ be the number of edges in the network. We require the following fact analogous to proposition 4.2.

PROPOSITION 4.3: Let c be the smallest edge capacity in a problem instance.

$$F^* \geq F^{(I)} \geq c \tag{4.7}$$

The algorithm we will present is applicable to networks in which the minimum edge-capacity c is at least $\ln N/\varepsilon$, where ε is a fixed constant in the interval (0,1). We now define ν in a fashion analogous to equation (4.2).

$$B(\nu c\ ,\ \frac{1-\nu}{\nu}) < \frac{\varepsilon}{N+1} \tag{4.8}$$

Let $f_i(e)$ be the variable denoting the flow of commodity i in edge e. The algorithm consists of the following four phases.

(1) Solve the relaxation linear program maximizing the total fractional flow. Let the fractional flow of commodity i in edge e be $f_i^*(e)$. Let the corresponding maximum fractional flow be F^*.

(2) Scale all flows down by the factor ν to obtain scaled flows f_i^s:

$$f_i^s(e) = f_i^*(e) \tag{4.9}$$

The corresponding scaled fractional flow is $F^s = \nu F^*$.

(3) For each commodity i, perform path-stripping and generate a set of paths that may be used to realize the flow of commodity i.

(4) Choose a path for realizing the flow of commodity i at random, as described in section 3.6.

In a manner similar to the proof of theorem 4.1, we can now prove the following performance guarantee theorem.

THEOREM 4.4: With probability at least $1-\varepsilon$, the procedure described above will find a maximum multicommodity flow of value at least

$$\geq \frac{F^{(I)}}{e} - \left[\frac{2F^{(I)}\ln N/\varepsilon}{e}\right]^{1/2} \tag{4.10}$$

We thus have a fully polynomial-time approximation scheme for instances of maximum multicommodity flow in which no edge capacity is smaller than ln N/ε for fixed positive ε.

4.5 The Weighted Sum of Bernoulli Trials

In preparation for a generalization of the packing program (4.1) in section 4.6, we study the following problem. Let $a_1, a_2, \ldots, a_r$ be reals in the interval (0,1]. Let $X_1, X_2, \ldots, X_r$ be independent Bernoulli trials with p_j being the probability that X_j assumes the value 1. We wish to study the following random variable:

$$S = \sum_{j=1}^{r} a_j X_j \tag{4.11}$$

Its expectation is given by

$$E[S] = \sum_{j=1}^{r} a_j p_j = m \tag{4.12}$$

We now prove a Chernoff-type bound on the probability that S deviates far above its expectation. In fact, we show that the appropriate version of theorem 3.5 holds in this case.

THEOREM 4.5: Let $\xi > 1$. Then

$$\Pr[\, S > \xi m \,] < \left[\frac{e^{\xi - 1}}{\xi^{\xi}}\right]^m \tag{4.13}$$

PROOF: The proof is very similar to the proof of theorems 3.2 and 3.5.

$$Pr[\, S > \xi m \,] \leq \frac{E[e^{tS}]}{e^{t\xi m}} \tag{4.14}$$

for any positive real t. This can be written as

$$\frac{\prod_{j=1}^{r} [\, p_j e^{ta_j} + 1 - p_j \,]}{e^{t\xi m}} \leq e^{-t\xi m} \prod_{j=1}^{r} \exp[\, p_j (e^{ta_j} - 1) \,] \tag{4.15}$$

For $t = \ln \xi$, this becomes

$$\xi^{-\xi m} \exp\left[\sum_{j=1}^{r} p_j \left[(\xi)^{a_j} - 1 \right] \right] \tag{4.16}$$

which is

$$\leq \xi^{-\xi m} \exp\left[\sum_{j=1}^{r} (\xi - 1) a_j p_j \right] = \left[\frac{e^{\xi - 1}}{\xi^{\xi}} \right]^m \quad \square \tag{4.17}$$

In a similar fashion, we can prove a theorem corresponding to theorem 3.4:

THEOREM 4.6: For $\gamma \in (0,1]$,

$$Pr[\, S - m < -\gamma m \,] < e^{-\frac{\gamma^2 m}{2}} < \left[\frac{e^{\gamma}}{(1+\gamma)^{(1+\gamma)}} \right]^m \tag{4.18}$$

4.6 The General Packing Problem

Let A be an $n \times r$ matrix in which each element a_{ij} is in the interval [0,1]. Consider the following packing program:

$$\begin{aligned} &\text{Max.} \sum_{j=1}^{r} a_{1j}\, x_j \\ &\text{s.t.} \sum_{j=1}^{r} a_{ij}\, x_j \leq k \,, \quad 2 \leq i \leq n \\ &x_j \in \{0,1\} \end{aligned} \tag{4.19}$$

This is a generalization of the packing program (4.1). In view of theorems 4.5 and 4.6, we now have a fully polynomial-time approximation scheme for the general packing program (4.20).

