A Simulated Annealing-Based Approach to Three-Dimensional Component Packing

S. Szykman

J. Cagan

Department of Mechanical Engineering,
Carnegie Mellon University,
Pittsburgh, PA 15213

This paper introduces a simulated annealing-based approach to three-dimensional component packing that employs simulated annealing to generate optimal solutions. Simulated annealing has been used extensively for two-dimensional layout of VLSI circuits; this research extends techniques developed for two-dimensional layout optimization to three-dimensional problems which are more representative of mechanical engineering applications. This research also provides a framework in which to solve general component layout problems.

1 Introduction

In this paper we present a simulated annealing-based approach to three-dimensional component packing, a subset of general layout problems, that employs simulated annealing to generate optimal solutions which maximize packing density and, if desired, fit components into a given container. Applications include shipping container packing, pallet loading and packing of the space shuttle cargo bay.

There is an abundance of literature pertaining to two-dimensional layout problems in the electrical engineering domain, where computational design tools that employ simulated annealing have been used to automate analog and digital circuit layout tasks for a number of years (e.g., Sechen and Sangiovanni-Vincentelli, 1985; Cohn et al., 1991). This technology is well established beyond the academic research community (see texts by Sechen, 1988; Wong et al., 1988; and Sherwani, 1992) and is used extensively for circuit layout in industry.

For circuit layout applications, the need to extend this technology to three dimensions has not arisen. Although VLSI circuits commonly extend into the third dimension, these circuits are modeled using multiple two-dimensional, or planar circuits. This so-called $2\frac{1}{2}$ dimensional model is adequate because although the circuits have thickness, electronic components remain essentially planar. For general component layout tasks in the mechanical engineering domain, the third dimension may not be negligible and a $2\frac{1}{2}$ dimensional formulation is not sufficient.

This research extends several techniques developed for two dimensional VLSI layout optimization to three-dimensional problems that are more representative of mechanical engineering applications. In the following section, we discuss other approaches to component layout and packing problems. In Section 3 we describe the simulated annealing algorithm and in Section 4 we present a series of experiments and results that show simulated annealing to be a promising approach for three-dimensional packing problems.

2 Related Work

Several research efforts have focused on layout problems that involve locating components in space while satisfying spatial relationships among them. These relationships consist of a noninterference condition, and may also include additional constraints on component placement. A variety of optimization techniques have been applied to three dimensional component layout problems by Udy et al. (1988), Fujita et al. (1991), Kim and Gossard (1991) and Landon and Balling (1994). These approaches have a number of disadvantages associated with the nonlinear nature of layout problems; they either require feasible starting points, linearization of nonlinear equations (which can lead to poor convergence) or require the use of gradients, and usually converge to local optima. For general problems having objective functions for which gradients must be calculated numerically, these techniques may be very time-consuming.

Component layout problems have also been approached using genetic algorithms by Kawakami et al. (1991) and Corcoran and Wainwright (1992). However, most of the research on packing problems has been done in the field of operations research, summarized in survey papers by Coffman et al. (1984), Dyckhoff (1990) and Dowsland and Dowsland (1992). A drawback of these methods is that they are generally heuristic methods which are only applicable to a specific class of problems, such as cutting parts from stock of fixed width or packing different sized rectangles into multiple bins of fixed dimensions.

The above techniques may be effective for a specific set of problems; however, they do not provide a basis from which general mechanical layout problems can be solved. This paper introduces an efficient approach to three-dimensional

Contributed by the Design Automation Committee for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received Feb. 1994; revised Oct. 1994. Associate Technical Editor: D. A. Hoeltzel.

component packing using the simulated annealing algorithm that is applicable to general layout problems. Simulated annealing can escape from local optima and is a zero-order algorithm that does not require the use of gradients, alleviating the drawbacks described above.

Because of the potential advantages of simulated annealing, there have been initial investigations into using the algorithm as a tool for solving packing problems. Cagan (1994) uses a simulated annealing-based technique to solve a geometrically constrained knapsack problem; Kämpke (1988) reports a solution to a one-dimensional bin packing test problem using simulated annealing that is superior to previously published solutions. Simulated annealing has also been shown to perform better than previously existing approaches on several classical test problems for facilities layout tasks (Jajodia et al., 1992).

3 Optimization by Simulated Annealing

3.1 The Simulated Annealing Algorithm. Simulated annealing (Kirkpatrick et al., 1983) is a stochastic optimization technique that has been used to solve continuous, ordered discrete and multi-modal optimization problems. In a typical simulated annealing algorithm an initial state is chosen and the value of the objective function at that point is calculated. A move is made to a new design state which is then evaluated. If this move leads to an improvement in the objective function, the new design is accepted and becomes the current design state.

