Rational Control of Multi-level Stochastic Design

J. STORRS HALL, LOUIS STEINBERG, BRIAN D. DAVISON
Laboratory for Computer Science Research
Rutgers University, New Brunswick, NJ

ABSTRACT

Search in a complex design space (e.g. in VLSI design) is only feasible when it is factored into smaller ones. A common and useful factoring is the hierarchy of abstraction. Such problems are then attacked by linking together stochastic search programs. At each point in the design process it is necessary to choose whether to continue looking for a better design at the current level, or working further on some design at a higher or lower level. We present a rational, i.e. utility-based, method for this choice, together with some experiments in the domain of VLSI placement and routing[1].

INTRODUCTION

The idea of structuring the process of design as a series of "levels of abstraction" is both appealing in theory and important in such real-world domains as microprocessor design. However, existing design automation in such domains is usually restricted to single level applications, e.g. VLSI channel routers, executed sequentially.

Design at a given level, such as module placement, often involves search. We are motivated in the present work by the notion that an overall control of search at all levels can influence (and improve) the search at one level by virtue of knowledge gained at other levels. Rational control (Russell and Wefald, 1991) is particularly apropos to design in that most of the work at higher abstraction levels is done to decide which subspace to search at lower levels.

RATIONAL BLIND RANDOM SEARCH

As an example, and as a basic component in the overall system, consider the problem of rational blind random search. Let us assume that we are searching in a space for which there is an evaluation function *S(x)* mapping objects into some numerical range, and value function *V* such that *V(S(x))* is the value of object x, and the distribution of *S* is known, and given by the density function *D*.

Furthermore let us assume that the cost to us of generating and evaluating one object is constant, and is measured in the same units as the valuations; we will designate it as *c*.

Also, let *z(x)* be the normal density function ("bell curve"), and define

$$tail(F,x) = \int_x^\infty F(y)dy$$

for any function F for which it is finite.

The "standard score" of object x is $\frac{S(x)-\mu}{\sigma}$. Since the scale of the range of S (the domain of V) is fairly arbitrary, we will generally assume that S returns a standard score in D, i.e. that $\mu = 0, \sigma = 1$. Further, wherever it is clear from context, we will say "object" instead of "standard score of the evaluation function of an object".

BLIND RANDOM SEARCH

A blind random search is a sequence of independent trials, each with probability *p* of success (the criterion of success will be discussed later), which stops when a trial is successful. The expected number of trials up to and including the first success in such a sequence is 1*/p*.

Thus the expected effort in blind random search over a normally distributed population to find a score better than *x* is 1*/tail(z,x)* trials.

Now as we perform the sequence of trials, we save the value of the best score seen so far; we will call this value *b*. We define *ival(x|b)*, the incremental value of finding a new score *x* given that we have one of score *b*, as

$$ival(x|b) = \max(0, V(x) - V(b)).$$

Let *probe(V,D,b)* be the value of a single blind random probe in a population with distribution *D* given we have a design of score *b*:

$$probe(V,D,b) = tail(D(x)ival(x|b),b).$$

[1] This work is supported by DARPA under grant number DABT-63-93-C-0064.

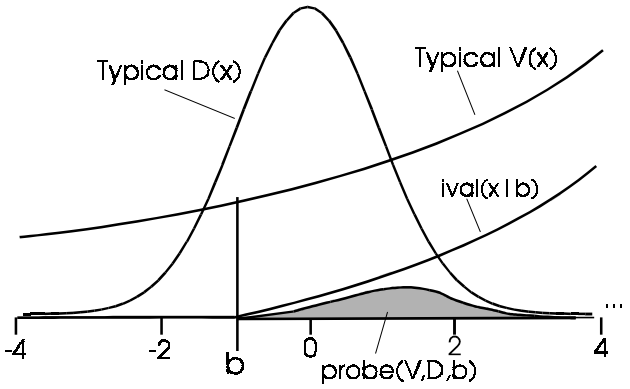

Figure 1. D(x) is the density function of the distribution of the objects. V(x) is the value of the objects. *b* is the best value we have found so far; *ival* gives the incremental value of finding a new object (its value minus that of b, or zero). *probe* is the area under the product of *ival* and *D*, and gives the total expected value of a new probe, given a current best score of *b*.