THEOREM 4.7: Let $M^{(I)}$ be the value of the integer optimum for the general packing program (4.20). When $k \geq \ln n/\varepsilon$, with probability at least $1 - \varepsilon$ scaling followed by randomized rounding finds an integer optimum of value

$$\geq \frac{M^{(I)}}{e} - \left[\frac{2M^{(I)} \ln n/\varepsilon}{e} \right]^{1/2} \tag{4.20}$$

Chapter 5

Discrete Ham-Sandwich Theorems and Integer Approximation

5.1 Overview

All the problems we have studied so far have been optimization problems. In this section we introduce some combinatorial problems that are related to our optimization problems in that they yield to similar solution techniques. These are the *set-balancing* problems studied by Olson and Spencer [29,30,36]. In section 5.2 we survey the results of Olson, Spencer and others. In section 5.3 we state the *integer-approximation problem*, a variation of set-balancing studied by Beck and Fiala [2]. This problem will be the basis for a technique we will develop in the next chapter for replacing randomized rounding by a deterministic procedure.

5.2 Balancing Families of Sets

In an instance of the *set balancing* problem, we are given a family θ of n finite sets $\theta = \{S_1, S_2, \ldots, S_n\}$. Let

$$S = \bigcup_i S_i \tag{5.1}$$

Let $r = |S|$. We wish to partition S into two parts θ_A and θ_B. For a given partition of S, the *discrepancy* of set S_i is defined as

$$\Delta_i = \left| \, | S_i \cap \theta_A | - | S_i \cap \theta_B | \, \right| \tag{5.2}$$

We wish to construct a partition so as to minimize the maximum discrepancy over all i

$$\Delta(\theta) = \max_i \{ \Delta_i \} \tag{5.3}$$

Using techniques from linear algebra it can be shown [2,18,29] that if $r > n$, we can assign $r-n$ elements of S to A or B without increasing the value of $\Delta(\theta)$. This assignment can moreover be done in deterministic polynomial time. Therefore, we only consider the case $r \leq n$ below.

The simplest algorithm for partitioning the elements of S is to assign each element of S independently to A or B by flipping a random coin to make the choice. It follows that the expected discrepancy

$$E[\Delta_i] = 0, \quad \forall i \tag{5.4}$$

We now wish to answer the following question: how far can Δ_i deviate from its expected value? Unlike the problems studied in chapters 3 and 4, we are now interested simultaneously in deviations of Δ_i both above and below its expectation. Using the Chernoff bound, Olson and Spencer [29] proved the following theorem.

THEOREM 5.1: There exists a partition of the elements of S such that

$$\Delta(\theta) \leq (n \ln 2n)^{1/2} \tag{5.5}$$

In the same paper, they proceed to show that such a partition can be constructed in *deterministic* polynomial time. In the next chapter we will show that their result for deterministic construction can be improved upon and generalized.

Subsequently Spencer [37] showed the following.

THEOREM 5.2: For any family of sets θ, there always *exists* a partition of S such that

$$\Delta(\theta) \leq 6\sqrt{n} \tag{5.6}$$

The proof uses an ingenious pigeonholing argument which unfortunately does not lead to a polynomial-time algorithm for constructing such a partition. This result is also the "best possible" in that there exists a family of sets θ for which $\Delta(\theta)$ is $\Omega(\sqrt{n})$. Spencer also showed that there existed a partition of S such that Δ_i is $O(\sqrt{i})$.

The family θ of subsets of S can be represented by means of an incidence matrix A in which rows represent sets S_i and columns represent the elements of S. The matrix entry a_{ij} is 1 if set S_i contains the j^{th} element of S. The matrix A is thus an $n \times r$ $0-1$ matrix.

It is natural to try and extend the balancing problem to the case when the matrix entries a_{ij} assume other values than 0 and 1. Instead of partitioning the elements of S, we can now speak of partitioning the columns of A. We call this the *matrix balancing* problem. Let $(x_1,\dots,x_r)$ be a vector such that $x_j \in \{-1,+1\}$. By associating each component of the vector with a column of A, we can consider the partitioning process as one of assigning signs to the columns. The maximum discrepancy with respect to a given vector x (partition) can then be defined as

$$\Delta_A (x) \quad = \quad \max_i \left| \sum_{j=1}^{r} a_{ij} x_j \right| \tag{5.7}$$

Spencer [37] shows that the probabilistic and deterministic constructions mentioned above hold (within constant factors) provided $|a_{ij}| \leq 1$ for all i,j. It is interesting to note that the problem of minimizing discrepancy can be cast as an integer program. We write constraints of the form

$$\sum_{j=1}^{r} a_{ij} x_j \leq \Delta \tag{5.8}$$

and

$$\sum_{j=1}^{r} a_{ij} x_j \geq -\Delta \tag{5.9}$$

Subject to these constraints, we minimize Δ for $x_j \in \{-1,+1\}$.