If the move leads to an inferior state, it may still be accepted with some probability. This probability is a function of a decreasing parameter called "temperature," based on an analogy with the annealing of metals, given by:

$$P_{accept} = e^{-\frac{\Delta C}{T}}, \tag{1}$$

where ΔC is the change in objective function due to the move and T is the current temperature. The temperature starts out high and decreases with time. Initially, moves made through the state space (and therefore the objective function space) are almost random, resulting in a broad exploration of the objective function space. As the probability of accepting inferior moves decreases, they tend to get rejected, allowing the algorithm to converge to an optimum.

Simulated annealing is well suited to three-dimensional packing problems that can be problematic to conventional gradient-based optimization methods due to discontinuities and/or severe nonlinearities in their objective function spaces. Discontinuities can be dealt with because simulated annealing is a zero-order algorithm requiring no derivative information. Since there is always a probability of accepting inferior steps (i.e., moves away from the direction of an optimum), simulated annealing algorithms are able to escape from local optima.

3.2 Adaptive Annealing Schedules. The temperature in a simulated annealing algorithm is controlled by an *annealing schedule*. The annealing schedule specifies an initial temperature, how often the temperature should be lowered, by how much it should decrease and a final temperature. Simple annealing algorithms have a fixed annealing schedule; the user selects an initial temperature, decides on a constant number of iterations to be taken at each temperature and sets a temperature reduction multiplier (less than, but usually close to, 1.0) by which the temperature is multiplied to decrease it. The user also decides on a final temperature or, equivalently, an overall number of iterations before the algorithm terminates.

The performance of an annealing schedule is problem-dependent; a temperature profile that performs well on one problem may perform poorly on another. In addition, it is difficult to determine *a priori* what are good values for the annealing schedule parameters. To address this issue, a number of adaptive annealing schedules have been proposed (e.g., Huang et al., 1986; Ingber, 1993) that use feedback about the algorithm's performance to control the temperature. The advantages of such adaptive schemes are (*1*) that they free the user from the task of devising a good annealing schedule, (*2*) the schedules are not fixed and are therefore more robust, and (*3*) because they use performance feedback, the temperature profile is automatically tailored to the current problem.

The algorithm used for the experiments in this paper employs an adaptive annealing schedule proposed by Huang et al. (1986), which uses statistical information about the objective function values of the visited states to control the temperature. With this schedule, the initial temperature is set to:

$$T_i = -\frac{3\sigma}{\ln P}, \tag{2}$$

where σ is the standard deviation of objective function values (obtained by taking a random walk through the state space and accepting every move), and P is the desired probability of accepting a state that is more than three standard deviations worse than the current one. Huang et al. suggest using a value of about 0.85 for P. The temperature is reduced when equilibrium is achieved at a given temperature or when a maximum number of steps has been exceeded. The temperature is updated using the following equation:

$$T_{new} = T_{current} * e^{-\left(\frac{0.7\,T_{current}}{\sigma}\right)}. \tag{3}$$

As the temperature drops and inferior steps are rejected more often, the distribution of objective function values changes. Therefore, rather than using the initial value of σ from Eq. (2), a new value of σ is calculated at each temperature.

The simulated annealing algorithm continues to run until a freezing condition is satisfied, at which point the algorithm presumably is no longer performing useful search and terminates. Huang et al. describe the annealing schedule and the equilibrium and freezing conditions in greater detail.

3.3 Multi-Objective Optimization. General layout tasks for engineering applications typically have a number of objectives which often compete with one another. Examples of such objectives are packing density, measures of performance affected by the arrangement of components, minimization of routing lengths for wires that connect components, as well as manufacturability or assembly considerations.

To allow optimization over multiple design objectives, the formulation of the work presented in this paper incorporates a generic objective function, F, consisting of a weighted sum of the form:

$$\mathrm{F} = W_1 f_1 + W_2 f_2 + \ldots + W_n f_n, \tag{4}$$

where f_i is the value of the ith objective and W_i is the weight for the ith term. The weights, W_i, allow the user to specify the relative importance of the multiple objectives. The objective function terms used for three dimensional component packing are described in greater detail in Section 4.

3.4 A Strategy for Efficient Move Selection. Simulated annealing algorithms take a step from a current state to a new state by applying an operator, or move, from a predetermined move set. In simple algorithms, the probability of selecting a given move is set and remains fixed as the algorithm runs. Moves that are useful (and therefore are accepted often) at high temperatures may tend to get rejected

at lower temperatures; the reverse may be true for other moves. Rather than having fixed probabilities, search can be made more efficient by making the probability of selecting a move high when the move is likely to be accepted, and low when it is likely to be rejected.

