01 Jan 1992

Composite Stock Cutting Through Simulated Annealing

Hanan Lutfiyya

Bruce M. McMillin
Missouri University of Science and Technology, ff@mst.edu

Pipatpong Poshyanonda

Cihan H. Dagli
Missouri University of Science and Technology, dagli@mst.edu

Recommended Citation

H. Lutfiyya et al., "Composite Stock Cutting Through Simulated Annealing," *Mathematical and Computer Modelling*, vol. 16, no. 1, pp. 57 - 74, Elsevier, Jan 1992.
The definitive version is available at https://doi.org/10.1016/0895-7177(92)90078-Y

Mathl. Comput. Modelling Vol. 16, No. 1, pp. 57–74, 1992
Printed in Great Britain.

0895-7177/92 $5.00 + 0.00

COMPOSITE STOCK CUTTING THROUGH SIMULATED ANNEALING

HANAN LUTFIYYA AND BRUCE MCMILLIN
Department of Computer Science, University of Missouri at Rolla
Rolla, Missouri 65401 U.S.A.
PIPATPONG POSHYANONDA AND CIHAN DAGLI
Department of Engineering Management, University of Missouri at Rolla
Rolla, Missouri 65401 U.S.A.

Abstract—This paper explores the use of Simulated Annealing as an optimization technique for the problem of Composite Material Stock Cutting. The shapes are not constrained to be convex polygons or even regular shapes. However, due to the composite nature of the material, the orientation of the shapes on the stock is restricted. For placements of various shapes, we show how to determine a cost function, annealing parameters and performance.

1. INTRODUCTION

For many important practical or theoretical problems, the objective is to choose a "best" solution out of a large number of candidate solutions, or the solution space. Such problems are typically known as combinatorial optimization problems. A combinatorial optimization problem is formalized as a pair (S, C), where S is the finite—or possibly countably infinite—set of configurations (also called configuration or search space) and C a cost function, $C : S \rightarrow R$, which assigns a real number to each configuration. For convenience, it is assumed that C is defined such that the lower the value of C, the better (with respect to the optimization criteria) the corresponding configuration. The problem now is to find a configuration for which C takes its minimum value, i.e., an (optimal) configuration i_0 satisfying

$$C_{\text{opt}} = C(i_0) = \min_{i \in S} C(i),$$

where C_{opt} denotes the optimum (minimum) cost.

One common combinatorial optimization problem that arises frequently in applications is the problem of allocating rectangular and/or irregular patterns onto a large stock sheet of finite dimensions in such a way that the resulting scrap will be minimized. This problem is common to many applications in aerospace, shipbuilding, VLSI design, steel construction, shoe manufacturing, clothing and furniture. This problem is commonly known as the stock cutting problem or the 2D bin packing problem.

The stock cutting problem is an example of a large scale optimization problem. It is unlikely to be solved by an algorithm whose run time is bounded by a polynomial function. This is an example of an NP-hard problem [1]. This means that this problem requires a computing effort that increases exponentially with the problem size. Since the stock cutting problem is of practical importance, efficient approximation algorithms, namely, algorithms that do not produce optimal but rather close-to-optimal solutions, have been developed. These include linear and

This work was supported in part by the National Science Foundation under Grant Numbers MIP-8909749 and CDA-8820714, in part by the AMOCO Faculty Development Program, in part by the Manufacturing Research and Training Center (MRTC), in part by the McDonnell Douglas Corporation, and in part by the University of Missouri Weldon Springs Fund.

We wish to acknowledge the programming efforts of Mr. Rob Zeigler and Mr. Chul-Eui Hong.

integer programming, dynamic programming, tree-search algorithms, and artificial intelligence and operations research (AI/OR) integration [2]. These proposed approaches have restrictions on the size and type of applications.

The paper is organized as follows: Section 2 discusses previous approaches to optimization of stock cutting, Section 3 presents Simulated Annealing through the concept of statistical mechanics. Section 4 describes the relevant parameters that control Simulated Annealing and Section 5 relates these parameters to the problem of stock cutting. Section 6 discusses the notion implementation of moving figures and Section 7 discusses the cooling schedule used. Experimental results are reported in Section 8.

2. PREVIOUS METHODS

Linear programming methods have been extensively researched. This work has been done by Gilmore and Gomory [3–5], Geoffrion and Marsten [6], Haessker [7] and Dyckhoff [8]. In general this method involves a solution of the problem through development of mathematical models consisting of an objective function that is to be minimized or maximized, and constraint functions indicating the limitations on the allowed values of the variables of the objective function. Both the objective and constraint functions are linear functions of independent variables. Any model may be transferred to the following standard form:

$$\begin{aligned}&\text{minimize } c_1x_1 + c_2x_2 + \cdots + c_nx_n,\\ &\text{subject to}\\ &\qquad a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1,\\ &\qquad a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2,\\ &\qquad a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m.\end{aligned}$$

The stock cutting problem can result in a mathematical model consisting of hundreds, and even thousands, of variables and constraints. The above mentioned work has concentrated on finding special structural characteristics of the model and developing techniques that exploit these structures. These methods have successfully been applied to a broad class of stock cutting problems. There are, however, many real problems for which these methods are not appropriate due to their structure or size. In many cases, this is caused by special restrictions. For such problems, other methods—often heuristic ones—are used.

Dynamic programming is one heuristic often used. Dynamic programming is an algorithm design method that takes a model of the problem and converts it into a series of single stage problems. This transformation is intuitively based on the principle that an optimal set of decisions has the property that, whatever the first decision is, the remaining decisions must be optimal with respect to the outcome which results from the first decision. The difficulty is in determining quickly the optimal decisions. Otherwise, the problem degrades into enumeration of the decisions and determining which is the best. This has exponential complexity. Studies of dynamic programming approaches to the stock cutting problem have been done by Beasley [9,10] and Sarker [11].

Another class of heuristics often used are tree-search methods. This method enumerates all possible solutions in a tree like organization. Many different tree organizations may exist for the solution space. Heuristics exist for finding the solution to the problem by traversing the tree. These heuristics will start out on one path and will terminate when either an optimal solution is believed to have been found, or the path is known to result in an unsatisfactory solution. It is difficult to determine which path to start on and, once on a particular path, determining whether the path is worth traversing, i.e., lower costs possible, or whether to proceed on a different path. This work has been done by Christofides and Whitlock [12], Hinxman [13] and Beasley [9,10].

Another heuristic used is the iterative improvement method. Application of the method of iterative improvement requires the definition of a solution space, a cost function and a set of moves that can be used to modify a solution. Define a solution $\mathbf{x_i} = (x_{i0}, x_{i1}, \ldots, x_{im})$ on m variables. A solution $\mathbf{y}$ is a neighbor (neighboring solution) of a solution $\mathbf{x}$ if $\mathbf{y}$ can be obtained from $\mathbf{x}$ via one of the moves. In this method, one starts with an initial solution represented

by $\mathbf{x_0} = (x_{00}, x_{01}, \ldots, x_{0m})$. At iteration i, if the current solution is $\mathbf{x_i}$, then its neighbors are examined until a neighboring solution $\mathbf{x_{i+1}}$ is found with a new cost. In that case, $\mathbf{x_{i+1}}$ is the new solution and the process is continued to examine the neighbors of the new solution. The algorithm terminates when it arrives at a solution which has no neighboring solution with a lower cost. This process tends to minimize the cost but can get trapped in a poor solution, i.e., it may be a local minimum, but not global. Figure 1 shows how this may happen. If $\mathbf{x_0}$ is the chosen initial configuration, then the iterative improvement method will choose configuration A as the optimum. If $\mathbf{x'_0}$ is the chosen initial configuration, then the iterative improvement method will choose configuration B as the optimum. Solution A is a local optimum. All neighboring solutions have a higher cost than A, hence iterative improvement is trapped at A. However, B is the configuration that minimizes the cost. It is the global solution. Thus iterative improvement is sensitive to the choice of the initial configuration.