If now we were to allow a relaxation $x_j \in [-1,+1]$, we find that the optimal values for the x_j are

$$x_j \quad = \quad 0 \ , \quad 1 \leq j \leq r \tag{5.10}$$

This holds regardless of the matrix A. The application of randomized rounding at this point is to simply assign ± 1 to x_j with equal probabilities. Thus the proof of theorem 5.1 can be viewed as a special case of randomized rounding.

5.3 The Integer-Approximation Problem

In this section we consider the following *integer-approximation* problem studied by Beck and Fiala [2]. Let A be a $n \times r$ matrix (we continue to assume that $r \leq n$). Given a vector $p = (p_1 , p_2 , \dots , p_r)$ of reals, we are to construct a vector $q =$

$(q_1 , q_2 , \ldots , q_r)$ of integers such that

$$\left| \sum_{j=1}^{r} a_{ij} \cdot (p_j - q_j) \right| \tag{5.11}$$

is "small" for all i.

We may assume without loss of generality that $p_j \in [0,1]$ for all j. Let us now consider the restricted class of solutions $q_j \in \{0,1\}$ for all j; the q_j are thus "rounded" versions of the p_j. With these restrictions Beck and Fiala prove the following theorem.

THEOREM 5.3: Given the matrix A and the vector p in an instance of the integer approximation problem, a vector q can be constructed in deterministic polynomial time such that

$$\left| \sum_{j=1}^{r} a_{ij} \cdot (p_j - q_j) \right| \le (8n \ln 2n)^{1/2} , \quad 1 \le i \le n \tag{5.12}$$

We will improve on this result in the next chapter. Spencer [37] has shown that for every input A and p, there exists an integer-approximation vector q such that

$$| \sum_{j=1}^{r} a_{ij} \cdot (p_j - q_j) | \le 6 \sqrt{n} , \quad 1 \le i \le n \tag{5.13}$$

As in the case of matrix-balancing, this existence result is not known to lead to an efficient constructive algorithm.

The set balancing problem is a *discrete ham sandwich* problem in the following sense. The ham sandwich theorem in topology states that given n measurable sets in Euclidean n-space, there exists a hyperplane which splits all n sets precisely in half. The set balancing problem is thus a discrete analog of the topological ham sandwich theorem. It seeks to simultaneously split n sets of elements (as near as possible) into two halves.

Set balancing can thus be viewed as a special case of integer approximation with $p_j = 0.5$, $1 \le j \le r$. Beck and Fiala only consider the case $a_{ij} \in \{0,1\}$. We will consider a more general case in the next chapter.

Chapter 6

Rounding sans Randomness

6.1 Overview

In this chapter we study a technique for replacing randomized rounding by means of a deterministic polynomial-time procedure. We begin by considering the integer-approximation problem introduced in the last chapter. We analyze the quality of the integer approximation generated by randomized rounding. We then use an interesting "method of conditional probabilities" to develop a deterministic algorithm that performs as well as randomized rounding. Using this method, we indicate in section 6.4 how the randomized algorithms of chapters 3 and 4 can be made deterministic.

6.2 Integer Approximation Revisited

Recall that in the integer approximation problem, we were given a matrix A and a vector p whose components are reals in the interval [0,1]. We are to compute an integer vector q with components from {0,1} such that every co-ordinate of $A\ (p - q)$ is small in absolute value.

We first use randomized rounding to show the existence of a provably good vector q. We then show that the probabilistic existence proof can be converted, in a very precise sense, into a deterministic approximation algorithm. We wish to bound the *discrepancies*

$$\Delta_i = \left| \sum_{j=1}^{r} a_{ij}\ (p_j - q_j) \right| \tag{6.1}$$

in terms of the inner-products

$$s_i = \sum_{j=1}^{r} a_{ij}\ p_j \tag{6.2}$$

6.2.1 The Existence Proof

Suppose we set each q_j to 1 with probability p_j, independently of all the other components of q. Consider the random variable $\Psi_i = \sum_{j=1}^{r} a_{ij}\, q_j$.

$$E[\,\Psi_i\,] = \sum_{j=1}^{r} a_{ij}\, E[\,q_j\,] = s_i \tag{6.3}$$

THEOREM 6.1: There exists an integer approximation vector q such that

$$\Delta_i \leq s_i\ D(s_i, 1/2n) \tag{6.4}$$

PROOF: We will show that if the integers q_j are selected using randomized rounding, the resulting vector will satisfy (6.4) with non-zero probability. We thus establish the existence of such a q using the *probabilistic method* [9].

Let us say the i^{th} bad event β_i occurs if Δ_i exceeds the bounds of (6.4). Consider the random variable Ψ_i. By (6.3), its mean is $\sum_{j=1}^{r} a_{ij} p_j = s_i$. By the definitions above,

$$Pr[\,\Psi_i > s_i + s_i\ D(s_i, 1/2n)\,] < 1/2n \tag{6.5}$$

$$Pr[\,\Psi_i < s_i - s_i\ D(s_i, 1/2n)\,] < 1/2n \tag{6.6}$$

Thus the probability of bad event β_i is $< 1/n$. Let us say a vector q is "good" if no bad event occurs. Since there are n possible bad events β_i, the probability that the vector produced by randomized rounding is not good is $< n\,(1/n) = 1$. Thus a randomly chosen vector q is good with non-zero probability, and the theorem follows. □

Note the importance of the strictness of the inequalities of theorems 4.5 and 4.6 in the proof of the above theorem.