Hustin and Sangiovanni-Vincentelli (1987) propose a strategy in which the probabilities of applying various moves are updated based on past performance after each temperature reduction. Move probabilities are calculated so that the moves most likely to be chosen at a given temperature are those that caused the most significant changes in objective function values at the previous temperature. As a measure of performance, we calculate for each of the m moves in the move set an average change in cost attributed to that move, called the *quality factor*. The quality factor for the ith move is defined as:

$$Q_i = \frac{1}{n_i} \sum_{\substack{\text{accepted} \\ \text{steps}}} |\Delta C|, \quad i = 1, \ldots, m, \tag{5}$$

where n_i is the number of times the ith move was attempted at the previous temperature, ΔC is the change in cost due to a step and m is the number of moves in the move set. Note that the values of ΔC are only summed over accepted steps because rejected moves do not affect the objective function. If a move has not been attempted (i.e., $n_i = 0$), the Q_i is set equal to zero.

To ensure that every move is given the opportunity to be useful, each move is assigned a minimum probability, P_{min}, usually around 2 percent. If we let P_r be the remaining probability (i.e., $1 - P_{min} * m$), we can distribute P_r among the various moves letting the probability of selecting the ith move be:

$$P_i = P_{min} + P_r \frac{Q_i}{Q_{total}}, \tag{6}$$

where Q_{total} is the sum of the quality factors over all of the moves.

As can be seen from Eq. (5), a quality factor can become large when a move is accepted often and/or when a move results in relatively large changes in the objective function. The larger the quality factor for a move, the greater the probability that it will be applied at the next temperature.

4 Three-Dimensional Component Packing

In this section, we describe the component packing algorithm and present results of experiments designed to test its effectiveness. In Section 4.1 we describe the results of applying simulated annealing to a basic component packing problem for which the objective is to maximize packing density. In Section 4.2 we expand the problem formulation to include the ability to specify containers into which components are required to fit.

The results given in Sections 4.1 and 4.2 show both qualitatively and quantitatively that simulated annealing is able to optimize three-dimensional component packings. However, global optima for these problems are not known, making it difficult to assess the overall performance of the algorithm. To address this problem, in Section 4.3 simulated annealing is applied to a series of simpler three-dimensional packing problems having known global optima, used as benchmark tests.

4.1 The Component Packing Problem

4.1.1 The Move Set. The simulated annealing packing algorithm uses a move set that borrows concepts from the VLSI (two-dimensional) layout literature. Three types of moves are used to perturb a layout: *translate*, *rotate*, and *swap* (which switches the location of two components). These moves are applied according to the move selection scheme described in Section 3.4. The move set consists of one rotate move and one swap move, but several (usually 5 to 15) translate moves. Each of the translate moves has a translate distance, D associated with it, where D ranges from a minimum value to a maximum value. Having multiple translate distances allows the algorithm to take steps of varying size.

When a translate move is applied, a component is moved a distance $D\mathbf{V}$ (i.e., a distance D in the direction $\mathbf{V}$) where $\mathbf{V}$ is a randomly generated unit vector. To speed up geometric interference testing rotations are currently constrained to multiples of 90 degrees (chosen at random by the rotate move). This limitation is independent of the simulated annealing algorithm and can be relaxed or eliminated when components having more complex geometries are used.

4.1.2 The Objective Function. For simple component packing tasks, the objective is to minimize the inverse of a packing density (maximizing the density). A simple inverse volumetric density given by:

$$f_1 = \frac{V_{bb}}{\sum_{i=1}^{n} V_i}, \tag{7}$$

is used as the first term of the generic objective function, Eq. (4), where V_{bb} is the volume of the bounding box of the design (a box that completely encloses the design), n is the number of components, and V_i is the volume of the ith component.

As a layout is perturbed, components are permitted to overlap. This allows components to move through one another and results in a more thorough search of the state space. Since component overlap is not desired in a final design, overlap is generally eliminated by penalizing it in the objective function (Jepsen and Gelatt, 1983). Thus the objective function also has a second term, f_2, given by:

$$f_2 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i-1} O_{ij}^2 \right), \tag{8}$$

where O_{ij} is the overlap between the ith and the jth component. Equation (8) corresponds to the sum of the squares of the terms that lie below the diagonal in an "overlap matrix." O_{ij} is defined as:

$$O_{ij} = O_{Xij} + O_{Yij} + O_{Zij}, \tag{9}$$

where O_{Xij}, O_{Yij} and O_{Zij} are the length of overlap between the ith and the jth components in the X, Y and Z coordinate directions, respectively. Calculation of this Manhattan distance-based overlap is computationally efficient given the geometric interference tests used. Overlaps for cylinders are usually determined using a bounding box around the cylinder. To calculate interferences between two cylinders that have the same orientation, this simplification can lead to more significant errors so overlaps are computed exactly.

