Submitted to : *Computer Aided Design*, 2000

A Survey of Computational Approaches to Three-dimensional Layout Problems

Jonathan Cagan[1] **Kenji Shimada** **Su Yin**

Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA

1 Introduction

Component layout plays an important role in the design and usability of many engineering products. The layout problem is also classified under the headings of packing, packaging, configuration, container stuffing, pallet loading or spatial arrangement in the literature. The problem involves the placement of components in an available space such that a set of objectives can be optimized while satisfying optional spatial or performance constraints.

Whereas the technologies for circuit board and IC chip layout have advanced significantly during the past two decades and many commercial CAD tools have been available, the same is not to be said for three-dimensional mechanical layout methods and tools. While the number of components to be placed in a mechanical system is modest compared to that of a VLSI system, the increased combinatorial complexity over the two-dimensional layout problem and the geometric complexity of 3-D non-uniform components and container spaces make the mechanical layout synthesis a challenging task. Current tools available in practice to designers to aid in the general mechanical layout process mostly remain at the stages of physical or electronic models with the assistance of manual adjustment and visual feedback. The needs arising in the product layout and rapid prototyping for compact and complex products, quick turnaround time and efficient use of resources justify the development of effective layout synthesis methods for 3-D components of complex geometry.

[1] Author of contact: (412) 268-3713, cagan@cmu.edu

The difficulty in automating the mechanical and electro-mechanical layout process stems from 1) the modeling of the design objectives and constraints, 2) the efficient calculation of the objectives and constraints, 3) the identification of appropriate optimization search strategies.