RATIONAL RANDOM SEARCH

If the expected value of doing one more generate-and-evaluate is greater than the cost of doing it, we should do it; otherwise we should stop. Rational random search (RRS) of a population of distribution *D* therefore consists of evaluating elements while *probe(V,D,b)* > *c* (updating *b* if appropriate after each trial).

In a case where *D* and *c* are known, we can precompute the score (not the object) *g* such that *probe(V,D,g)=c*. The tail of *D* beyond *g* is called the terminal region for the search, and *g* is the lower bound for the score the search will produce. Implicitly, *g* is a function *g(V,D,c)* of the value, distribution, and cost.

RRS can be seen to be a blind search sequence terminating with probability *tail(D,g)* (the probability mass of the terminal region). Thus its expected cost is *c/tail(D,g)*. We can improve the implementation by precomputing *g* and searching until we find a score greater than *g*; there is no need to maintain *b* and compute *probe(V,D,b)* at each step.

The expected absolute as opposed to incremental value under the tail above *x* is:

$$tail(V \cdot D, x) = \int_x^\infty V(y)D(y)dy$$

The expected value *EV* (implicitly *EV*(*V*,*D*,*c*)) of the search is *tail(V.D,g)/tail(D,g)*, that is to say, the expected value under the tail scaled up to a probability of 1. The expected cost *EC* (implicitly *EC*(*V*,*D*,*c*)) is, as mentioned, *c/tail(D,g)*. By definition of *g*,

$$c = probe(V, D, g) = \int_g^\infty D(x)(V(x) - V(g))dx$$

$$= \int_g^\infty D(x)V(x)dx - V(g)\int_g^\infty D(x)dx$$

so $c = tail(V.D,g) - V(g)tail(D,g)$. Dividing through by *tail(D,g)*,

$$EV = EC + V(g),$$

for any functions *V* and *D*.

HILLCLIMBING

In spaces with a reasonable neighborhood function, hillclimbing is a more efficient method of search than blind random search. For example, for one typical circuit, randomly selected layouts had a mean area of 501.3 and standard deviation of 26.9. Hillclimbing from these layouts cost an average 1833.6 evaluations and found an average area of 367.3. This corresponds to a standard score of 4.96296. Random search would have been expected to take more than 3 million evaluations to produce the same result.

However, we can analyze multi-start hillclimbing search (i.e., repeated hillclimbs, each starting from a random point in the space) as a random search in the population of hilltops, and with a unit cost which is the expected number of evaluations in one hillclimb times the cost of one evaluation. The above analysis applies with each hillclimb considered a single probe.

MULTI-LEVEL SEARCH

In our design system framework, the problem is broken into a cascaded sequence of abstraction levels. The objects of each level are designs that are complete in their own terms (for example, a netlist that represents the entire circuit being designed), but each of which could be implemented by many possible objects at lower (more concrete) levels.

In the following we will number the levels so that higher (more abstract) levels have higher numbers, and lower (more concrete) levels have lower numbers. We will refer to objects, scores, and values an level *n* as *n*-objects, *n*-scores, and *n*-values. For example, a placement might be a 5-object, and have a set of routings, 4-objects, corresponding to it.

The only value represented by any 1-object is the expected value of the 0-level objects it lets us generate. In particular, if the 1-score of an object gave us the distribution of the corresponding 0-subspace (and knowing the cost of 0-search), we could derive the expected value at level 0 corresponding to the 1-object, using the methods of the previous section.

CLASSICAL TOP-DOWN DESIGN

Consider classical top-down design as a baseline case. Let us assume for the sake of argument that there is no information to be had from designing at level *n*, which can inform the design process at level *n*+1. That is, any *n*+1-object with a given *n*+1-score is equivalent to any other *n*+1-object with the same score in terms of the search and design at lower levels.

We can without loss of generality use standard scores for each level. This is because higher-level scores are intended solely as estimates of lower-level scores, so that if the mean or standard deviation of a higher-level population differs from that of a lower, applying a linear correction to the higher-level score is not only allowable but required. We will make the further assumption that scores are normally distributed, and that scores at adjacent levels have joint normal distributions. Then the *score-contingent distribution* at level n ${}_sD_n$ given an n+1-score s_{n+1} is normal with mean ρs_{n+1} and standard deviation $\sqrt{1-\rho^2}$. (ρ is the Pearson's correlation coefficient of the joint normal distribution.)