6.3 The Method of Conditional Probabilities

We now show that the probabilistic existence proof of theorem 6.1 can be converted to a deterministic construction of a good vector q. We use an interesting "method of conditional probabilities"; the deterministic algorithm will mimic the probabilistic existence proof in a very strong sense.

It is instructive to model the computation by means of a decision tree. Consider a complete binary tree T of r levels. Level j of T represents the setting of q_j to 0 or 1. For instance, if q_1 were set to 1, we proceed from the root of T to its left son; if q_1 were set to 0, we proceed to the right son. Thus, assigning the variables $q_1, q_2, \cdots$ in sequence to 0 or 1 amounts to walking down T from the root to a leaf. Each leaf corresponds to one of the 2^r possible vectors q. In terms of the bounds of theorem 6.1, we could then speak of "good" leaves and "bad" leaves.

Randomized rounding is equivalent to taking the left son at level j with probability p_j, and the right son with probability $1-p_j$; the choices at the various levels are made independently. Theorem 6.1 tells us that T always has a good leaf. Our task is to walk down the tree to a good leaf in deterministic polynomial time.

At a typical stage of the computation, we are at some node at level j in the tree, $1 \le j \le r$. We have already walked down the first $j-1$ levels, assigning $q_1, \cdots, q_{j-1}$ in the process. We now wish to proceed to one of the two sons of the current node (i.e., assign q_j to 0 or 1).

Suppose (although this will not be the case) that randomized rounding were executed at levels j through r. Let $P_j(q_1, \cdots, q_{j-1})$ denote the conditional probability of a bad event occurring given $q_1, \cdots, q_{j-1}$ and assuming that randomized rounding is used to compute $q_j, \cdots, q_r$. Then

$$P_j(q_1, \cdots, q_{j-1}) = \\ p_j \; P_{j+1}(q_1, \cdots, q_{j-1}, 1) + (1-p_j) \; P_{j+1}(q_1, \cdots, q_{j-1}, 0) \qquad (6.7)$$

$$\Rightarrow \; P_j(q_1, \cdots, q_{j-1}) \ge \\ \min\{P_{j+1}(q_1, \cdots, q_{j-1}, 1), P_{j+1}(q_1, \cdots, q_{j-1}, 0)\} \qquad (6.8)$$

The following algorithm then suggests itself: for $j = 1$ to r, at level j we set q_j to 0 or 1 so as to minimize $P_{j+1}(q_1, \cdots, q_{j-1}, q_j)$. The existence of at least one good leaf (theorem 6.1) implies that $P_1 < 1$; combining this inductively with equation (6.8), we conclude that

$$1 > P_1 > P_2(q_1) > P_2(q_1, q_2) > \cdots > P_r(q_1, \cdots, q_{r-1}) > P(\mathit{Leaf}) \quad (6.9)$$

where *P(Leaf)* is the probability that we have reached a bad leaf. Every leaf is either bad or good; accordingly, *P(Leaf)* is either 0 or 1. But our procedure takes us to a leaf for which *P(Leaf)* < 1, so *P(Leaf)* *must* be 0 and the leaf we have reached *must* be good.

From an algorithmic standpoint, the difficulty lies in computing these conditional probabilities efficiently. Let $U_j(q_1, \cdots, q_{j-1})$ be an upper bound on $P_j(q_1, \cdots, q_{j-1})$ for all j, that can be *efficiently computed.* Further, let $U_j(q_1, \cdots, q_{j-1})$ have the property that

$$U_j(q_1, \cdots, q_{j-1}) \geq \min\{U_{j+1}(q_1, \cdots, q_{j-1}, 1), U_{j+1}(q_1, \cdots, q_{j-1}, 0)\} \quad (6.10)$$

Our algorithm would then be: for $j = 1$ to r, assign to q_j that value which minimizes $U_{j+1}(q_1, \cdots, q_{j-1}, q_j)$.

At each stage:

(a) The function U is an upper bound on the function P (temporarily omitting subscripts, *etc.* for brevity);

(b) By (6.10), U never rises in the course of the computation;

(c) The algorithm can be run efficiently since U can be computed efficiently.

We call this the *method of pessimistic estimators*, since at each stage we bound the probability of failure from above. If we could find a pessimistic estimator such that $U(\mathit{root}) < 1$, we are guaranteed to succeed.