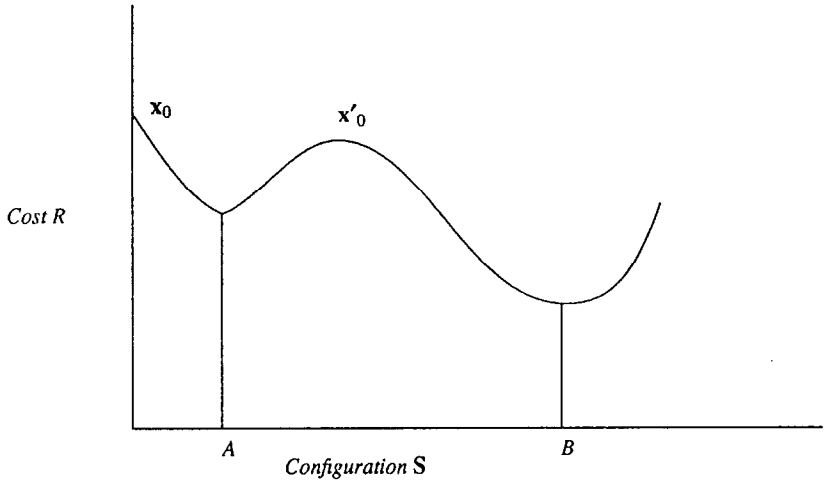

Figure 1. Local minimum problem. The initial guess is important in iterative improvement. A poor initial guess $\mathbf{x_0}$ leads to a local minimum and a sub-optimal solution, A. A good initial guess, $\mathbf{x'_0}$ leads to the optimal solution, B.

Because most of the above approaches either do not give an optimal or near optimal solution, or are not applicable to a wide variety of applications, a new approach using simulated annealing is considered. Simulated annealing can be used to give near optimal solutions and be used for all applications.

3. SIMULATED ANNEALING

3.1. Motivation

Simulated annealing is a method of optimization that tries to avoid the pitfalls inherent in other optimization methods, such as the iterative improvement approach; i.e., it seeks the global or near global minimum of a function without getting trapped in a local minimum. Simulated annealing is designed to optimize functions of several hundred variables or more, and is especially attractive when the functions are not smooth, i.e., have many local minima. Simulated annealing has been used in solving circuit routing, image processing and the traveling salesman problem [14–19]. Possible use of simulated annealing for solving the stock cutting problem is proposed by Dagli [20] as a promising tool based on a small problem.

3.2. Statistical Mechanics

At the heart of the method of simulated annealing is an analogy in statistical mechanics. Statistical mechanics is the central discipline of condensed matter physics, a body of methods for analyzing aggregate properties of the large numbers of atoms, to be found in samples of liquid or solid matter. Because the number of atoms is of order 10^{23} per cubic centimeter, only the most probable behavior of the system at a given temperature is observed in experiments. Since at a

given temperature, the behavior may vary, determining the most probable behavior is done by considering the average behavior of a collection of identical systems. This is called an ensemble of systems. In this ensemble, each configuration, defined by the set of atomic positions, $\mathbf{r} = (r_i)$, of the system is weighted by its Boltzmann probability factor,

$$e^{-E(r_i)/k_B T},$$

where $E(\mathbf{r})$ is the energy of the configuration, k_B is Boltzmann's constant and T is temperature.

A fundamental question in statistical mechanics concerns what happens to the system in the limit of low temperature T. For example, whether the particles (could be referring to atoms or molecules) remain fluid or solidify, and if they solidify, whether they form a crystalline solid or a glass. Ground states, i.e., states with lowest energy, and configurations close to them in energy are extremely rare among all configurations, yet they dominate at low temperatures because as T is lowered the Boltzmann distribution collapses into the lowest energy state or states.

Consider a two-dimensional network of points arranged in a square lattice with each point, labeled by its integer coordinates i, j, connected to its four nearest neighbors, where two points i_1, j_1 and i_2, j_2 are nearest neighbors if

$$|i_1 - i_2| \leq 1 \text{ and } |j_1 - j_2| \leq 1.$$

Suppose we place in each point a particle with a magnetic moment (spin), and that each particle can be in one of two states, conventionally labeled

$$r = -1 \text{ and } r = +1,$$

or called "spin down" or "spin up," respectively. Suppose that each particle interacts with its nearest neighbors. Then the energy of the system can be written as

$$E = \frac{-J}{2} \sum_{i_1 j_1 i_2 j_2} C_{i_1 j_1 i_2 j_2} r_{i_1 j_1} r_{i_2 j_2},$$

where $C_{i_1 j_1 i_2 j_2}$ is a connection matrix such that

$$C_{i_1 j_1 i_2 j_2} = \begin{cases} 1, & \text{if } |i_1 - i_2| \leq \text{ and } |j_1 - j_2| \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$

This is a very simplified but physically reasonable model, which was introduced by Ising [21], for a two-dimensional substance exhibiting ferromagnetic behavior. J (a positive constant) is the contribution to the total energy given by a pair of adjacent spins, the sign of the contribution being negative if the two adjacent spins are aligned (i.e., both up or both down) and positive otherwise. The system, in order to minimize its energy, tends to align all its spins in the same direction. Therefore, the ground state configuration (state with minimal energy) is very simple, with all the spins pointing in one direction. In practice, the magnetization is rarely perfect, i.e., the spins are slightly shifted towards the interior of the $(-1, 1)$ segment i.e.,

$$r_i = 1 - \varepsilon_1 \text{ or } r_i = -1 + \varepsilon_2,$$

where $0 < \varepsilon_{1,2} \ll 1$. This is corrected by replacing

$$r_i \longrightarrow \text{ sign } r_i.$$

The system is crystallized when E is minimal. The Boltzmann's distribution implies that E is minimal at a low temperature.

In practical contexts, low temperature is not a sufficient condition for finding ground states of systems. Experiments that determine the low temperature state of a system that gives us minimal energy are done by a method called annealing. First, the system is raised to a high temperature,

then the temperature is lowered slowly and the system spends a long time at temperatures in the vicinity of the freezing point. Enough time must be spent at each temperature for each of the spins to reach a steady state. In other words, each spin must be given enough time to polarize towards 1 or -1. This is called thermal equilibrium or quasi equilibrium. If the temperature is lowered too quickly, the system does not have sufficient time to achieve equilibrium and the resulting state might have high energy.

As mentioned before, studies have shown that the distribution of atoms in a system, at a temperature T, satisfies the Boltzmann's distribution. This probability expresses the idea that a system in thermal equilibrium at temperature T, has its energy probabilistically distributed among all different energy states. Even at low temperatures, there is a chance, albeit very small, of a system being in a high energy state. Therefore, there is a corresponding chance of a system to get out of a local energy minimum in favor of finding a more global minimum.