In Section 4.2, a second penalty term is added to the objective function to fit components into a specified container. Because the penalty terms go to zero in a successful optimization (i.e., a final design is free of component overlap and all components fit inside the specified container), this is not a true example of multi-objective optimization. However, the linearly weighted objective function given by Eq. (4) is a common approach to multi-objective optimization and is also applicable if additional objectives are added to the problem.

4.1.3 Experimental Results. Szykman and Cagan (1993) introduce a simulated annealing approach to component

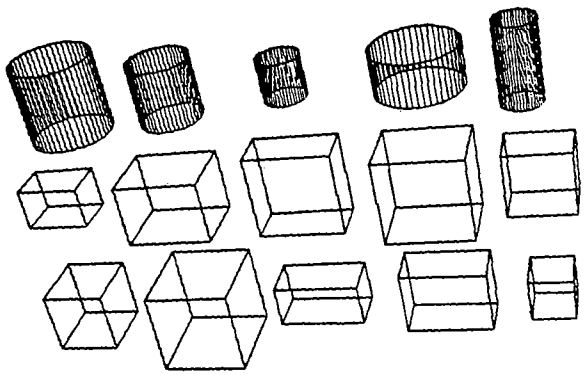


Fig. 1 Components used for the packing problem

packing which utilizes a move set that adds and removes components to build up complete designs from incomplete ones, while not permitting component overlap. We shall refer to this algorithm as the *build-up* algorithm. Because the current algorithm has a move set that perturbs component locations and orientations in a complete design, as described in Section 4.1.1, we refer to it as the *perturbation* algorithm.

At high temperatures a simulated annealing algorithm accepts virtually all inferior steps and resembles a random search. At low temperatures, inferior steps tend to be rejected and the algorithm behaves like a downhill search accepting only improvement steps. Szykman and Cagan (1993) present results that compare the performance of the build-up approach with four different fixed (nonadaptive) annealing schedules to random search and downhill search—the two extremes of a simulated annealing algorithm.

In this section, we compare the perturbation approach to the random search results and the best group of build-up results previously obtained.[1] We then examine the quality of the results obtained and the consistency of the algorithms. No comparison is made with the downhill search results because that algorithm was unable to place all of the components before becoming trapped in a local minimum.

The packing task used for the experiments consists of placing 15 components—10 blocks and 5 cylinders[2]—of varying sizes (see Fig. 1). The ratios of the maximum block dimension to the minimum block dimension range from 1.0 to 2.0 and the ratios of cylinder length to radius range from 0.5 to 2.0. The maximum and minimum dimensions are the same for the blocks and the cylinders.

Three different perturbation algorithms were run twelve times each. The first algorithm, called algorithm A, applied moves using constant probabilities and used a simple fixed annealing schedule (see Section 3.2). The second algorithm (algorithm B) had constant move probabilities but the fixed annealing schedule was replaced with the adaptive one described in Section 3.2. The third algorithm (called *final perturbation*) kept the adaptive annealing schedule and replaced the constant move probabilities with the technique for move selection described in Section 3.4.

In Fig. 2, the results of these runs are compared to previously obtained results from the random search and build-up algorithms (also twelve runs each). The objective