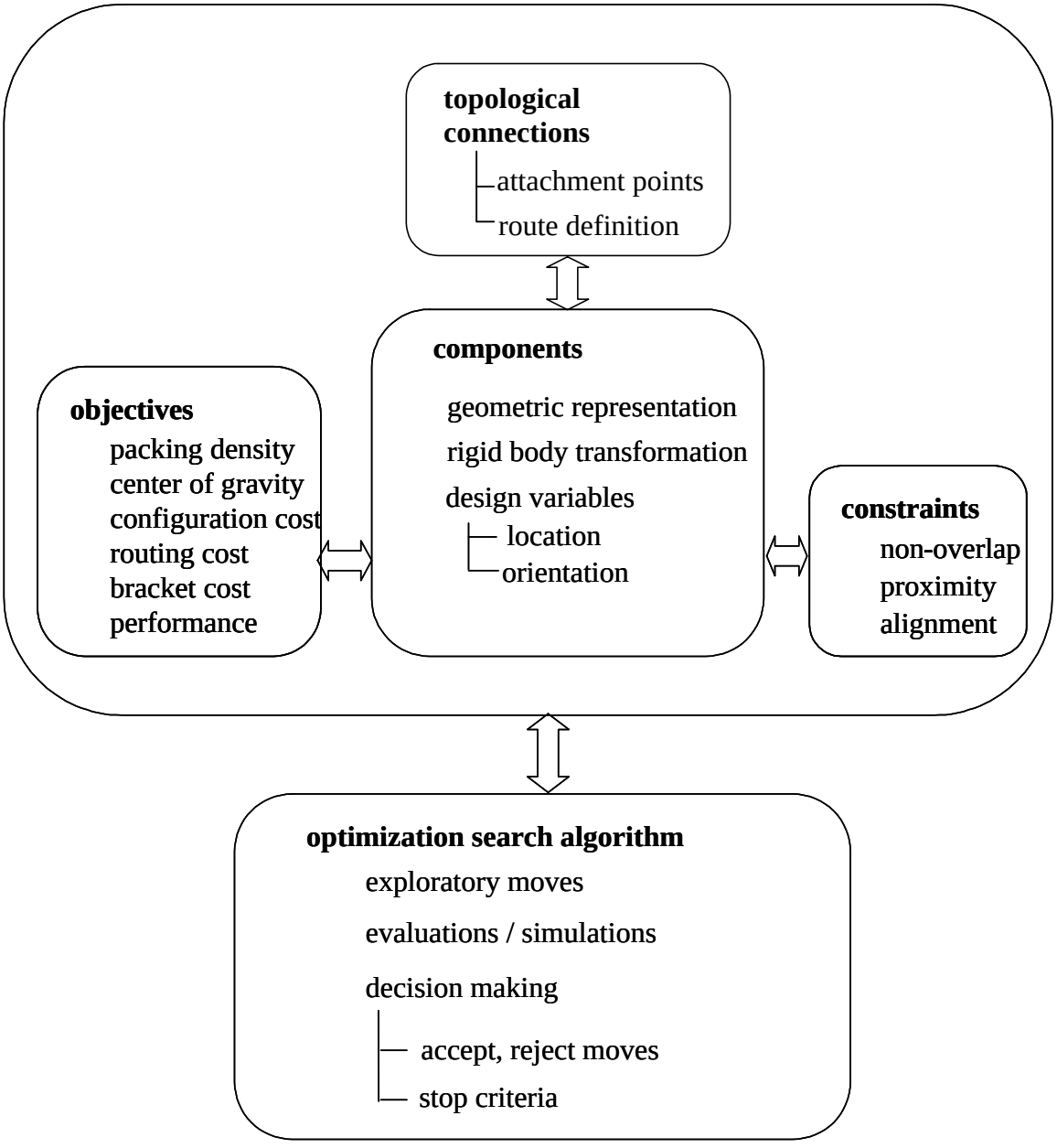

Figure 1. Major constituent parts for generic layout synthesis.

A number of design goals can be modeled as layout objectives. Simulations may be necessary to test the thermal, stress or vibration properties of the package. In addition, a set of constraints often has to be satisfied to ensure the applicability of the layouts. Efficient calculations of objectives and constraints are necessary to solve the layout problem in reasonable time since the analyses of objectives and constraints can be computationally expensive and a large number of evaluations may be required to achieve convergence. The search space of the layout problem is nonlinear and multi-modal, making it vital to identify a suitable algorithm to navigate the space and find good quality solutions.

Figure 1 illustrates the major constituent parts for solving a generic layout problem. Geometric representations of components and the container space are necessary for the check of interference and clearance. Since the interference calculation between components of complex geometry is computationally expensive, different levels of detail may be desirable for coarse and refined evaluations at different stages of the problem solving process. The locations and orientations of components are the design variables to be determined. Rigid body transformation is utilized to record the position and orientation of each component in the global coordinate frame. The layout and packaging goals are usually formulated as objective functions. The objectives may reflect the cost, quality, performance and service requirements. Various constraints may be necessary to specify spatial relationships between components. A common constraint is no component overlap and no container protrusion. Other constraints may include the proximity or alignment between components. Topological connections are necessary if tube routing, for example, is involved in addition to the component placement. The specifications of components, objectives, constraints, and topological connections define a layout problem and an optimization search algorithm takes the problem formulation and identifies promising solutions by evaluating design alternatives and evolving design states. Analyses of objectives and constraints vary from problem to problem. However, the optimization search technique and geometric representation and the resulting interference evaluation are problem independent and are thus the focus for a generic layout tool.

This paper examines the characteristics of the layout space and provides a survey of the current state-of-the-art in technologies for the three-dimensional layout problem. The paper is organized as follows: Section 2 shows a fractal analysis of the layout space and a continuum of optimization search algorithms as a basis for further discussions on various search techniques and their effectiveness. Section 3 reviews the optimization algorithms and geometric representations utilized in the layout problem, the advantages and limitations of different methods, and suitable

areas of applications. Section 4 discusses the key issues and strategies in building an effective and efficient layout synthesis tool.

2 Layout Space Characteristics and Solution Approaches

2.1 Spectral Density Analysis of the Layout Space

It is generally agreed that the 3-D layout space is nonlinear and multi-modal. Deterministic algorithms are unable to navigate such a space for globally near-optimal solutions, and stochastic algorithms are usually required for solutions of good quality. Are there any properties or regularities of the space that might help explain why some search methods are more effective than others in the layout space exploration? While it is impossible to visualize the space in general since it is multi-dimensional and the design variables are coupled, an energy landscape approach is utilized here to show the characteristics of the 3-D layout space.