Then for level 0, we can derive a g value as in the single distribution case, for each possible score-contingent distribution. That is, we define a function $g_0(s_1)$ which is the cutoff point for search at level 0 in the distribution expected given we are searching below a 1-object of score s_1. Since the expected value of the search in this distribution is EC+V(g), the overall profit of searching it is V(g). Thus the value of having a 1-object of score s_1 is $V(g_0(s_1))$. We now define a new value function $V_1(s_1) = V(g_0(s_1))$. We can now apply the same algorithm to level 1 to find a value function V_2, and so forth.

For purposes of the algorithm, we will assume we have a "root object" o_{n+1}.

Algorithm W (waterfall, or standard top down design)

[1] Set object x=o_{n+1} and distribution d=D_n. Set i=n.
[2] Calculate g from d, c_i, and V_i.
[3] Generate objects from d until one is found with a score higher than g. Set x to this object.
[4] If i is 0, we are done, and x is our result.
[5] Otherwise, set d to the contingent distribution at level i-1 defined by s(x).
[6] Decrement i and go to step 2.

Algorithm *W* is optimal given the assumption that there is no more information usable at level *n* than the score-contingent distribution at level *n*-1 for any given object.

The analysis given above for rational random search at a single level is a good example of the pitfalls of using a "single-step" assumption. It assumes that probes are independent. However, in the multiple-level case, while probes at one level may be independent, probes at different levels are very dependent, as are multi-level sequences of probes. The only way to use the single-step style analysis effectively here is to analyze all the sequences of probes one might conceivably do; i.e. to analyze algorithms.

UPWARD INFORMATION FLOW

In our experience, raw evaluations of 1-level phenomena are almost entirely uncorrelated with raw 0-level scores (in experiments with placement and routing). This is ameliorated by the addition of 1-level scoring heuristics which attempt to predict 0-level performance, as well as including 1-level phenomena in the 0-level evaluation, as mentioned above. These together induce a moderate correlation (i.e. with correlation coefficients in the neighborhood of 0.4).

Given a specific *n*+1-object, we can get a better estimate for the mean and standard deviation of its contingent distribution at level *n* by sampling. This sampling has both a cost and a value, since the means of the *n*+1-object-contingent distributions vary considerably from the mean of the corresponding score-contingent distribution. In experiments in the placement and routability levels, the standard deviation for score-contingent distributions were about twice those of object-contingent distributions.

This empirical observation invalidates the assumptions that underly the optimality of the waterfall algorithm. In fact, if the variability of object-contingent distributions within the score-contingent one if large enough, it can fail to terminate at all. We next address that variability .

We can begin the general case by finding a g-function for level 0 based on object-contingent distributions instead of score-contingent ones. However, at level 1, there is no direct correspondence between these distributions and scores. We define a new family of distributions OVD which are the distributions of objects at level i+1 which have the same level i object-dependent distribution. The score-contingent distribution at level i is the convolution of OVD and object-contingent distribution.

The key to the analysis of sampling is to understand that sampling is worthless unless it changes the decision we would have made using an object's heuristic score alone.

THE WAGES OF SIN

There are two classes of error to be considered: First, rejecting an object whose actual score is below g but whose "hidden true score" is above it; and second, accepting an object whose actual is above but whose true is below, resulting in a substandard object-contingent distribution on the next level.

In the first case, the penalty is severe. Since we had an acceptable search-terminating object, if only we had known it, we will now proceed to do the entire search again (almost) uselessly. Since the search is random, the

probes we have done to date account for nothing; the expected cost of continuing is the same as the initial expected cost, $c_i/tail(D,g)$. The expected value of the improvement we get over the object we have is guaranteed to be less than EC by definition of *g*.

In the second case, the contributions are reversed: we're saving effort (though we shouldn't) and getting a less valuable result. Since the tail of OVD under consideration this time is below *g*, we're guaranteed that the difference in value is *greater* than the saved search cost.

If $OVD_x(y)$ is the OVD of actual score *x* and "hidden true score" *y*, and V_i is the backed-up value function (i.e. V(*g*(object-contingent distribution))), then, for $x<g$, we want to sample if

$$V_i(g)tail(OVD_x, g) + c_{i-1} < tail(V_i \cdot OVD_x, g)$$

and therefore *k*, the lower bound for sampling, is the value of *x* for which this is an equality; and for $x>g$, sample if

$$V_i(x) + tail(OVD_x, g)V(g) + c_{i-1} < V(g) + tail(OVD_x, g)V(x)$$

and similarly the upper-bound value *h* is the value of *x* that makes this equal.