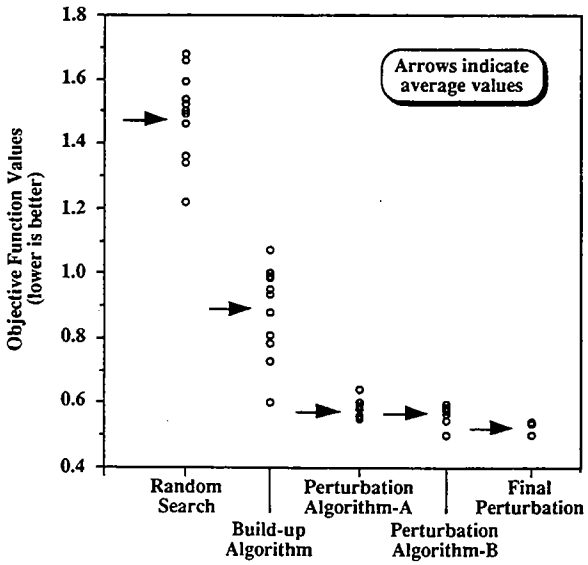


Fig. 2 Objective function values for various component packing approaches

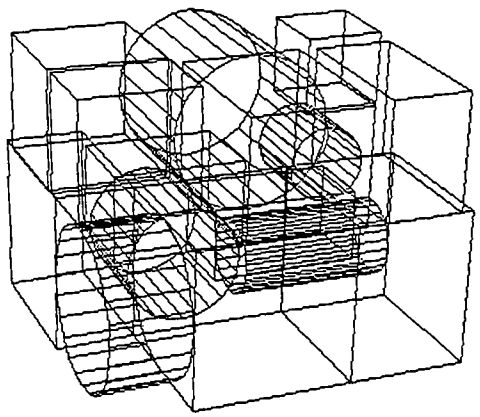


Fig. 3 Best layout using the final perturbation algorithm

Table 1 Summary of experimental results

Algorithm	Best Value	Average Value	Standard Deviation
Random Search	1.222	1.478	0.137
Build-Up	0.605	0.893	0.137
Perturbation A	0.553	0.579	0.026
Perturbation B	0.504	0.573	0.026
Final Perturbation	0.503	0.527	0.020

function values for the best designs found for each of the runs are plotted; the arrows indicate the average values of the results for each algorithm. The best values, average values and standard deviations for the experiments are given in Table 1. The packing corresponding to the best design found by the final perturbation algorithm is shown in Fig. 3; note that the result is a tightly packed cube as expected for a dense component packing.

4.1.4 Discussion of Experimental Results. The quality of the average and best results is clearly an important factor in determining the effectiveness of the perturbation approach. As expected, both the build-up and the perturbation algorithms produce better (i.e., more densely packed) layouts than the random search algorithm (see Table 1). The perturbation algorithms perform on average 35–41 percent better than the build-up algorithm and 61–64 percent better than the random search algorithm.

[1]For reasons that do not apply to the current algorithm, the build-up approach uses a surface area density analogous to the volumetric density given by Eq. (7). In Sections 4.1.3 and 4.1.4, we use this area density in order to compare the perturbation approach to previous results from the build-up approach. In the remainder of this paper (and in general), the perturbation algorithm uses a volumetric density term as an objective.

[2]Components in the current implementation are limited to blocks and cylinders to simplify geometric interference testing. Work on representing more general objects and performing interference tests on them has been done by Udy et al. (1988) and Gursoz et al. (1991) among others.

A second indicator of the effectiveness of the algorithm is the standard deviation of the results, which provides a measure of the consistency of convergence to good designs. The standard deviation of the random search and the build-up algorithms are both 0.137 (see Table 1). In contrast, the standard deviations of the perturbation algorithms are 81–85 percent less. Not only do the perturbation algorithms find higher quality results than the build-up approach, but they do so much more consistently. The difference in consistency of performance is illustrated qualitatively by the spread in the results shown in Fig. 2.

The perturbation algorithms perform on average considerably better than the build-up algorithm due to a number of ways in which they differ. The most significant of these differences are that perturbation algorithm A:

- permits and penalizes overlap between components (as described in Section 4.1.2),
- has a new move set that allows small changes in objective function values (described in Section 4.1.1),
- eliminates the need for an objective function to evaluate incomplete designs[3].

Although the build-up approach can, in principle, be implemented to allow overlap, the latter two distinctions are fundamental differences which cannot be separated from the basic algorithms. In particular, allowing small changes in the objective function improves convergence by letting the algorithm explore a neighborhood of designs having similar objective function values. Further improvements in performance of the perturbation algorithm are achieved by adding an adaptive annealing schedule (algorithm B) and adding to that a better move selection technique (final perturbation algorithm).

The best design found by algorithm B is better than that found by algorithm A; the average results and standard deviations are about the same (see Table 1). However, as described in Section 3.2, the advantage of the adaptive annealing schedule is that the temperature profile is automatically tailored to the current problem, eliminating wasted search at temperatures that are too high, or after "thermal" equilibrium has been reached. Algorithm A (without the adaptive annealing schedule) ran for 500,000 iterations. The average run for algorithm B (with the adaptive schedule) was less than 150,000 iterations—a 70 percent reduction. Changing the fixed annealing schedule in algorithm A to run 150,000 iterations instead of 500,000 results in an average final objective function value of 0.703 (compared to 0.573 for algorithm B). The effect of the adaptive annealing schedule is quite significant.