The spin variables are discrete and updated asynchronously, i.e., spin variables do not polarize all at once. The annealing process of the system described by Ising can then be represented formally as follows:

$$r_i(t+\varepsilon) = \text{sign}\,(r_i(t)\,\Delta_i(t))\,(1 - e^{-\Delta_i \frac{(t)}{T}}) - r_i(t)\,e^{-\Delta_i \frac{(t)}{T}},$$

where $r_i(t)$, $r_i(t+\varepsilon)$ represent the value of the spin variable r_i at time t and $t+\varepsilon$, respectively, and $\Delta_i = r_i(t+\varepsilon) - r_i(t)$.

For large temperatures, the second term of the above equation dominates. However, as the temperature decreases, the first term will dominate.

3.3. Metropolis Algorithm

It is sometimes difficult to work with ensembles of physical systems. Metropolis [22] introduced a simple algorithm to simulate the annealing process. For a given temperature T, the Metropolis method is a way to sample states of the physical system with the Boltzmann distribution. A state $\mathbf{x}_0$ is randomly chosen. At each iteration, the algorithm then chooses (again randomly) a small perturbation Δx_i in the system and calculates the energy change ΔE caused by the perturbation

$$\Delta E = E(\mathbf{x}_i + \Delta x_i) - E(\mathbf{x}_i).$$

If $\Delta E < 0$, then the perturbation is "accepted," for it means that is energetically favorable for the system; otherwise, it is accepted with probability

$$e^{-\Delta E/k_B T}.$$

When the perturbation is accepted, one continues the process with the perturbed state $\mathbf{x}_i + \Delta x_i$ replacing the old one; otherwise, a new perturbation Δx_i is attempted. This process is continued until thermal equilibrium is reached, i.e., until the probability distribution of the configurations approaches the Boltzmann distribution. It can be shown that the sequence of states obtained in this way is distributed according to the Boltzmann's distribution. Note that the $\mathbf{x}_i$'s are the encoding of the spin variables. The Metropolis method is widely used in physics to study numerically the thermodynamical properties of large systems than cannot be treated with analytical methods. The procedure is shown in Figure 2.

The Metropolis method can be used to simulate the annealing process of the substance given in the previous section, the Ising model, as follows. Let $\mathbf{r}_0$ represent the values of all spin variables. At each iteration, a spin variable r_i is chosen to have its value changed to $-r_i$. This is the small perturbation of the system. The new energy is calculated according to Ising's equation for energy. ΔE is computed. If $\Delta E \leq 0$, then this new state is accepted. The Boltzmann distribution implies that it is possible for a system to be at a higher energy state. This is simulated by the Metropolis procedure by computing a probability of accepting the new state. Thermal equilibrium is reached as the new states generated approach the Boltzmann distribution.

```
Metropolis' Procedure
  Begin
    Choose some random initial configuration S;
    Repeat
      S' := Some random neighboring configuration;
      Δ := E(S') − E(S);
        Prob := min(1, e^{−Δ/k_B T});
        if random (0, 1) ≤ Prob  then  S := S';
    until false;
  End
```

Figure 2. The Metropolis Procedure probabilistically samples states of the configuration.

Physical systems	Optimization problems
State	Configuration
Energy	Cost function
Ground state	Optimal solution
Quick cooling	Iterative improvement
Careful annealing	Simulated annealing

Figure 3. Analogy between physical systems and optimization problems.

3.4. Simulated Annealing Technique

Simulated annealing was proposed independently by Kirkpatrick [23] and by Cerny [24], as a method for minimizing functions of many variables. The idea was derived from the algorithm by Metropolis [22]. There is an analogy between a combinatorial optimization problem and the problem of determining the lowest-energy ground state of a physical system. This analogy is summarized in Figure 3.

The states of a system correspond exactly to the configuration of the combinatorial optimization problem. The ground state corresponds to the optimal configuration, i.e., a configuration that minimize the cost function. The finding of the optimal configuration corresponds to determining the ground state. This process is called simulated annealing. The temperature becomes a control parameter.

With simulated annealing, a high value of T is started with, so that the probability of the system being in a given state is independent of the cost of that state. T is slowly reduced, by making sure that at each new value of T enough steps of the Metropolis procedure are made to guarantee that thermal equilibrium has been reached. The procedure is continued until $T = 0$. If the cooling has been slow enough, the final state reached is the ground state of the combinatorial optimization problem being considered; i.e., the values of $\mathbf{x}_i$ so obtained realize the absolute minimum of the cost function. In practice, in many cases one is not really interested in finding the absolute minimum. Rather, in many interesting situations the minimum configuration is highly degenerate. In other words, there are many minima with values very close to the absolute minimum value. A general simulated annealing algorithm is given in Figure 4.

Initially, the control parameter, T, is given a high value and the system is assumed to start at some state S. To simulate reaching thermal equilibrium at a temperature T, the generic simulated annealing algorithm generates a sequence of configurations of the cost function. A generation mechanism is defined, so that given a configuration S, another configuration S' can be obtained randomly from the neighborhood of S. Δ is defined as the difference in the costs of each configuration, i.e., $\Delta = C(S') - C(S)$. The probability for a state S' to be the next configuration is determined as follows:

$$\text{Prob (new state } = S') = \begin{cases} 1, & \text{if } \Delta \leq 0, \\ e^{-\Delta/T}, & \text{if } \Delta \geq 0. \end{cases}$$

```
Generic Simulated Annealing Algorithm
  begin
    S := Initial solution S_0;
    T := Initial temperature T_0
    while (stopping criterion is not satisfied) do
      begin
        while (not yet in equilibrium) do
          begin
            S' := Some random neighbor of S;
            Δ := C(S') − C(S);
              Prob := min(1, e^{−Δ/T});
              if random (0,1) ≤ Prob then  S := S';
            end;
          Update T;
        end;
      Output best solution;
    end;
```

Figure 4. The simulated annealing algorithm makes probabilistic decisions based on a control parameter T, called temperature.

The acceptance criterion is implemented by drawing random numbers from a uniform distribution on $[0, 1)$. Thus, there is non-zero probability of continuing with a configuration with higher cost than the current configuration. This is continued until equilibrium is reached. The control parameter, T, is then lowered in steps, with the system being allowed to approach equilibrium for each step. The algorithm is terminated for some small value of T. The final configuration is taken as the solution of the problem at hand.

Determination of the initial temperature T, the decrement factor of T at each step (i.e., how T is updated) and the determining of thermal equilibrium is called the cooling schedule. Convergence with various cooling schedules has been proven [25]. Convergence at each temperature, or control parameter, means that the inner loop has reached thermal equilibrium. The convergence of the entire algorithm means that an optimum configuration, or cost value, has been reached. An exponentially long time may be required for convergence, depending on the size of the problem. Various heuristics, in addition to the physical analogy, must therefore be used in determining the cooling schedule.

4. SIMULATED ANNEALING PARAMETERS

The crucial parts of the algorithm are the definition of "move," or the way in which the configuration is perturbed and the cooling schedule.