BACKED-UP EXPECTED PROBE COST

The overall search is now the same as in the waterfall case, except that for the region near g where the cost of sampling is less than the expected error cost, we sample. For the major part of this region, the only sampling we can afford is a single probe at level *i-1*; we will do a simplified analysis assuming all samples are one probe. The lower bound of the sampling region is *k* and the upper bound *h* ($k < g < h$). We weight the backed-up cost *bc* with the likelihood the sample will be taken:

$$bc_n = c_n + c_{n-1}(tail(k_n) - tail(h_n)) \quad .$$

We can use this formula for cost and re-calculate *g*:

$$probe(V_n, D_n, g_n) = c_n + bc_{n-1} \cdot tail(k_n)$$

and iterate until *g* converges. Unless OVD is very wide, this is relatively fast, since the sampling region will occupy a very small part of the search distribution and thus contribute little to the cost.

Now suppose we have an *n*-object whose score is above *k*. We evaluate one *n*-1-object beneath it. The resulting *n*-1-score, let us call it *x*, tells us something about the distribution of the *n*-1-objects below the *n*-object. The simplest procedure, and a usable heuristic, is simply to take *x* as an estimate of the *n*-1-distribution mean; *i.e.* throw away the *n*-object if *x* is less than *g*, and accept it otherwise.

RATIONAL SAMPLING

A more precise procedure is to use the standard statistical sampling estimates of the *n*-1-distribution given by *x* and possible subsequent samples. The sampling can continue as long as the cost of the next sample is less than the remaining cost due to error given the distribution as determined by the samples taken thus far. The main phenomenon is that as we sample, we know more, so OVD, the expected variability in object-contingent distributions, shrinks.

OVD gives us the probability the real object-contingent distribution (OCD) will be centered at point *x* (More precisely, within *dy*/2 of *x*, $p = OVD_x(y)dy$. The *dy*' s can be carried through the entire computation and then cancel out, so we will ignore them in the following). The score-contingent distribution SCD_x is the convolution of OVD*x* and OCD. It gives us the *ab initio* probability of finding a sample at point *x*.

If *d* is the event of finding OCD centered at point *x*, and *s* is the event of finding a sample (whose score we shall also designate *s*), p(*d*|*s*) is a function of *x* giving the probability of OCD being there given the sample *s*. I.e. it is what OVD should look like given the sample. Using Bayes' theorem twice,

$$p(d|s) = \frac{p(s|d)\,p(d)}{p(s)}, \text{ i.e., } \frac{OCD_x(s)OVD_x}{SCD_x(s)}$$

(where $OCD_x(s)$ means the value at *s* of the OVD centered at *x*). Note that the numerator is a product of functions and the denominator is a constant.

Thus to get the new OVD we need only multiply the old one times a function that is the OCD shifted to and reflected about the sample, and divide by the value of SCD at the sample.

The only proviso is that for a possible second sample, the true prior isn' t the original SCD any more, but rather the convolution of OCD and the new OVD from the first sample. Let us call the appropriate true prior, which represents the best estimate of the actual distribution we have, D.

GENERALIZING MULTI-LEVEL SAMPLING

The major problem with full sampling is that the nearer the "secret score" of *x* to *g*, the more sample probes will be necessary to determine on which side it really is. This can be largely ameliorated by the expedient of using the same probes for sampling and the next level of search. This means that we will pay sampling costs for only those candidates we ultimately reject; probes for successful samples will be charged to the account of the next level.

To this end we can compute a *g* value for D, the best estimate of the sampled distribution. If the value

$V_{i-1}(g(D)) > V_i(g_i)$,

the sample indicates the current i-level candidate is good, since after all V_i is defined as an estimate of V_{i-1} for g of the distribution we expect to be searching.

As long as $V_{i-1}(g(D)) > V_i(g_i)$, then, we should continue sampling/searching. If a probe falls above $g(D)$, we should accept in the waterfall sense at level i-1.

If $V_{i-1}(g(D))$ falls below $V_i(g_i)$, we abandon the i-level candidate and continue the search at level i.