The move selection technique described in Section 3.4 uses past performance to dynamically adjust the probabilities of applying a given move. Moves that are most likely to be accepted are applied more often than those that are likely to be rejected. Adding the move selection technique (final perturbation algorithm) did not significantly affect the quality of the best design found, but improved both the average performance and the consistency of the algorithm. The final perturbation algorithm runs at a rate of over 1150 iterations/second on a DEC Alpha 3000 workstation. In the remainder of the paper, "perturbation algorithm" refers specifically to the final perturbation algorithm.

4.2 Fitting Components into a Container. The extension to requiring that components fit into a given container is straightforward. For this problem we specify an additional "component" that represents the container, and add to the objective function a third term, f_3, that penalizes components which protrude from the container:

$$f_3 = \sum_{i=1}^{n} P_i^2, \tag{10}$$

where n is the number of components and P_i is the protrusion from the container of the ith component, given by:

$$P_i = P_{Xi} + P_{Yi} + P_{Zi}, \tag{11}$$

where P_{Xi}, P_{Yi} and P_{Zi} are the lengths of protrusion of the ith component in the X, Y and Z coordinate directions, respectively. This penalty is similar to the overlap penalty given by Eqs. (8) and (9), but penalizes the protrusion of each component from the container, rather than the length of overlap between two components. Figure 4 shows a packing of the fifteen components from Fig. 1 that were constrained to fit inside a short wide container, using the perturbation approach.

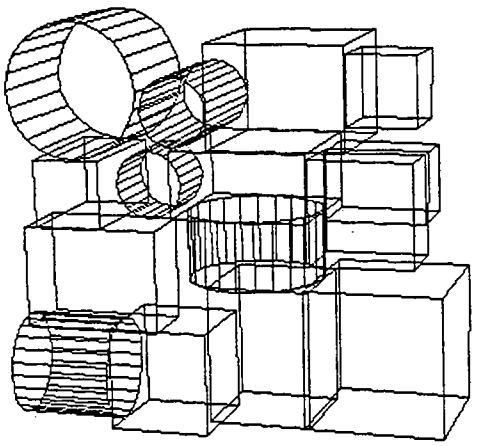


Fig. 4 Packing into a shallow wide container

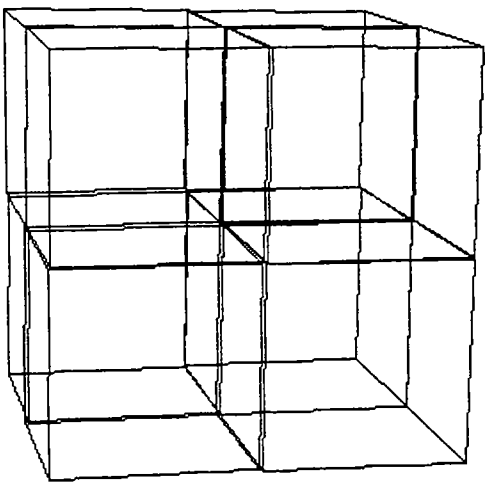
4.3 The Search for a Global Optimum. The best design found by the perturbation approach in the experiments presented in Section 4.1 has a volumetric packing density of over 80 percent. Because the global optimum for this problem is not known, however, it is difficult to appraise the overall performance of the perturbation algorithm. Given impractically rigorous mathematical conditions, simulated annealing is guaranteed to find a global optimum (Lundy and Mees, 1986); for general problems these conditions are not satisfied.

In order to assess the performance of the perturbation algorithm, several three-dimensional packing problems with known global optima were created and solved using the algorithm. These benchmark tests were created by taking a cube and dividing it into a number of smaller blocks. The resulting blocks are given a random initial configuration and the layout is optimized for packing density using the perturbation algorithm. Since the components can be reassembled into the initial cube, the global optimum is a 100 percent packing density.

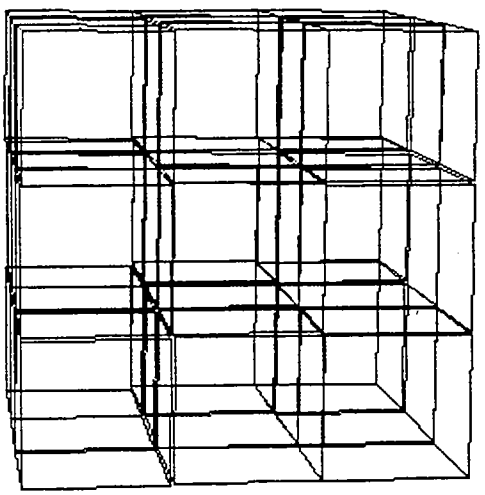
The first test problem consists of packing eight identical cubes and the second test problem consists of packing 27 identical cubes. Solutions to these two problems found by the perturbation algorithm are shown in Figs. 5(a) and 5(b), respectively.