4.1. Moves

An important parameter in simulated annealing is the definition of "move," or the way a configuration is perturbed to get a new configuration. Most researchers agree that at high temperatures, large moves should be made, but as the temperature is lowered, smaller moves should be made. A large move is defined as a move that causes a large difference in the value of the cost function. A small move is defined as a move that causes a small difference. Large moves at high temperatures allows the algorithm to avoid local minima.

4.2. Cooling Schedule

A summary of some effective cooling schedules is presented in [25]. The initial value of the control parameter T should be large enough to allow the algorithm to accept any new configurations regardless of the change in cost. Let the notation T_i refer to the temperature at the i^{th} iteration of the outer loop. The updating of the temperature usually takes the form of the decrement rule $T_{k+1} = f(T_k)$. The decrement rule controls the outer loop and should cool the system slowly enough to provide an accurate solution without running an excessively long time.

The value of the final temperature determines when the system should halt processing. This is usually when there is little improvement in the optimization. The quantity *little* must be specified by the user and must be determined empirically such that solutions are accurate and terminate in a reasonable amount of time.

The time required for the inner loop to reach quasi-equilibrium is another parameter of the cooling schedule. There is a trade off between time and accuracy. The theory [25] says that for convergence, every state must be visited infinitely often. Obviously, in practical uses, this is not feasible. The length of the inner loop should be long enough to provide a good solution, but not so long as to provide little extra information for the time spent. Therefore, some heuristic must be used to determine inner loop iterations.

One common cooling schedule is described by van Laarhoven and Aarts [25]. Let the notation $\Delta\overline{C}_{ij}$ refer to the difference in the cost of the system when the system goes from configuration i to configuration j.

DEFINITION 4.1 (ACCEPTANCE RATIO). *The acceptance ratio χ is the ratio between the number of configurations accepted by the algorithm and the total number of configurations generated by the algorithm for a given temperature.*

As pointed out before, the value of T_0 should be high enough such that virtually all generated configurations are accepted. This corresponds to χ close to one. The value of T_0 can be obtained by monitoring the evolution of the system during a number of runs of the algorithm before the actual optimization process starts, and adjusting the cooling parameter in such a way that a constant value of the acceptance ratio is maintained. The value of T_0 is then given by the final value T obtained by updating T_0 m times according to the expression:

$$T = \Delta\overline{C}^{(-)} \left(\ln \frac{m_2}{m_2 \chi - (1-\chi) m_1} \right)^{-1},$$

where m_1 and m_2 are the number of generated configurations with $\Delta\overline{C}_{ij} \leq 0$ and $\Delta\overline{C}_{ij} > 0$ ($m_1 + m_2 = m_0$), respectively, and $\Delta\overline{C}^{(-)}$ the average value of those $\Delta\overline{C}_{ij}$ for which $\Delta\overline{C}_{ij} > 0$.

Determination of the start value of the cooling parameter in the way described above can only be done reliably in those cases where the values of the cost function for different configurations are sufficiently uniformly distributed. Otherwise, the above will result in a value of T_0 which is too small and, therefore, will cause the algorithm to get stuck in a local optimum. In this case one is forced to determine T_0 differently.

The notation $C_i(T_k)$ refers to the cost of the configuration when the temperature or control parameter is T_k. The term $\overline{C}(T_k)$ approximates the statistical expectation of cost at temperature T_k, and is the average cost over n accepted moves, achieved after the inner loop has reached equilibrium, i.e.,

$$\overline{C}(T_k) = \frac{1}{n} \sum_{i=1}^{n} C_i(T_k).$$

Similarly, $\overline{C}^2(T_k)$ approximates the second moment of cost

$$\overline{C}^2(T_k) = \frac{1}{n} \sum_{i=1}^{n} C_i^2(T_k).$$

The sample variance of cost is defined as

$$\sigma^2(T_k) = \overline{C}^2(T_k) - [\overline{C}(T_k)]^2.$$

The decrement rule f is established by the following equation:

$$T_{k+1} = T_k \left[1 + \frac{\ln(1+\delta) T_k}{3\sigma(T_k)} \right]^{-1}.$$

The smaller σ, the slower the system cools. Also note the dependence on $\sigma(T_k)$; the larger the variance of costs for the inner loop, the slower the system cools. The final value of the control parameter, T_f, is taken to be the first value of the control parameter, that satisfies

$$\frac{\sigma^2(T_k)}{T_f(\overline{C}(T_0) - \overline{C}(T_f))} < \varepsilon,$$

for some small value of ε. The smaller ε, the longer the algorithm runs.

Theoretically, the thermal equilibrium is reached after each configuration is visited infinitely often. However, for practical purposes an approximation of when the algorithm has reached thermal equilibrium must be used. However, there is no practical, theoretical method of determining thermal equilibrium. In this paper, the number of iterations of the inner loop is dependent on the number of moves that were accepted. Hence, as an annealing run approaches a solution, the longer the inner loop will run.

5. DISCUSSION OF THE COST FUNCTION

The composite, or oriented 2D bin packing, stock cutting problem is the problem of cutting patterns from a stock sheet such that the scrap area is minimized, and stock orientation is important. It is important to note that most applications do not allow the patterns to be rotated any random angle. In other words, patterns may rotate for a limited number of rotation angles. Various cost functions based on different parameters can be used. Research has led us to believe that a cost function based on a weighted sum of distances between patterns within the configuration represents a computationally feasible function that allows us to minimize the scrap area.

DEFINITION 5.1 (SHAPE SET). *Let the run of shapes to be cut be* $\mathbf{s} = (s_i)$, *for* $i = 1, \ldots, S$. $\mathbf{s}$ *is the shape set.*

Let $\mathbf{h}_{s_i} = (h_{s_ijx}, h_{s_ijy})$ where $j = 1, \ldots, H_i$. (h_{s_ijx}, h_{s_ijy}) is the one end of a line segment and $(h_{s_i(j+1)x}, h_{s_i(j+1)y})$ is the other end of a line segment of shape s_i. We will denote this line segment by $e_{s_ij,j+1}$.

DEFINITION 5.3 (ROTATIONAL ANGLE). θ_k *is the* k^{th} *rotational angle of an object.*

DEFINITION 5.4 (OBJECT). *For each* s_i *and* θ_k, *let*

$$[h_{r_{ik}jx} h_{r_{ik}jy}] = [h_{s_ijx} h_{s_ijy}] \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix},$$

where $\mathbf{r}_{ik}$ *is the* k^{th} *rotation of an object* $\mathbf{s}_i$.

An object is $\mathbf{h}_{s_i}$ for some shape s_i of the shape set $\mathbf{s}$, or is the k^{th} rotation of shape s_i.

5.1. Finding Minimal Area

The cost function used is a weighted sum of distances between patterns within the configuration. An index, which we call the *affinity relation* and is denoted by a_{ij}, represents the tendency of object i to attract object j. The affinity relation between pattern i and j is the weight associated with the distance between objects i and j.