ALGORITHM FOR SAMPLING SEARCH

First we calculate the value functions V_i for each level as in the waterfall case: for level 0, we derive a g value for each possible *object*-contingent distribution. Again the expected value of the search in this distribution is EC+V(g), and the overall profit of searching it is V(g). The value of having a 1-object with a true hidden score s_1 is $V(g_0(s_1))$, and thus the value of having an object with actual score s1 is $\int_{-\infty}^{\infty} OVD_{s_1}(x)V(g_0(x))dx$. To define the new value function $V_1(s_1)$ we can therefore convolve $V(g_0(x))$ with $OVD_s(x)$. We can now apply the same algorithm to level 1 to find a value function V_2, and so forth. We will again use a virtual o_{n+1} as an ancestor of the total level-n distribution to simplify the algorithm. We will use a vectors g[], h[], k[], p[], and d[] to hold the active values for each level.

Algorithm S (multi-level design with sampling)

[1] Set object $p[n]=o_{n+1}$ and distribution $d[n]=D_n$. Set $i=n$, and $bc_i=c_i$. Set g[n+1] to minus infinity.
[2] Generate an object from p[i]; call it x.
[3] Calculate a new d[i] using the method of section 7.
[4] Calculate g[i] from d[i], bc_i, and V_i. Calculate h[i] and k[i] using the method of section 5. (h=k=g if i=0.)
[5] Calculate a new bc_i using h and k in the method of section 6. Repeat [4] and [5] until the new g[i] is acceptably close to the previous one.
[6] If $V_i(g[i])<V_{i+1}(g[i+1])$, set $i=i+1$ and go to [2].
[7] If $s(x)<k[i]$ goto [2].
[8] If i is 0, we are done, and x is our result.
[9] Otherwise, set i=i-1, p[i] to x, d[i] to $SCD_i(s(x))$. Go to 2.

Algorithm S ameliorates the inability of Algorithm W to handle object-contingent variability. It should be noted, however, that it relies on the assumption that the OCDs are the same shape and differ only in their means.

IMPLEMENTATION AND EXPERIMENTS

Our present experiments deal with an abstraction hierarchy in VLSI layout. The following work concerns placement and routing. Our goal is to take a netlist and find a placement for the modules that is both compact in and of itself, and which allows for an efficient routing of the wires between their specified ports.

A common representation for placement at this level is a slicing structure, which corresponds to a partition tree. In addition to the partitions, the layout contains information regarding orientation and reflection of the subtrees at each node. We separate the representation into an "abstract placement", which consists of the partition tree and the orientation information, and the "concrete placement", which adds the reflection information to an abstract placement.

The size and shape of the overall layout is determined by the abstract placement; concrete placements for a given abstract one have a wide variation in wirability.

Since the area of the chip is a prime concern and is the major phenomenon of interest at level 1, total area was the primary level 1 evaluation function. At level 0 we are concerned with the nearness of modules that have connections between them.

CORRELATION BETWEEN LEVELS

After beginning with the two-level, iterated hillclimb model, it became clear that the correlation between the spaces was very important. The catch is that the information that determines the level 0 value simply isn't present at level 1; it hasn't been decided yet. This property extends to every level in an abstraction hierarchy and is thus an intrinsic part of the problem.

Our approach was first to invent a heuristic at level 1 that estimates the wireability at level 0 from the tree structure. We refer to this as the "channel heuristic" and it essentially measures the distance in the slicing structure tree of modules that are connected. (Each net corresponds to a specific subtree.) Secondly we adopted the overall requirement that lower-level values must dominate. This meant that full account of area must be taken by the level 0 evaluation that previously only counted wires. (Since the area is already known, this simply means adding it.)

Before the adoption of these techniques, the correlation between level 1 and 0 scores was statistically indistinguishable from 0. Afterward, it increased to about 0.4, which is adequate. It is a point of concern, and one not yet completely established, how the correlation coefficient varies with the particular circuit being designed. It is the most difficult of the statistics we use to estimate reliably on the fly. Even with a sample size of 1000 (10-module circuits) the value of 0.39 has a 95% confidence interval of 0.36 to 0.42. For 411 30-module circuits the value was 0.41 with a confidence interval that completely includes the other confidence interval. This is hardly proof, but is a heartening early indication that the correlation coefficient does not vary significantly with the specific circuit.