Because the blocks in the first two test problems are identical, these problems have a number of symmetrical solutions. A third test problem which consists of packing eight unique rectangular blocks was created. In this problem, the largest block has 3.4 times the volume of the smallest block, no two blocks are identical and every block has three sides of different lengths. This packing task is considerably

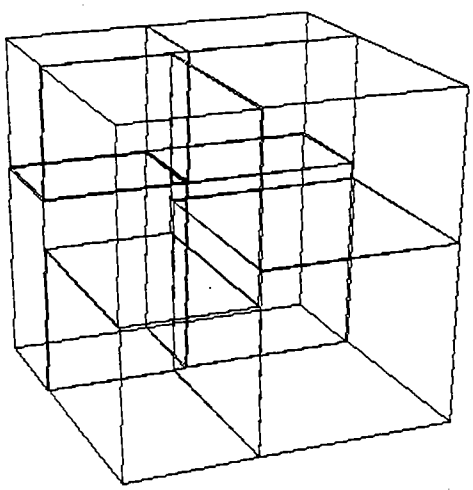
[3]A potential drawback of the build-up approach is that due to the move set, which adds and removes components, design states are often incomplete designs making evaluation difficult.



(a)



(b)



(c)

Fig. 5 Cubes composed of 8 cubes (a), 27 cubes (b), and 8 unique blocks (c)

more difficult than the previous two problems both because the blocks are not interchangeable, and because the correct orientation for each block must be found. The solution to this problem found by the perturbation algorithm is shown in Fig. 5(*c*).

Because simulated annealing is a stochastic optimization technique, annealing algorithms often become trapped in local optima near the global optimum, returning different answers to the same problem over multiple runs. Surprisingly, the perturbation algorithm was able to find the globally optimal configuration for the first test problem nineteen times out of twenty runs. The performance of the algorithm was less consistent for the second and third problems. However, the algorithm was able to find the globally optimal configuration, and the average packing density for each of the two problems was over 85 percent.

It should also be noted that packings will approach but usually not reach 100 percent packing density. For instance, the packing shown in Fig. 5(*a*) has a packing density of 97.5 percent though it is obviously the configuration for the global optimum. This is a result of minute gaps between components that are due to having a finite minimum move size. These gaps can be reduced by reducing the minimum move size if they are unacceptably large.

For general three-dimensional layout problems the global optima are usually not known, and thus global optimality can rarely be verified. However, the results of the benchmark tests confirm that for certain classes of problems the perturbation algorithm is able to find the global optimum and consistently performs well, demonstrating that this is a promising approach for three-dimensional layout optimization.

5 Concluding Remarks

Due to current trends toward product miniaturization, mechanical engineering layout optimization has become an important area of research. The simulated annealing technology used for two-dimensional VLSI layout, including an adaptive annealing schedule, a component move set and a dynamic move selection strategy, is well established and has proven to be viable in industrial practice. This work proposes the perturbation algorithm for optimal component packing, a subset of general layout problems, which successfully extends these VLSI techniques to three dimensions for mechanical engineering applications.

The packing examples presented in this paper are limited to blocks and cylinders to simplify geometric interference testing; however, this limitation is independent of the simulated annealing approach. Future research will examine the tradeoffs between accommodating more complex geometries and the cost of increased computational resources. To solve general component layout problems, this work is also being extended to include a spatial constraint language and additional engineering design objectives.

Acknowledgments

The authors would like to thank Rick Clark, Pratip Dastidar, Tom Mitchell, Rob Rutenbar and Paul Weisser for their invaluable discussions regarding this work, as well as the National Science Foundation (under grant # DDM-9258090) and United Technologies (supplying matching funds) for supporting this research.

References

Cagan, J., 1994, "A Shape Annealing Solution to the Constrained Geometric Knapsack Problem," *Computer-Aided Design*, Vol. 28, No. 10, pp. 763–769.

Coffman, E. G., Jr., Garey, M. R., and Johnson, D. S., 1984, "Approximation Algorithms for Bin-Packing—An Updated Survey," *Algorithm Design for Computer System Design*, Ausiello, G., M. Lucertini and P. Serafini, eds., Springer-Verlag, New York, pp. 49–106.

Cohn, J. M., Garrod, D. J., Rutenbar, R. A., and Carley, L. R., 1991, "KOAN/ANGRAM II: New Tools for Device-Level Analog Placement and Routing," *IEEE Journal of Solid-State Circuits*, Vol. 23, No. 3, pp. 330–342.