DEFINITION 5.5 (EDGEWISE ADJACENT). *Define* δ_{ijklmn} *between each pair of objects* i, l, *for each* j^{th} *and* m^{th} *rotational angles of objects* i *and* l, *respectively, and for each pair of line segments* $e_{r_ijk,k+1}$ *and* $e_{r_lmn,n+1}$, *such that*

$$\delta_{ijklmn} = \begin{cases} 1, & \text{if } \left[\frac{h_{r_ijky} - h_{r_ij(k+1)y}}{h_{r_ijkx} - h_{r_ij(k+1)x}} - \frac{h_{r_lmny} - h_{r_lm(n+1)y}}{h_{r_lmnx} - h_{r_lm(n+1)x}}\right] \leq \varepsilon, \\ 0, & \text{otherwise.} \end{cases}$$

δ_{ijklmn} *is one if the slopes of the line segments* $e_{r_ijk,k+1}$ *and* $e_{r_lmn,n+1}$ *differ no more than* ε, *where* ε *is a very small number.* δ_{ijklmn} *having a value of one implies that the* j^{th} *rotation of*

pattern i and the m^{th} rotation of the pattern l may be placed adjacent to each other by placing the line segments $e_{r_{ijk},k+1}$ and $e_{r_{lmn},n+1}$ adjacent to each other. The corresponding objects i and j are said to be edgewise adjacent.

EXAMPLE 5.1. In Figure 5, assume that both objects are at their 0^{th} rotational angle and no other rotational angle is permissible,

$$\mathbf{h}_1 = (0,8),(0,0),(4,0),$$
$$\mathbf{h}_2 = (13,13),(12,10),(10,14).$$

The slopes of edges are computed as follows:

$$e_{1,12} : \frac{8-0}{0-0} \to \infty, \quad e_{1,23} : \frac{0-0}{0-4} = 0, \quad e_{1,31} : \frac{8-0}{0-4} = -2,$$
$$e_{2,12} : \frac{13-10}{13-12} = 3, \quad e_{2,23} : \frac{10-14}{12-10} = -2, \quad e_{2,31} : \frac{14-13}{10-13} = -\frac{1}{3}.$$

Edges $e_{1,31}$ and $e_{2,23}$ have equivalent slopes. Hence, the two objects may be placed adjacent to each other by placing edges $e_{1,31}$ and $e_{2,23}$ adjacent to each other. Figure 6 shows this.

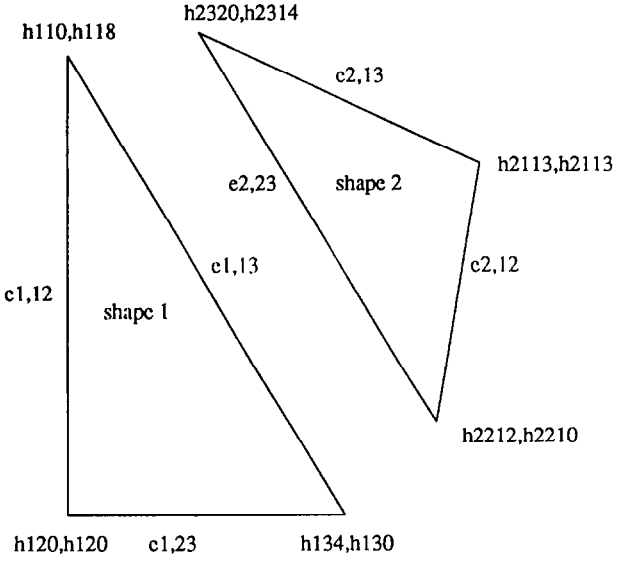

Figure 5. Shape representation.

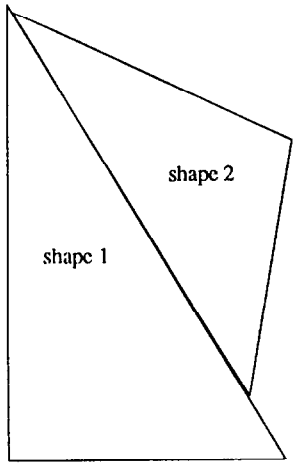

Figure 6. Maximize edgewise adjacency.

DEFINITION 5.6 (LENGTH OF EDGEWISE ADJACENCY).

$$a_{ijklmn} = \min\left(\left[(h_{r_{ijkx}} - h_{r_{ij(k+1)x}})^2 + (h_{r_{ijky}} - h_{r_{ij(k+1)y}})^2\right]\delta_{ijklmn},\right.$$
$$\left.\left[(h_{r_{lmnx}} - h_{r_{lm(n+1)x}})^2 + (h_{r_{lmny}} - h_{r_{lm(n+1)y}})^2\right]\delta_{ijklmn}\right).$$

DEFINITION 5.7. *The affinity relation for object i and l is the following:*

$$a_{il} = \max(a_{ijklmn}).$$

THEOREM 5.1. *For each pair of objects i, j, the affinity relation represents the positioning of the object that maximizes the length of edgewise adjacency.*

PROOF. a_{ijklmn} is the length of the edgewise adjacency between the line segments $e_{r_{ijk},k+1}$ and $e_{r_{lmn},n+1}$. Note that if δ_{ijklmn} is zero, then a_{ijklmn} is also zero. This indicates that there is no edgewise adjacency between the line segments $e_{r_{ijk},k+1}$ and $e_{r_{lmn},n+1}$. If, on the other hand, δ_{ijklmn} is equal to one, then a_{ijklmn} is equal to the length of the shorter segment. ∎

5.2. Cost Function

The cost function is the following:

$$-\left(\alpha \sum_{ij} \frac{1}{d_{ij}} a_{ij} + \beta \sum_{i_j} \frac{1}{d_{ij}}\right),$$

where α, β are positive real numbers that indicate the contribution of each of the components in the cost function. Choosing α too small will result in configurations positioned close to the origin, but far from an optimal placement. The cost function for the simulated algorithm consists of two independent terms.

Table 1.

distance \ affinity relation	L	H
L	H	L
H	HH	H

Table 2.

distance \ affinity relation	L	H
L	L	H
H	LL	L

5.2.1. First Term of Cost Function

Consider the first term of the cost function is the following:

$$\sum_{ij} d_{ij} a_{ij}.$$

This term is minimized as the higher the affinity relation between two objects the smaller the distance between the two objects. Since the affinity relation represents the positioning of the corresponding objects such that the box surrounding the two shapes is small, then the higher the affinity relation, the greater possibility of saving scrap area. Hence, minimizing the distance between two objects that have a high affinity relation should bring the objects closer the position that maximizes their edgewise adjacency. Table 1 illustrates the status of the cost relative to the status of the affinity relation and distance values.

There is one problem associated with this function. If the adjacency between two patterns is zero, then the cost associated with those two patterns is 0, irrespective of the distance between them. Our solution to this problem was to maximize this term:

$$\sum_{ij} \frac{1}{d_{ij}}, a_{ij},$$

or equivalently, minimize this term

$$-\sum_{ij} \frac{1}{d_{ij}} a_{ij}.$$

Table 2 illustrates the status of the cost relative to the status of the affinity relation and distance values. Note that the cost associated with two patterns is maximized when their adjacency is high and distance between the patterns is low. If the adjacency is 0, it doesn't affect the result.

It is important to note that the cost function used in this paper does not always find the optimal solution. The affinity relation represents the positioning of the corresponding objects, such that the box surrounding the two shapes is small. However, this box is not minimal.