Sorkin explored the effectiveness of simulated annealing algorithms on VLSI layout space (Sorkin, 1991, Sorkin, 1992). He revealed that the space is fractal-like. A fractal is a geometric figure that has built-in self-similarity in which the figure repeats itself on an ever-diminishing scale (Peitgen and Saupe, 1988). Fractalness has strong connections to natural shapes such as coastlines, snowflakes, clouds, and mountains. A function *f* is fractal if the distribution of $f(X')$ conditional on $f(X)$, X, and X', is normal with mean *0* and variance proportional to $d(X,X')^{2H}$, where *H* is a parameter in *(0,1)* that represents the scaling property of the fractal space. Since this definition involves probability distribution that is difficult to check, a simpler measure of fractalness is to sample over random values of X and X' for a given function *f* and check the satisfaction of the power-law relation:

$$\mathrm{E}[(f(X') - f(X))^2] \propto d(X',X)^{2H}, \qquad (1)$$

where E is the expectation of the function, $d(X', X)$ is a function representing the distance between state X and X', or the number of steps it takes to change a state from X to X'. Equation (1) is the mathematical expression of the fractalness, or the self-similarity characteristics.

The characteristics of the 3-D layout space are analyzed by following the practice discussed in Sorkin (1992). A random walk is taken on the 8-cube packing problem, where 8

identical cubes are packed into a container of 8 times the size of the individual cubes (Cagan et. al, 1998). A random walk is a sequence of points $X(t)$ (t = 0, 1, 2, …) where $X(t+1)$ is generated by a random move from $X(t)$. The time-energy series and spectral density of the random walk are then examined. The energy (the objective function value) time-series $f(X(t))$ plot is shown in Figure 2 and its spectral density $S(f)$ of the energy time-series $f(X(t))$ is generated using MATLAB and the statistics package S-Plus and shown in Figure 3.