Corcoran, III, A. L., and Wainwright, R. L., 1992, "A Genetic Algorithm for Packing in Three Dimensions," *Applied Computing: Technologi-*

cal Challenges of the 1990's—Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing, Kansas City, KS, pp. 1021–1030.

Dowsland, K. A., and Dowsland, W. B., 1992, "Packing Problems," *European Journal of Operational Research*, Vol. 56, pp. 2–14.

Dyckhoff, H., 1990, "A Typology of Cutting and Packing Problems," *European Journal of Operational Research*, Vol. 44, pp. 145–159.

Fujita, K., Akagi, S., and Hase, H., 1991, "Hybrid Approach to Plant Layout Design Using Constraint-Directed Search and an Optimization Technique," *Advances in Design Automation 1991: Proceedings of the 17th ASME Design Automation Conference*, Vol. 1, Miami, FL, September 22–25, pp. 131–138.

Gursoz, E. L., Choi, Y., and Prinz, F. B., 1991, "Boolean Set Operations on Non-Manifold Boundary Representation Objects," *Computer-Aided Design*, Vol. 23, No. 1, pp. 33–39.

Huang, M. D., Romeo, F., and Sangiovanni-Vincentelli, A., 1986, "An Efficient General Cooling Schedule for Simulated Annealing," *ICCAD-86: IEEE International Conference on Computer-Aided Design—Digest of Technical Papers*, Santa Clara, CA, November 11–13, pp. 381–384.

Hustin, S., and Sangiovanni-Vincentelli, A., 1987, "TIM, a New Standard Cell Placement Program Based on the Simulated Annealing Algorithm," IEEE Physical Design Workshop on Placement and Floorplanning, Hilton Head, SC, April.

Ingber, L., 1993, "Simulated Annealing: Practice Versus Theory," *Mathematical and Computer Modelling*, Vol. 18, No. 11, pp. 29–58.

Jajodia, S., Minis, I., Harhalakis, G., and Proth, J-M., 1992, "CLASS: Computerized LAyout Solutions Using Simulated Annealing," *International Journal of Production Research*, Vol. 30, No. 1, pp. 95–108.

Jepsen, D. W., and Gelatt, Jr., C. D., 1983, "Macro Placement by Monte Carlo Annealing," *Proceedings of the IEEE International Conference on Computer Design*, November, pp. 495–498.

Kämpke, T., 1988, "Simulated Annealing: Use of a New Tool in Bin Packing," *Annals of Operations Research*, Vol. 16, pp. 327–332.

Kawakami, T., Minagawa, M., and Kakazu, Y., 1991, "Auto Tuning of 3-D Packing Rules Using Genetic Algorithms," *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS '91*, Vol. 3, Osaka, Japan, November 3–5, pp. 1319–1324.

Kim, J. J., and Gossard, D. C., 1991, "Reasoning on the Location of Components for Assembly Packaging," ASME JOURNAL OF MECHANICAL DESIGN, Vol. 113, No. 4, pp. 402–407.

Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P., 1983, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, pp. 671–679.

Landon, M. D., and Balling, R. J., 1994, "Optimal Packaging of Complex Parametric Solids According to Mass Property Criteria," ASME JOURNAL OF MECHANICAL DESIGN, Vol. 116, pp. 375–381.

Lundy, M., and Mees, A., 1986, "Convergence of an Annealing Algorithm," *Mathematical Programming*, Vol. 34, pp. 111–124.

Sechen, C., 1988, *VLSI Placement and Global Routing Using Simulated Annealing*, Kluwer Academic Publishers, Boston.

Sechen, C., and Sangiovanni-Vincentelli, A., 1985, "The TimberWolf Placement and Routing Package," *IEEE Journal of Solid-State Circuits*, Vol. 20, No. 2, pp. 510–522.

Sherwani, N., 1992, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, Boston.

Szykman, S., and Cagan, J., 1993, "Automated Generation of Optimally Directed Three Dimensional Component Layouts," *Advances in Design Automation 1993: Proceedings of the 19th ASME Design Automation Conference*, Vol. 1, Albuquerque, NM, September 19–22, pp. 527–537.

Udy, J. L., Balling, R. J., Benzley, S. E., and Landon, M. D., 1988, "Computation of Interferences Between Three-Dimensional Objects and the Optimal Packing Problem," Advances in Engineering Software, Vol. 10, No. 1, pp. 8–14.

Wong, D. F., Leong, H. W., and Liu, C. L., 1988, *Simulated Annealing for VLSI Design*, Kluwer Academic Publishers, Boston.