THEOREM 5.2. *The affinity relation does not guarantee that the positioning of the corresponding objects is such that the bounding box is minimal.*

PROOF. See Figures 8, 9 and 10. It is easy to compute the affinity relation for shapes 1 and 2. Edgewise adjacency is maximized in both Figures 9 and 10, however the bounding box in Figure 9 is larger than the bounding box in Figure 10. ∎

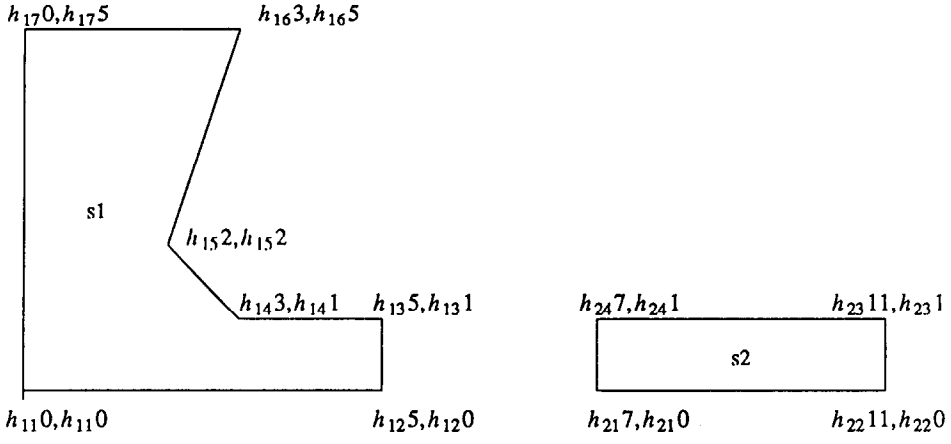

Figure 8. Candidate figures are placed to maximize edgewise adjacency and minimize the bounding box.

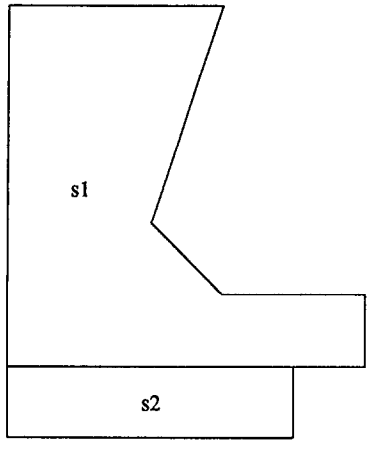

Figure 9. Maximize edgewise adjacency.

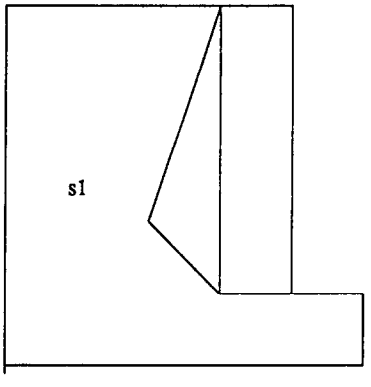

Figure 10. Minimal bounding box.

5.2.2. Second Term of the Cost Function

The cost function will also try to minimize the following:

$$-\sum_i \frac{1}{d_{i_0}},$$

where d_{i_0} represents the distance of pattern i from the origin. This forces the clustering of objects into a small bounding box.

5.3. Overlap

One component of the the cost function should include a penalty function on the area of overlapping patterns. It is easy to formulate this component if the objects are rectangular in nature [26]. However, we would like to be able to work with more irregular patterns. Formulating algebraic constraints that prevent the overlapping is very difficult.

Our method of handling overlaps uses a three dimensional grid of indicators where the output of indicator V_{ijk} is as follows:

$$V_{ijk} = \begin{cases} 1, & \text{if object } i \text{ occupies grid } jk, \\ 0, & \text{otherwise.} \end{cases}$$

$V_{ijk} = 1$ implies that object i occupies grid point jk.

DEFINITION 5.8 (OVERLAPPING). *A configuration overlaps when there exists the following situation:*

$$\sum_i V_{ijk} > 1,$$

for a fixed j and fixed k.

6. MOVES

6.1. Types of Moves

Three types of moves are allowed in order to be able to change the arrangement of the given object. These three types of moves, Φ, are the following:

Φ_1: Displace an object to a new location.
Φ_2: Interchange two objects.
Φ_3: Change the orientation of an object.

6.1.1. Description of Φ_1

An object i, where $\mathbf{h}_{s_i} = (h_{s_i jx}, h_{s_i jy})$, $i = 1, 2, \ldots, H_i$, is the set of vertices, is randomly selected for displacement to a randomly selected new location. The new location is determined by randomly selecting a displacement in the horizontal direction, represented by x_disp, and by randomly selecting a displacement in the vertical direction, represented by y_disp. The new set of vertices can be computed as follows:

$$\mathbf{h}'_{s_i} = (h_{s_i jx} + x\_\text{disp}, h_{s_i jy} + y\_\text{disp}).$$

Large-distance moves usually imply large values of ΔC. At low temperatures, only moves which approximately satisfy $\Delta C \leq 0$ have a reasonable chance of being accepted. Hence, at low temperatures, the large distance moves are almost invariably rejected. In order to generate moves which have a reasonable probability of acceptance, these large distance moves are prohibited by the use of a range-limiter window [27]. When an object is selected for displacement, such as shape 1 in Figure 5, the range-limiter window is centered at (x_0, y_0), corresponding to the center of object i. The randomly selected new location for object i must lie within the range-limiter window. At the beginning of the annealing process, the window size is set to be large enough to contain all the objects, and it shrinks slowly as the temperature decreases. In fact, the height and width of the window are proportional to the logarithm of the temperature:

$$x\_\text{disp} = x\_\text{dim} \left\{ \frac{\rho^{\log_{10}(T)}}{\lambda} \right\},$$

$$y\_\text{disp} = y\_\text{dim} \left\{ \frac{\rho^{\log_{10}(T)}}{\lambda} \right\}.$$

x_dim represents the window span in the x-direction and y_dim represents the window span in the y-direction at the initial temperature (T_0). The value of λ was chosen such that for the initial temperature, the term in the braces in the right hand side of the above equations is normalized to one. That is,

$$\lambda = \rho^{\log_{10} T_0}.$$

6.1.2. Description of Φ_2

Φ_2 will cause an interchange of objects i and k, where

$$\begin{aligned} \mathbf{h}_i &= (h_{s_i jx}, h_{s_i jy}), \qquad j = 1, 2, \ldots, H_i, \\ \mathbf{h}_k &= (h_{s_k jx}, h_{s_k jy}), \qquad j = 1, 2, \ldots, H_k. \end{aligned}$$

Let $c_i = (x_i, y_i)$ and $c_k = (x_k, y_k)$ denote the centers of objects i and k, respectively. Compute the following:

$$x_i\_\text{disp} = \begin{cases} x_i - x_k, \text{ if } x_i > x_k, \\ x_k - x_i, \text{otherwise}, \end{cases} \qquad x_k\_\text{disp} = \begin{cases} x_k - x_i, \text{ if } x_k > x_i, \\ x_i - x_k, \text{otherwise}, \end{cases}$$

$$y_i\_\text{disp} = \begin{cases} y_i - y_k, \text{if } y_i > y_k, \\ y_k - y_i, \text{otherwise}, \end{cases} \qquad y_k\_\text{disp} = \begin{cases} y_k - y_i, \text{if } y_k > y_i, \\ y_i - y_k, \text{otherwise}. \end{cases}$$

The displacements can be added to the chosen objects in a similar manner to the displacements displayed in Φ_1.