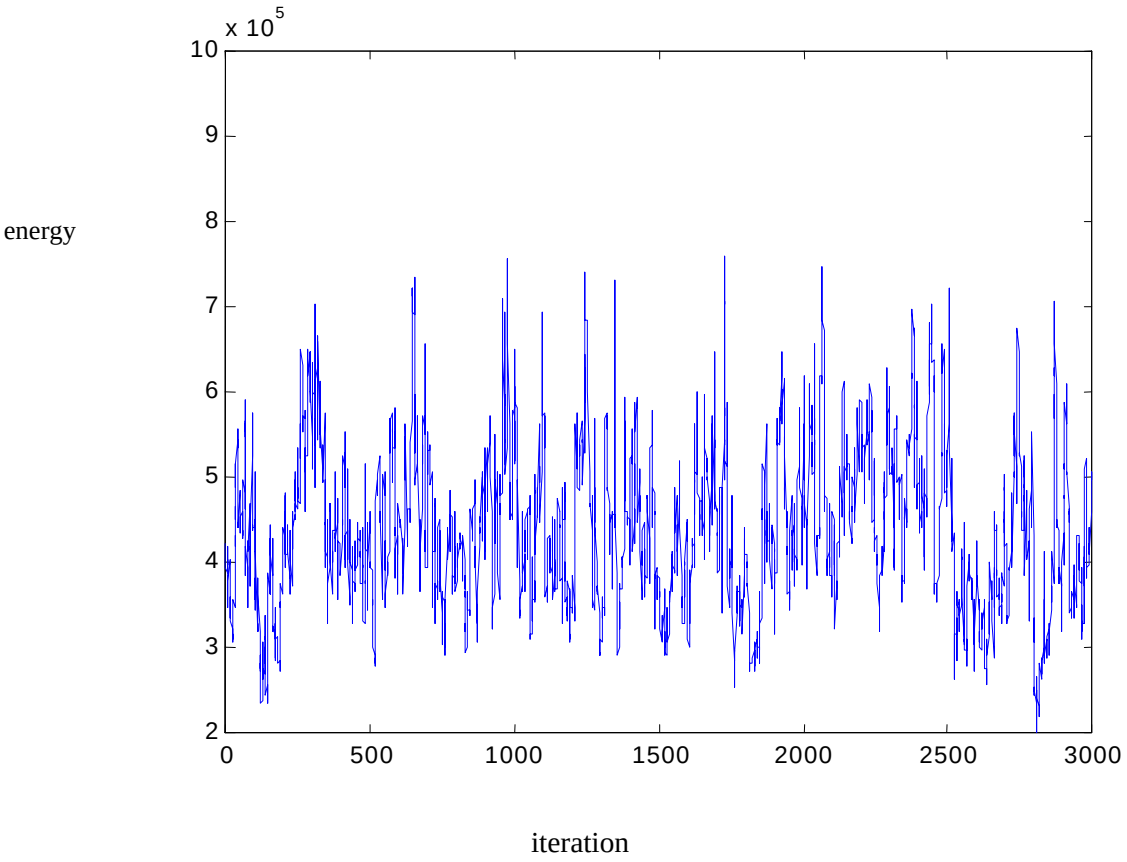

Figure 2. Energy versus time-series plot of a random walk on a layout space.

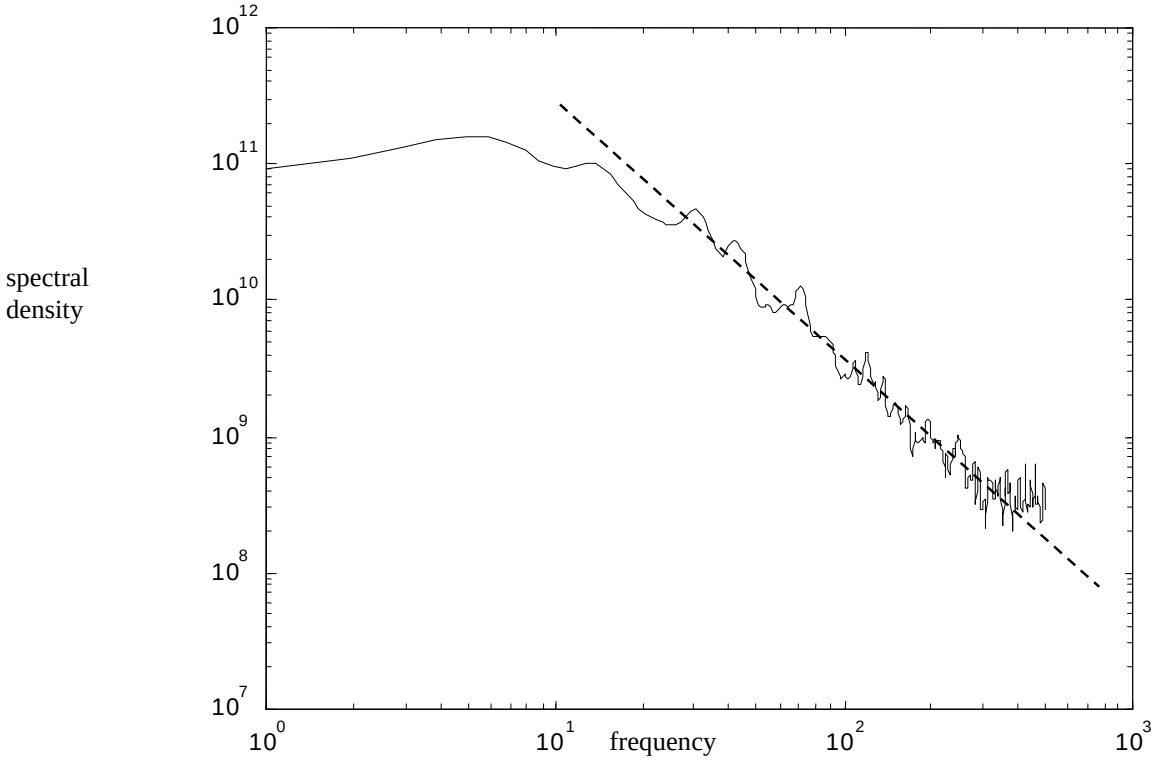

Figure 3. Spectral density of the layout space energy time-series.