6.3.1 Moment-Generating Functions and the function U

We now derive a suitable function U; the manner in which we do so parallels the proofs of the bounds in theorems 4.5 and 4.6, and the existence proof of theorem 6.1. Recall that we said that the i^{th} bad event β_i is said to occur if, for the vector q that we compute, the i^{th} discrepancy Δ_i exceeds the limits prescribed by theorem 6.1. Let

$$L_{i+} = s_i [\, 1 + D(s_i, 1/2n)\,] \tag{6.11}$$

$$L_{i-} = s_i [\, 1 - D(s_i, 1/2n)\,] \tag{6.12}$$

Thus, bad event β_i occurs when $\Psi_i > L_{i+}$ or when $\Psi_i < L_{i-}$.

6.3.1.1 Bounding the initial probability of a bad event

Consider the probability of bad event β_i resulting from Ψ_i exceeding L_{i+}, at the beginning of the computation (at the root of T). Following (4.15), for any real $t_i \geq 0$

$$\Pr [\, \Psi_i > L_{i+}\,] < e^{-t_i L_{i+}} \prod_{j=1}^{r} [\, p_j e^{a_{ij} t_i} + 1 - p_j\,] \tag{6.13}$$

6.3.1.2 Updating the Bound

We now consider the effect of setting q_k to 0 or 1. Suppose some q_k were assigned the value 1. Given this information, the conditional probability that Ψ_i exceeds L_{i+} is the probability that the sum of the remaining random variables exceeds $L_{i+} - a_{ik}$. This is bounded above by

$$e^{-t_i (L_{i+} - a_{ik})} \prod_{j \neq k} \mathrm{E} [\, e^{t_i a_{ij} q_j}\,] = e^{-t_i L_{i+}} e^{a_{ik} t_i} \prod_{j \neq k} [\, p_j e^{a_{ij} t_i} + 1 - p_j\,] \tag{6.14}$$

Thus the conditional probability of Ψ_i exceeding L_{i+} given $q_k = 1$ is just bounded by replacing the moment-generating term $p_k e^{a_{ik} t_i} + 1 - p_k$ by $e^{a_{ik} t_i}$ in the bound function - an intuitively correct idea. Likewise, it can be verified that setting $q_k = 0$ has the effect that the term $p_k e^{a_{ik} t_i} + 1 - p_k$ is replaced by 1.

6.3.1.3 The function U

The probability that *any one* of the random variables Ψ_i exceeds its upper limit is bounded above by the sum of the individual probabilities in (6.13):

$$\sum_{i=1}^{n} e^{-t_i L_{i+}} \prod_{j=1}^{r} [\, p_j e^{c_{ij} t_i} + 1 - p_j\,] \tag{6.15}$$

So far, we have discussed deviations of the random variables Ψ_i *above* their means; a similar analysis gives a bound on the probability that for some i, Ψ_i falls *below* its lower limit

L_{i-}. Adding this bound to (6.15), we obtain an upper bound on the probability that *any* β_i occurs.

$$U(root) =$$

$$\sum_{i=1}^{n} \left[e^{-t_i L_{i+}} \prod_{j=1}^{r} [\, p_j e^{a_{ij} t_i} + 1 - p_j \,] + e^{-t_i L_{i-}} \prod_{j=1}^{r} [\, p_j e^{-a_{ij} t_i} + 1 - p_j \,] \right] < 1 \qquad (6.16)$$

The last inequality stems from our proof of theorem 6.1; indeed, we used the above bound (through theorems 4.5 and 4.6) in its proof, with $t_i = \ln [\, 1 + D(s_i, 1/2n) \,]$. We use these values t_i in our computation of U. Equation (6.16) gives us the value of U at the root of T. We saw (section 6.3.1.1) the effect of assigning some q_k to 0 or 1; the updated value of U is always an upper bound on the probability of a bad event, conditioned by the assignment of q_k.

It remains to show that for any k, one of the two possible assignments of q_k reduces the value of U. We will show that this property is satisfied by $U(root)$; a similar argument applies to subsequent stages. We thus examine the effect of setting q_1. Equation (6.16) for U can be written as

$$\sum_{i=1}^{n} B_i \,(p_1 e^{a_{i1} t_i} + 1 - p_1) + C_i \,(p_1 e^{-a_{i1} t_i} + 1 - p_1) = \qquad (6.17)$$

$$p_1 \sum_{i=1}^{n} (\, B_i \, e^{a_{i1} t_i} + C_i \, e^{-a_{i1} t_i} \,) + (1 - p_1) \sum_{i=1}^{n} (B_i + C_i) \qquad (6.18)$$

where B_i and C_i are fixed numbers. If q_1 is set to 1, the new value of U is

$$\sum_{i=1}^{n} (\, B_i \, e^{a_{i1} t_i} + C_i \, e^{-a_{i1} t_i} \,) \qquad (6.19)$$

while if q_1 is set to 0 the new value of U is

$$\sum_{i=1}^{n} (\, B_i + C_i \,) \qquad (6.20)$$

Since (6.18) is a convex combination of (6.19) and (6.20), it is no less than the smaller of (6.19) and (6.20). Thus we can proceed from the root of T to one of its sons in such a manner that U does not rise. A similar argument for the general step (updating U as we proceed) shows that the value of U does not rise in the course of the computation. Thus $1 > U(Leaf) > P(Leaf)$.