6.1.3. Description of Φ_3

An object i is randomly selected to have its orientation changed. A rotational angle θ_k is randomly chosen. $h_{r_{ik}}$ is the set of vertices after object i has been rotated. It is computed as follows:

$$[h_{r_{ik}jx}, h_{r_{ik}jy}] = [h_{s_i jx}, h_{s_i jy}] \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix},$$

where $\mathbf{r}_{ik}$ is the k^{th} rotation of object $\mathbf{s}_i$. Hence,

$$h_{s_i} = h_{r_{ik}}.$$

6.2. Selection of Moves

Moves are selected by randomly picking an integer number on the interval [1,3], where 1 corresponds to choosing Φ_1, 2 corresponds to choosing Φ_2 and 3 corresponds to choosing Φ_3. We experimented with several discrete probability distribution functions.

We first experimented with a discrete uniform distribution on the integers 1,2,3, where the probability distribution function $f(x)$ is the following:

$$f(x) = \frac{1}{3}, \quad \text{where } x = 1, 2, 3.$$

In subsequent experiments the probability distribution function was the following:

$$f(x) = \begin{cases} p, & \text{if } x = 1, \\ q, & \text{if } x = 2, \\ 1 - p - q, & \text{if } x = 3, \end{cases}$$

where $p + 1 < 1$ and $p, q > 0$. The choice of p and q can have an important effect on the final solution.

Objects of moves, the displacements and rotational angles were chosen from a discrete uniform distribution.

7. COOLING SCHEDULE

Using the method described in Section 4 for determining the initial value of the control parameter T_0, resulted in a value of T_0 that was too small. This result made it difficult to determine the decrement value and the stopping criteria described in Section 4. Hence, the following heuristic was used in determining values for the simulated annealing parameters.

A fixed number of runs of the algorithm, denoted by N, was assumed. The decrement function was assumed to be of the form:

$$T_{k+1} = \alpha T_{k-1}, \quad \text{where } k = 1, 2, 3, \ldots,$$

where is $0.85 < \alpha < 0.95$.

This is justified, because the decrement rule described in Section 4 would allow only small changes in the value of the control parameter. Several researchers have used this decrement rule and it has been shown to be successful [16,28].

We also chose a final value of the control parameter, T_f, to be a small enough number such that the acceptance ratio was very small. The initial value of T_0 was computed as follows:

$$T_0 = T_f^{-N} T_f.$$

The length of the inner loop, L, was based on the intuitive argument that for each value of T_k, a minimum amount of moves should be accepted. However, as T_k approaches 0, moves are accepted with decreasing probability, and thus L becomes very long. Hence, the length of the inner loop is ceiled by some constant. Other researchers [16,28] have used this rule.

8. EXPERIMENTAL RESULTS

We implemented the procedure described in this paper on placement of regular and irregular polygons described through line-segment geometry. Discretization of this geometry was accomplished through application of a scan algorithm to fill each polygon with V_{ijk}. The pairwise affinity relation a_{ij} was computed offline from the main annealing procedure for fixed rotations of 0, 90, 180 and 270 degrees.

Calibration of Procedure

To calibrate the annealing procedure, regular convex polygons were placed with the result shown in Figure 11. While, by Theorem 5.2, the method is not guaranteed to minimize the area of the bounding box, these results are clearly good.

Figure 12 depicts the performance of the annealing algorithm as the temperature control parameter in the annealing procedure decreases. Notice that the cost function does not decrease monotonically as would be the case with straight iterative improvement.

Figures 13 and 14 depict a typical irregular shape packing. Notice that placing the regular figures took only a small initial temperature while placing complex figures takes a large initial temperature and a small decrement. These results, however, are coarse approximations and better annealing schedules are expected to exist which will decrease the run time significantly.

9. SUMMARY

There is not yet enough practical experience with the method of simulated annealing to say definitively that it will realize its current promise. The method has several extremely attractive features, rather unique when compared with other optimization techniques. First, it is not "greedy," in the sense that it is not easily fooled by the quick payoff achieved by falling into unfavorable local minima. Second, changes which cause the greatest cost difference are sifted over when the control parameter T is large. These decisions become more permanent as T is lowered, and attention then shifts more to smaller refinements in the solution.

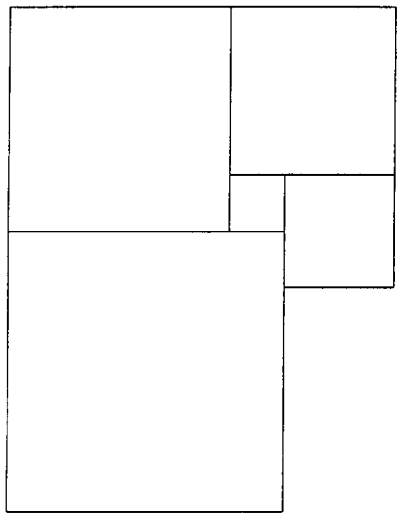

Figure 11. Placement of five regular polygons. $T_0 = 1500$. Decrement= 0.95. Length of Markov chain = 40. Number of chains = 1000.

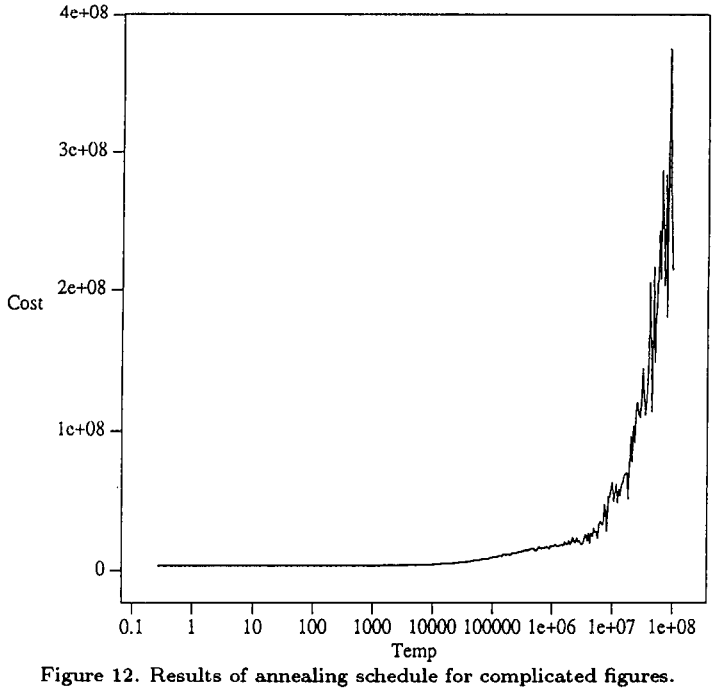

Figure 12. Results of annealing schedule for complicated figures.

10. FUTURE RESEARCH

Simulated Annealing is clearly a time-consuming procedure. Two possible methods exist for speeding up the process. The first is based on the granularity of the discretization and the second appeals to parallel processing.

The simulated annealing algorithm is fast for coarse granularities since the moves, Φ, are less computationally expensive than for finer granularities. However, the finer granularities are necessary to prevent overlapping and to achieve a usable result. Annealing at several different granularities can utilize a coarse approximation at the high T, and utilize a fine granularity at lower values of T. This structure yields a multigrid approach to the problem such as exploited in [29]. The problem structure is formulated as a pyramid as shown in Figure 15.