The spectral density analysis is used to characterize the time correlations of sampling points from the random walk. It can be shown that the spectral density (Figure 3) of a random walk on a 3-D layout space has spectral energy which is power-law in frequency, as is reflected by the trend of a logarithmic slope (shown by the dashed line) of $\frac{1}{f^2}$ holding down to the frequency of about 10Hz. Since a random walk is a fractional Brownian motion only if the walk steps are significantly smaller than the dimension of the space, the power-law relation applies only for the frequencies above a certain point. The fractalness of the layout space is thus inferred by the satisfaction of the power-law relation.

As a comparison, a random walk is taken on the design space of a linear objective function (a 45° line) and its energy time-series and spectral density plots are shown in Figure 4 and Figure 5 respectively. The characteristics of the random walk resembles those of a white noise since the energy time-series is uncorrelated from point to point and thus its spectral density turns out to be

flat, representing equal energy for all frequencies. In contrast, the energy time-series of a random walk on a 3-D layout space consists of more slow (low frequency) than fast (high frequency) fluctuations (Figure 3), and the power-law relation as shown in Eq. 1 is satisfied.

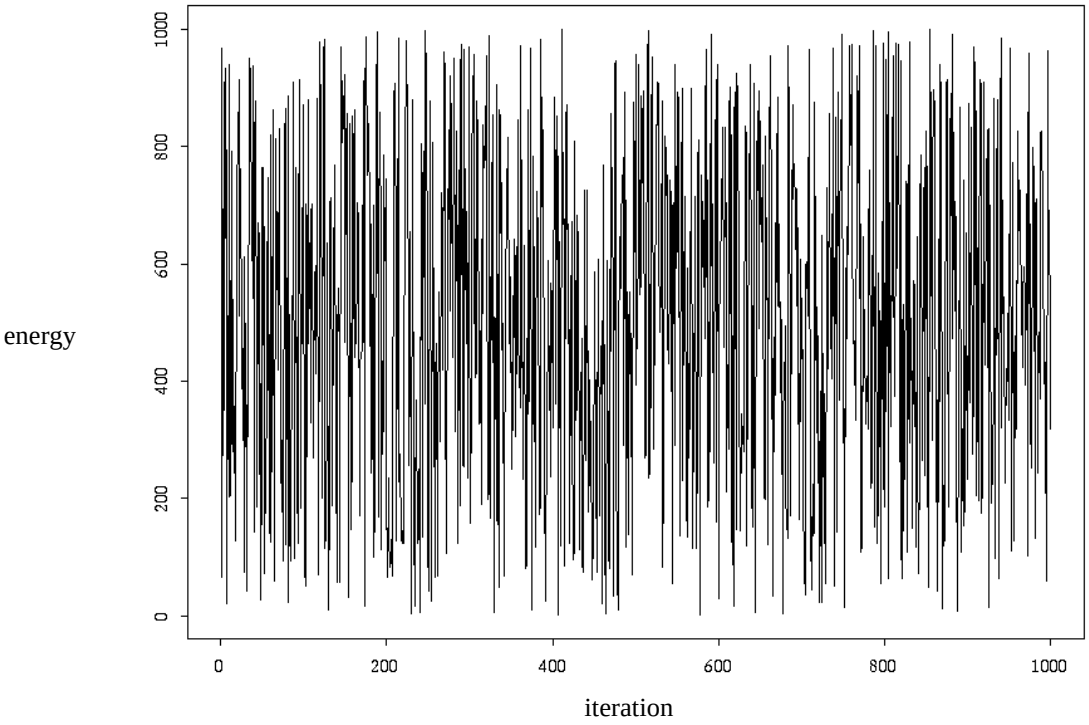

Figure 4. Energy versus time-series plot of a random walk on a linear space.