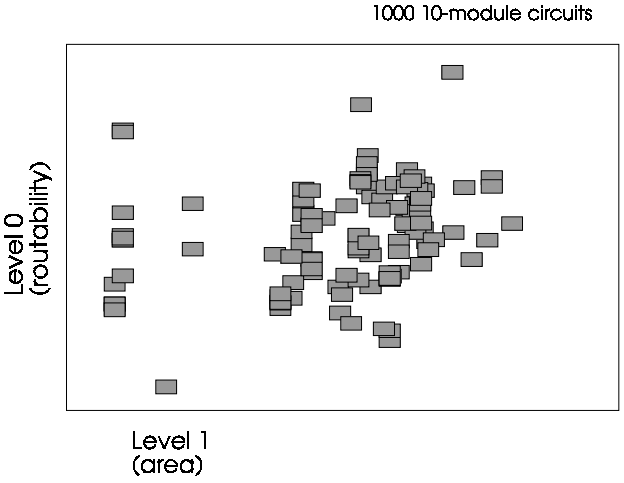

Figure 2. Bivariate distribution for level 0 and 1 heuristics for 1000 10-module circuits. Both axes are standard scores. The outliers to the left are caused by the sparsity of the level 1 space; they represent hilltops converged to by several separate random-start hillclimbs. This phenomenon disappears as circuit complexity increases.

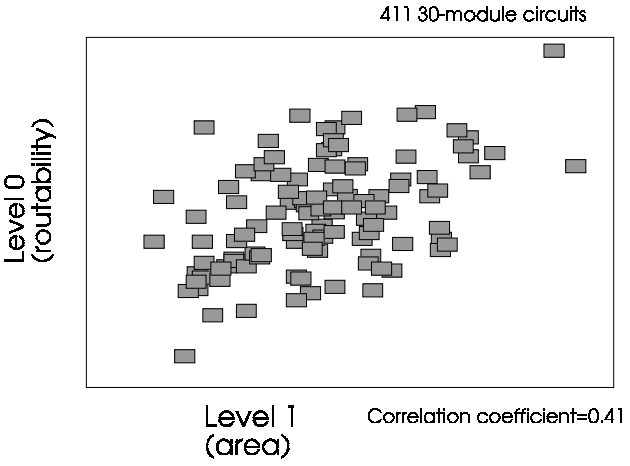

Figure 3. Distribution for 411 circuits with 30 modules. As circuit complexity increases, the distributions become more nearly normal, but the correlation coefficient does not appear to change significantly.

RELATED WORK

Russell and Wefald [91] is the prime reference in rational control. Their analysis of partially expanded nodes for MGSS2 is similar in many respects to the present one.

SUMMARY AND CONCLUSIONS

Multi-start hillclimbing can be analyzed as random search in the space of hilltops.

Rational random search in a population with a given distribution, probe cost, and value function has a lower bound g on its expected result that can be precalculated and the search run without dynamic cost/benefit computation.

The expected value of such a search is its expected cost plus the value of the score g.

Rational search in abstraction hierarchies for design is aided by knowledge of the correlation of the value distributions at adjacent levels. These correlations can be enhanced by appropriate heuristic evaluation functions and a formulation of the hierarchy such that lower level evaluations dominate higher level ones.

The statistical properties of the microprocessor design domain, at least in the placement and routing area, are well behaved enough for the approach to be useful. The properties improve as problem sizes approach those of commercial interest.

Rational control of search in design hierarchies can cause higher levels to modify their behavior based on information that arises from lower levels, ameliorating a major weakness of top-down design.

Dynamic rational control of multi-level design is likely to be worthwhile because of the high computational demands of the domain itself. This will be enhanced by efficient implementations of various integrals of density and value functions. Heuristic simplifications of multi-level rational control are possible, have been tested, are efficient, and work reasonably well.

REFERENCES

1. Hennessy, John, and Patterson, David: **Computer Architecture: A Quantitative Approach**, Morgan Kaufman, San Mateo CA, 1990
2. Russell, Stuart and Norvig, Peter: **Artificial Intelligence: A Modern Approach**, Prentice Hall, Englewood Cliffs NJ, 1995
3. Russell, Stuart and Wefald, Eric: **Do the Right Thing: Studies in Limited Rationality,** MIT Press, Cambridge MA, 1991
4. Sherwani, Naveed: **Algorithms for VLSI Physical Design Automation,** Kluwer Academic Publishers, Boston, 1995