The second alternative is to appeal to parallel processing. The main difficulty is the maintenance of a global state S of the annealing process. This is further complicated by the desirability of using distributed memory multiprocessors or multicomputers, [30] which have no single global picture of the state. We are currently working on multigrid methods for this problem. In an

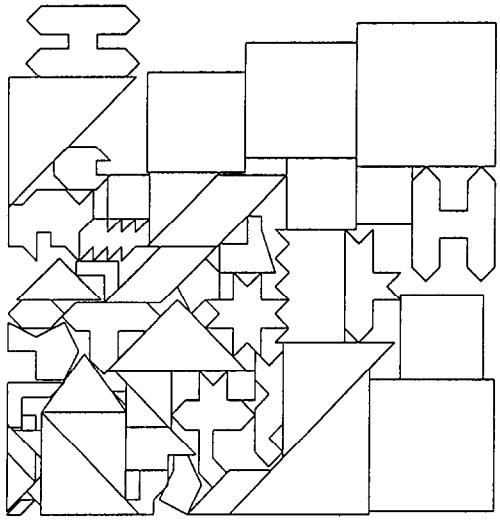

Figure 13. Complicated oriented bin packing results under variable width constraints. $T_0 = 10^6$. $T_\infty = 0.38$. Decrement= 0.99. Length of Markov chain= 2000. Packing density= 81.7%.

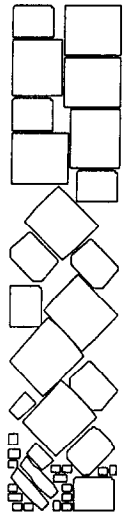

Figure 14. Complicated oriented bin packing results under fixed width constraints. $T_0 = 10^8$. $T_\infty = 0.25$. Decrement = 0.99. Length of Markov chain = 2050. Packing density = 81.4%.

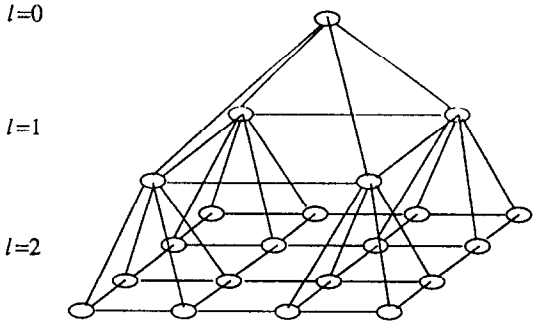

Figure 15. The pyramid architecture for a multigrid solution. At each level, the granularity increases by 4.

asynchronous multigrid, both the number of processors and the granularity change simultaneously during the problem solution. At the start of the annealing process, at high T, large moves, and most moves are accepted. To minimize the amount of state information that must be communicated, the number of processors involved is kept low. As the temperature decreases, more rejected moves are made and accepted moves are smaller. Thus, less state information needs to be communicated between processors, and more processors may be employed. Thus, in Figure 15, execution and processor utilization moves down through the pyramid hierarchy as the temperature decreases.

References

1. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, (1979).
2. C.H. Dagli, Knowledge-bases systems for the cutting stock problems, *European Journal of Operational Research* **44**, 160–166 (1990).
3. P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting-stock problem, *Operations Research* **9**, 849–859 (1961).
4. P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting-stock problem, *Operations Research* **11**, 863–888 (1963).
5. P.C. Gilmore and R.E. Gomory, Multistage cutting-stock problems of two and more dimensions, *Operations Research* **13**, 94–120 (1965).
6. A.M. Geoffrion and R.E. Marsten, Integer programming algorithms: A framework and state-of-the-art survey, *Management Science* **18** (19), 465–491 (1972).

7. R.W. Haessler, A note on computational modifications to the Gilmore-Gomory cutting stock algorithm, *Operations Research* **28** (4), 1001–1005 (1980).
8. H. Dyckhoff, A new linear programming approach to the cutting-stock problem (1981).
9. J.E. Beasley, Algorithms for unconstrained two-dimensional guillotine cutting, *J. Opl. Res. Soc.* **36** (4), 297–306 (1985).
10. J.E. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure, *Operations Research* **33** (1), 49–64 (1985).
11. B.R. Sarker, An optimum solution for one-dimensional slitting problems: A dynamic programming approach, *J. Opl. Res. Soc.* **39** (8), 749–755 (1988).
12. N. Christofides and C. Whitlock, An algorithm for two-dimensional cutting problems, *Operations Research* **25**, 30–44 (1977).
13. A.I. Hinxman, The trim-loss and assortment problems: A survey, *European Journal of Operational Research* **5**, 8–18 (1980).
14. C. Sechen and Sangiovanni-Vincenelli, The TimberWolf placement and routing package, *IEEE J. Solid State Circuits* **SC-20**, 510–522 (1985).
15. C. Sechen and Sangiovanni-Vincenelli, TimberWolf3.2: A new standard cell placement and global routing package, *Proceedings 23rd Des. Automation Conf.*, Las Vegas, pp. 432–439, (June, 1986).
16. C. Sechen and Sangiovanni-Vincenelli, Placement and global routing of integrated circuits using the simulated annealing algorithm, Ph.D. Dissertation, University of California at Berkeley, (1986).
17. E. Bonomi and J. Lutton, The N-city travelling salesman problem: Statistical mechanics and the Metropolis Algorithm, *SIAM Rev.* **26**, 551–568 (1984).
18. S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, Optimization by simulated annealing, IBM Research Report RC 9355, (1982).
19. P. Carnevalli, L. Coletti and S. Paternello, Image processing by simulated annealing, *IBM J. Res. Develop.* **29**, 569–579 (1985).
20. C.H. Dagli, Neural networks in manufacturing: Possible impacts on cutting stock problems, *Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing*, pp. 531–537, IEEE Computer Society Press, Los Alamos, CA, (1990).
21. Ising, A contribution to the theory of ferromagnetism, *Z. Phys* **31**, 253 (1925).
22. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, Equation of state calculations by fast computing machines, *J. of Chem. Physics* **21**, 1087–1092 (1953).
23. S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, Optimization by simulated annealing, *Science* **220**, 671–680 (1983).
24. V. Cerny, Thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm, *J. Opt. Theory Appl.* **45**, 41–51 (1985).
25. L. Aarts and P. van Laarhoven, Statistical cooling: A general approach to combinatorial optimization problems, *Philips J. Res.* **40**, 193–226 (1985).
26. D.D. Caviglia, G.M. Bisio and F. Curatelli, Neural algorithms for cell placement in VLSI design, Presented at the *International Conference on Neural Networks* (1989).
27. C. Sechen, Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing, *25th ACM/IEEE Design Automation Conference*, (1988).
28. D. Johnson, C. Aragon, L. McGeoch and C. Schevon, *Optimization by simulated annealing: An experimental evaluation, Parts I and II*, AT&T Bell Laboratories, (1987).
29. S. Franks, R. Khanna and B. McMillin, PAFMV—Pairwise asynchronous multigrid, *Proceedings 1990 International Conference on Parallel Processing*, August, 1990, pp. I-388-I-392; *Operations Research* (29) 1981, 1092–1104.
30. W. Athas and C. Seitz, Multicomputers: Message-passing concurrent computers, *Computer*, 9–25 (August, 1988).