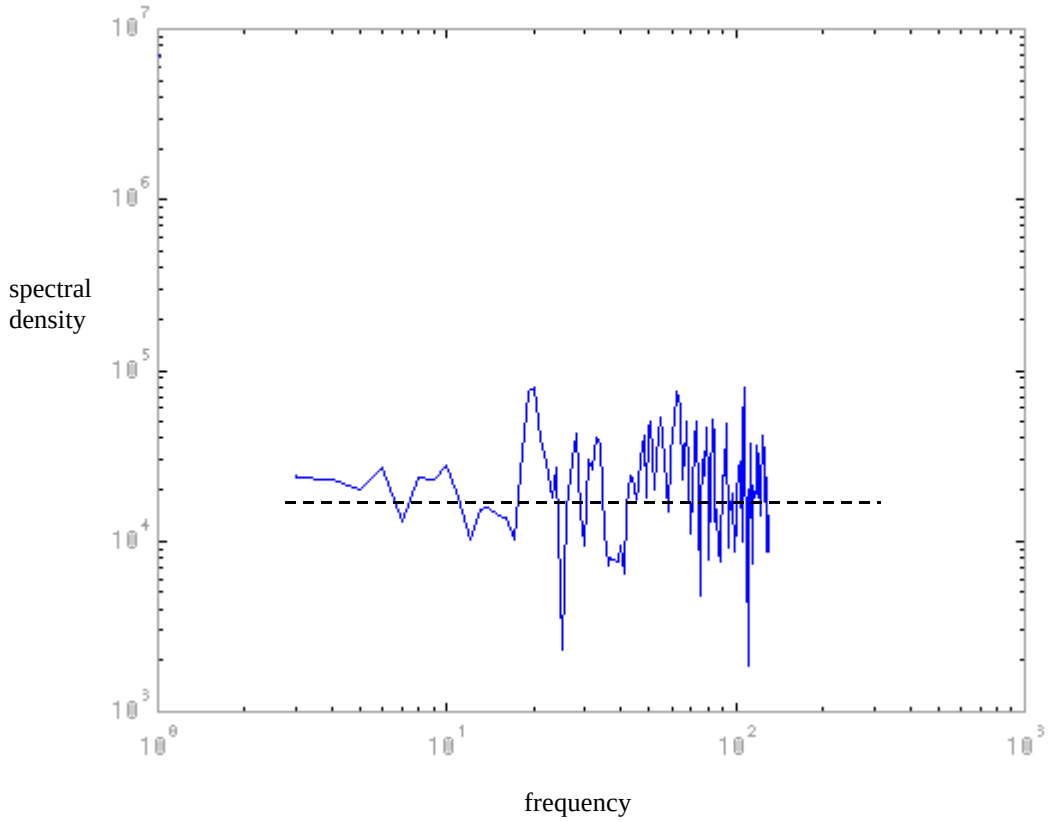

Figure 5. Spectral density of the linear space energy time-series.

The layout space, then, as represented by the 8-cube problem, is clearly characterized as fractal-like. The implications are that deterministic, downhill search algorithms will generally converge to inferior minima. Thus stochastic algorithms are required to solve the general layout problem.

2.2 A Continuum of Search Algorithms

A continuum is shown in Figure 6 along which a variety of optimization search algorithms are placed according to the amount of randomness each method possesses, with deterministic methods such as gradient method (GM) on the one end, random search (RS) on the other. It is often necessary for an algorithm to have some degrees of randomness to escape from inferior local optima because of the fractalness of the layout space. However, too much randomness can make the search exhaustive and the convergence difficult to achieve. There is a trade-off between the amount of computing time invested and the quality of solutions obtained. While the amount of randomness appropriate for an algorithm is problem-dependent and related to the complexity of a design space and the expected quality of the design, the desired place for an algorithm on the

continuum should be somewhere close to the deterministic end, but have some stochastic characteristics. An arrow is placed in Figure 6 to represent a suitable place for the layout algorithm, which is not far from deterministic algorithms to ensure efficiency, but at the same time has stochastic elements introduced to enable global exploration.

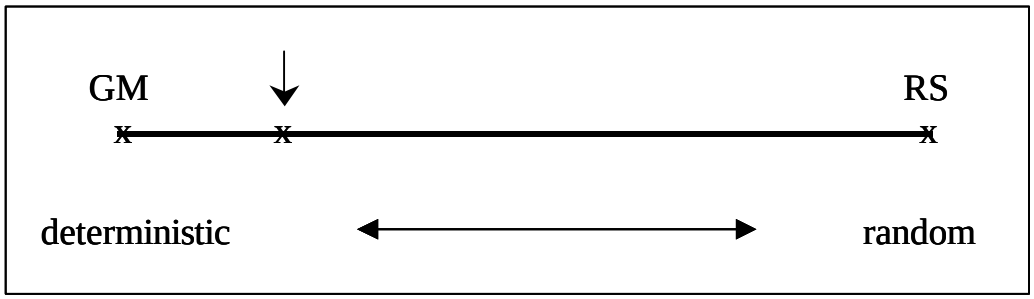

Figure 6. A brief continuum of layout search algorithms.