THEOREM 6.2: The method of pessimistic estimators yields in deterministic polynomial time an integer vector q such that

$$\Delta_i \leq s_i \; D(s_i, 1/2n) \;, \quad 1 \leq i \leq n \tag{6.21}$$

This improves on the result of Beck and Fiala [2] who studied the case $a_{ij} = 0$ or 1 (theorem 5.3). Using reasoning similar to (3.38) and (3.39), we find that the discrepancies guaranteed by our algorithm are asymptotically smaller than those of the Beck-Fiala algorithm whenever s_i is $o(n)$. Even when s_i grows as n, our constant factors are better.

We will now consider applications of the theory developed above to approximately solving our integer programs in deterministic polynomial time.

6.4 Deterministic Rounding

The integer approximation problem we have just studied is typical of the rounding problems we have been concerned with throughout this thesis. It is therefore reasonable to expect that the techniques that removed randomization in the case of integer approximation should work for our other integer programming problems. We show now that this is indeed the case. We begin by outlining the method of pessimistic estimators as applied to the global routing problem. We then state theorems concerning the other integer programs we have considered; the details are straightforward and are omitted.

6.4.1 Global Routing

In section 3.1 we formulated the global routing problem of chapter 2 as an integer linear program. We require this formulation, together with the analysis of section 3.5, in applying the method of pessimistic estimators. We now give an outline of the deterministic equivalent of randomized rounding as applied in section 3.3, drawing suitable analogies with our treatment of integer approximation in section 6.3.

We route the nets $r_i \in R$ sequentially. Recall that $T(r_i)$ is the set of possible routings for net r_i; t_{ij} is the i^{th} route in $T(r_i)$. The indicator variable x_{ij} in the integer program denotes the presence or absence of t_{ij} in the routing.

The decision tree T now has $|T(r_i)|$ branches at level i. Randomized rounding consists of choosing the j^{th} branch at level i with probability x_{ij}^*, where x_{ij}^* is the value assigned to variable x_{ij} in the linear program solution.

We may set ε to 1 in theorem 3.6 and view it as an existence proof, analogous to theorem 6.1. Note that the strictness of the bound of theorem 3.5 is important for the existence proof. The method of conditional probabilities then applies, just as in section 6.3. We now indicate how to construct a pessimistic estimator for the failure probability.

Each sum-to-one constraint (3.1) gives rise to a moment-generating term. Let

$$C^E = C^* \left[1 + D(C^*, \frac{\varepsilon}{N}) \right] \tag{6.22}$$

and

$$t = \ln \left[1 + D(C^*, \frac{\varepsilon}{N}) \right] \tag{6.23}$$

Also, let $\rho_e = \{ i : \text{ some tree in } T(r_i) \text{ contains } e \}$ for all $e \in E$. Corresponding to (6.13), the probability of the width of an edge e exceeding C^E is bounded by

$$e^{-tC^E} \prod_{i \in \rho_e} \sum_{j : e \in t_{ij}} x_{ij}^* e^t \tag{6.24}$$

Summing over all edges, we obtain the upper bound function

$$U = \sum_{e \in E} e^{-tC^E} \prod_{i \in \rho_e} \sum_{j : e \in t_{ij}} x_{ij}^* e^t \tag{6.25}$$

For $i = 1$ to $|R|$, we choose the tree t_{ij} which minimizes the function U. By reasoning similar to that leading to theorem 6.2, we have the following theorem corresponding to theorem 3.6.

THEOREM 6.3: The method of pessimistic estimators produces a routing of width

$$\leq C^{(I)} \left[1 + D(C^{(I)}, \frac{1}{N}) \right] \tag{6.26}$$

A similar deterministic algorithm can be devised for the multicommodity flow approach of sections 3.6 and 3.7.

6.4.2 Packing and Maximum Multicommodity Flow

We now state the "deterministic" versions of various theorems proved by randomized rounding. Corresponding to theorem 4.7, we have:

THEOREM 6.4: Scaling and pessimistic estimators find an integer k-matching of cardinality

$$\geq M^s \left[1 - D(M^s , \frac{1}{n}) \right] \tag{6.27}$$

For the maximum multicommodity flow problem, we have corresponding to theorem 4.4:

THEOREM 6.5: Path-stripping, scaling and pessimistic estimators find a multicommodity flow of total magnitude

$$\geq F^s \left[1 - D(F^s , \frac{1}{N}) \right] \tag{6.28}$$

Chapter 7
Conclusion

7.1 Main Results

We conclude by summarizing the main contributions of this thesis. From a theoretical standpoint, this work re-examines an old problem: that of approximately solving a computationally hard integer program by using the solution to its rational relaxation. Our results pertain to a collection of integer programs of practical interest. From a practical point of view, this work tackles an important problem in the design of integrated circuits. The algorithms developed here offer performance guarantees, and our preliminary experimental experience is encouraging.

We now list the main results in greater detail.

(1) The problem of global routing in gate-arrays is shown to be NP-Complete, thus laying a basis for the search for heuristics for this problem.

(2) The global routing problem is formulated as an integer program, and a provably good approximation algorithm is presented for this integer program. The approximation algorithm uses the solutions to the relaxation linear program; these solutions are rounded using an interesting randomized method. The quality of the approximation is proved using new bounds on the tail of the binomial distribution.

(3) Initial experimental work in global routing with our randomized algorithm has produced encouraging results. Further work with larger gate-arrays is necessary before the practical usefulness of the method becomes clear. Of particular concern is the size (and consequent cost in computer time) of the linear programs that have to be solved. On the other hand, it is possible to trade off linear program size and routing quality by controlling the choice of routes used in an experiment.

(4) The randomized rounding algorithm has been shown to be applicable to certain pack-

ing and multicommodity flow problems from combinatorial optimization and operations research, and to some combinatorial problems concerning set balancing.

(5) An interesting "method of conditional probabilities" is used to convert randomized rounding to a deterministic procedure yielding the same performance guarantees. The deterministic procedure is made polynomial-time by using "pessimistic" upper bounds on the probabilities of certain events occurring in the algorithm. This is a somewhat surprising result. While it says that randomization is unnecessary for our rounding problems, it is interesting that the deterministic algorithm works by mimicking the proof of the randomized algorithm. Although our deterministic algorithm could thus be derived from purely combinatorial methods, it is the use of the probabilistic method that led to its conception and understanding.

7.2 Further work

A number of avenues for further research are now apparent. We derived a Chernoff-like upper bound on the tail probability of the weighted sum of Bernoulli trials in theorems 4.5 and 4.6. The bound is tightest when all the probabilities $p_1 \cdots p_r$ are equal. It would be interesting to derive lower bounds on the tail probability, especially when the probabilities $p_1 \cdots p_r$ assume disparate values.

In the analysis of randomized rounding in section 3.5, and again in the construction of the pessimistic estimator in section 6.3, we always sum the probabilities of all bad events. These bad events are surely correlated. Is it possible to prove a tighter bound using algebraic properties of the coefficient matrix? When the sum of the entries in every column of the coefficient matrix is bounded above by some number g, Karp *et al.* [18] give a technique for rounding such that all discrepancies are bounded above by g.

The method of conditional probabilities was used in chapter 6 to convert our randomized rounding algorithms into deterministic ones. What other randomized algorithms/constructions can be made deterministic using the method of conditional probabilities? In order to make the conversion computationally efficient, we have to find good pessimistic estimators for the conditional probabilities that arise in the problem. One cannot always hope to compute a tight pessimistic estimator in polynomial time.

Given that parallel algorithms are currently in vogue, it is natural to ask whether our deterministic rounding algorithm can be parallelized. It is to be noted that randomized rounding can be done in constant parallel time due to our use of independent coins.

Discrepancy bounds of the form of theorem 6.1 define a set of $2n$ halfspaces in Euclidean r-space. Theorem 6.1 assures the existence of a lattice point (all r co-ordinates are integral) in the convex body defined by the common intersection of these halfspaces. Can this be related to Minkowski's convex body theorem [26]?

7.3 Acknowledgements

Thanks ....

To Clark Thompson, for being a kind teacher and friend, and a patient and forever encouraging guide. I hope I will be half as good an advisor as he was, some day.

To Richard Karp, for his inspiration and valuable advice throughout this work, and for imbuing the engineer in me with a sense of mathematical discipline.

To Dorit Hochbaum, for her friendly advice and encouragement, and for serving on my thesis committee.

To Joel Spencer, for his advice on polishing the results in chapter 6. Also, theorems 3.2, 3.4 and 3.6 together with the resulting corollaries were derived jointly with him.

To Antony Ng, for performing the experiments described in section 3.8 which lend practical significance to this work. It was a pleasure to work with him.

To Tom Leighton and Charles Leiserson for their keen interest in this work, and for several suggestions. The result described in section 3.7 was obtained jointly with them.

To my parents, for their constant enthusiasm and encouragement during my graduate studies, and for their sacrifices in ensuring that I had the best education possible.

And to my wife, Srilatha, for her support and love throughout this work.

I also wish to thank Semiconductor Research Corporation's grant SRC 82-11-008, the National Science Foundation's grant ECS-8110684, an IBM Doctoral Fellowship and a University of California Fellowship for funding this research and supporting my doctoral studies.

References

1. D. Angluin and L. G. Valiant, "Fast probabilistic algorithms for Hamiltonian circuits and matchings," *Journal of Computer and System Sciences*, vol. 19, pp. 155-193.