Various algorithms used in the layout problem are discussed in the next section. The pros and cons of each algorithm are summarized and the corresponding positions on the continuum are characterized.

3 Literature Survey

The layout problem can have different formulations, but it is usually abstracted as an optimization problem. An assignment of the coordinates and orientations of components that minimizes the cost and satisfies certain placement requirements is sought. The problem can be viewed as a generalization of the quadratic assignment problem and therefore belongs to the class of NP-hard problems (De Bont, et al., 1988). Consequently it is highly unlikely that exact solution to the general layout problem can be obtained in an amount of time that is bounded by a polynomial in the size of the problem, resulting in prohibitive computation time for large problems. Heuristic algorithms are typically used to generate acceptable solutions.

Much of the literature concerns with simpler two-dimensional rectangular layout and three-dimensional cuboid layout. Circuit board layout and glass or metal cutting applications are examples of 2-D rectangular layout. Shipping container stuffing and vehicle loading are typical 3-D cuboid layout problems.

While it is possible to solve rectangular layout problems with a small number of objects of a few sizes using linear programming or branch and bound approaches (Dyckhoff 1990), the combinatorial and geometrical complexity makes it hard to obtain optimal solutions to the general

3-D layout problem efficiently. Certain approximations of arbitrary shapes of objects may be necessary, and heuristics are often used to guide the search for good solutions.

Recent research efforts have led to several approaches for the layout of 3-D objects of complex geometry. The genetic algorithm approaches (Ikonen et al. 1997), simulated annealing approaches (Kolli et al. 1996, Cagan et al. 1998), a hybrid approach using a combination of simulated annealing and expert systems (Hills and Smith, 1997), and an extended pattern search approach (Yin and Cagan, 2000a; Yin and Cagan, 2000b) are among the ones that have shown promise.

This section reviews the related work in layout optimization, with a focus on the following two key aspects: 1) the optimization algorithms used for the exploration of the design space and the identification of good solutions; and 2) the geometric representation for 3-D components of complex shapes and the interference evaluation techniques.

3.1 Layout Search Algorithms

A variety of optimization algorithms have been applied to the layout problem. Some of the approaches may be efficient for specific types of problems, but often place restrictions on component geometry, allowable degrees-of-freedom, and the objective function formulation. Others are applicable to a wider variety of problems but may require prohibitively long computing time to solve even simplistic problems. Layout algorithms can be classified into different categories according to search strategies used for design space exploration. Heuristic rule-based algorithms, traditional optimization algorithms, genetic algorithms, simulated annealing algorithms, extended pattern search algorithms, and hybrid algorithms are discussed in this section. The emphasis is on the mechanical and electro-mechanical applications. The topic presented in this section corresponds to the building block of the optimization search algorithm shown in Figure 1.

3.1.1 Heuristic Rule-Based Approaches

Heuristic rule-based algorithms are often used in operations research to solve packing problems. Dowsland and Dowsland (1992) presented a survey of packing problems. The basis for heuristic rule-based algorithms is a set of rules to determine efficient packing of boxes into containers without any additional restrictions. Since the solution space is large, it is important that the search for good solutions is not so exhaustive as to require inordinate amount of computing time.

Heuristic rules are often derived from common sense or experiences and they can provide insights into the mechanisms behind efficient packing. For example, practical constraints such as load stability and ease of loading can be incorporated into the rules instead of being checked after the completion of the packing.

Although heuristics can be fairly simple, the range of possible adjustments that may be utilized in order to provide improved packing can be considerable. It is clearly not possible to try all variations of all placement rules and it is difficult to know which are most likely to lead to effective packing for a specific problem. Since heuristics are likely to be domain-dependent, it is important to match one's requirement to the heuristic's capabilities.