2. József Beck and Tibor Fiala, ""Integer-Making" Theorems," *Discrete Applied Mathematics*, pp. 1-8, 1981.

3. R. Beresford, "Evaluating Gate-Array Technologies," *VLSI DESIGN*, vol. V, no. 1, pp. 48-52, Jan '84.

4. Patrick Billingsley, *Probability and Measure,* John Wiley & Sons, 1979.

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

6. H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations," *Annals of Math. Stat.*, vol. 23, pp. 493-509, 1952.

7. G.B. Dantzig, *Linear Programming and Extensions,* Princeton University Press, Princeton, NJ, 1963.

8. Alfred E. Dunlop and Brian W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits," *IEEE Trans. on Computer-Aided Design*, vol. CAD-4, no. 1, pp. 92-98, Jan. 1985.

9. Paul Erdős and Joel Spencer, *The Probabilistic Method in Combinatorics,* Academic Press, 1974.

10. S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 691-703, Dec. 1976.

11. Z. Füredi, "Maximum degree and fractional matchings in uniform hypergraphs," *Combinatorica*, vol. 1, no. 2, pp. 155-162, 1981.

12. Abbas El Gamal, "Two-dimensional stochastic model for interconnections in master-slice integrated circuits," *IEEE Transactions on Circuits and Systems*, vol. CAS-28, pp. 127-137, Feb 81.

13. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman & Company, San Francisco, 1979.

14. M. Hanan, "On Steiner's Problem with Rectilinear Distance," *SIAM Journal of Applied Math.*, vol. 14, no. 3, pp. 255-265, 1966.

15. T. C. Hu and M. T. Shing, "A Decomposition Algorithm for Circuit Routing," *Mathematical Programming Study*, vol. 24, pp. 87-103, 1985.

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

17. R.M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, ed. R.N. Miller, J.W. Thatcher, pp. 85-104, Plenum Press, New York, 1972.

18. R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani, "Global Wire Routing in Two-Dimensional Arrays," *Proc. 24th Annual Symp. on Foundations of Computer Science*, pp. 453-459, October 1983.

19. L. G. Khachian, "A Polynomial Algorithm for Linear Programming," *Soviet Math. Doklady*, vol. 20, pp. 191-194, 1979.

20. D. E. Knuth, "Big Omicron and Big Omega and Big Theta," *SIGACT News*, vol. 8, no. 2, pp. 18-24, April-June 1976.

21. Mark R. Kramer and Jan van Leeuwen, "Wire-Routing is NP-Complete," Technical Report RUU-CS-82-4, Rijksuniversiteit Utrecht, The Netherlands, February 1982.

22. J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph," *Proc. Amer. Math. Soc.*, vol. 7, pp. 48-50, 1956.

23. E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, 1976.

24. C. Y. Lee, "An algorithm for path connections and its applications," *IRE Trans. Electronic Computers*, vol. EC-10, pp. 346-365, September 1961.

25. L. Lovász, "On the ratio of optimal and fractional covers," *Discrete Mathematics*, vol. 13, pp. 383-390, 1975.

26. H. Minkowski, *Geometrie der Zahlen*, Leipzig, Teubner, 1910.

27. Antony P-C Ng, Prabhakar Raghavan, and Clark D. Thompson, "Experimental Results with a Linear Program Global Router," *Submitted to Computers and Artificial Intelligence*, April 1986.

28. Antony P-C Ng, Prabhakar Raghavan, and Clark D. Thompson, "A Specification Language for Describing Rectilinear Steiner Tree Configurations," *ACM Design Automation Conference*, 1986.

29. 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.

30. John E. Olson, "A Balancing Strategy," *Journal of Combinatorial Theory, Series A*, vol. 40, pp. 175-178, 1985.

31. Christos H. Papadimitriou and Kenneth Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., 1982.

32. C. Quinn and M. Breuer, "A force directed component placement procedure for printed circuit boards," *IEEE Trans. Circuits and Systems*, vol. CAS-26, Jun. 1979.

33. Prabhakar Raghavan and Clark D. Thompson, "Randomized Routing in Gate-Arrays," CSD/84/202, Computer Science Division, UC Berkeley, Sep. 1984.

34. Prabhakar Raghavan and Clark D. Thompson, "Provably Good Routing in Graphs: Regular Arrays," *Proceedings of the Seventeenth ACM Symposium on Theory of*

Computing, pp. 79-87, ACM, New York, May 1985.

35. Prabhakar Raghavan and Clark D. Thompson, "Randomized Rounding: A technique for provably good algorithms and algorithmic proofs," *To appear in Combinatorica*, 1986.

36. Joel Spencer, "Balancing Games," *Journal of Combinatorial Theory, Series B*, vol. 23, no. 1, pp. 68-74, Aug. 1977.

37. Joel Spencer, "Six Standard Deviations Suffice," *Transactions of the American Mathematical Society*, vol. 289, no. 2, pp. 679-706, June 1985.

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