Wang (1983) presented an approach to two-dimensional rectangular packing by successively "gluing" together pairs of rectangles to produce a set of feasible subsolutions. For the non-rectangular packing, the geometric complexity of placing the pieces directly onto the stock sheet is generally prohibitive. Adamowicz and Albano (1976) and Israni and Sanders (1985) proposed an approach to first nest the pieces into regular modules.

The wall-building approaches (George and Robinson, 1980, Bischoff and Marriott, 1990) are the common methods to deal with 3-D cuboid packing problems. Sections of a container across the full width and height are packed first. Identical items are grouped together to develop layers. An ordering of boxes based on decreasing volume (Gehring et al., 1990) is used to develop layers.

Dai et al. (1994) proposed a heuristic algorithm for the generation of three-dimensional non-cuboid packing. An octree representation (Dai and Cha, 1994a) was used to approximate the geometry of the components. The packing algorithm is based on the idea of matching the octree nodes to identify the proper order and orientation of the components. The objects are packed into the container sequentially and only rotations of multiples of 90 degrees are allowed. The method is quite effective in cases where the total volume of packed items is much smaller than that of the container but not for problems with tight packing.

Heuristic algorithms are deterministic methods that are efficient for specific types of problems. However, it is difficult to generate rules applicable for components of irregular shapes or optimize objectives other than the packing density. These algorithms are typically deterministic, falling to the left on the continuum of Figure 6.

3.1.2 Traditional Optimization Approaches

Traditional optimization algorithms, such as branch & bound methods, linear programming methods and gradient-based algorithms, can be used to solve a narrow class of packing problems efficiently.

Scheithauer and Terno (1995) presented a branch & bound algorithm to compute optimal solutions for instances of the one-dimensional cutting stock problem, where the objective function and constraints are linear functions of the design variables. Beasley (1985) proposed a linear optimization method with 0/1-variables for the two-dimensional non-guillotine cutting problem of rectangular objects using a Lagrangean relaxation and a subgradient iteration to compute upper bounds. Only translations of the objects are allowed in these methods. Models for the general cases where rotations are used are not linear. For packaging applications, except some simpler problems (such as the 1-D bin-packing problem), it can be challenging to find a feasible solution to start with. The performance of the algorithm degrades as the problem size increases.

Gradient-based algorithms use gradient information in seeking the optimal solutions. By calculating the gradient of the objective function and then searching in the negative direction, we can limit the search to a specific direction. Since the objective function is often nonlinear, a new direction at a new state needs to be calculated and the process to be repeated a number of times. The gradient information provides some insight into choosing reasonable search directions. In the cases where analytical gradient information is unavailable, finite-difference approximations may be used.

Landon and Balling (1994) used gradient-based methods to place and orient 3-D solid objects within containers according to spatial and mass property criteria. The gradients of the mass properties are calculated by differentiating the appropriate formula with respect to the design variable being considered. The objects are modeled using a boundary representation. The minimum separation distance between a pair of objects is found by searching for the closest two points within the two respective solids. The interference distance is calculated as the distance by which an object needs to translate away from the other object in a specified direction until the two objects barely touch. Both distances are computed by solving a constrained optimization problem. Newton's method is used to return to the constraint boundary.

Kim and Gossard (1991) formulated the packaging task as a constrained optimization problem. The objective function is an aggregate "energy" function composed of packaging goals

and penalty functions. Packaging goals can be described as spatial relationships among components. Two common spatial relationships are *near* and *far* relationships. They can be conceptualized as a translational spring attached to the two objects. The near relationship tends to move an object toward another object, and the far relationship holds when the spring behaves like a repulsive spring. The near relationship is useful for a compact package and far relationship is applicable when an object may be influenced by proximity to another object such as through heat transfer or electromagnetic interference. The coplanar, coaxial and parallel relationships are represented as equality constraints. This formulation of the packaging problem results in an underdetermined system containing more variables than equations. Variables that are not determined by equality equations are computed through energy minimization.

In Kim and Gossard's work, the solid modeler uses a dual representation of the objects: a boundary representation and a constructive solid geometry (CSG) tree. The boundary representation is useful for specification of constraints and the CSG tree's half-space representation specifies a solid by unions and intersections of multiple half-spaces. The interference between objects is approximated by the translation necessary to eliminate the interference. The non-interference constraint as an energy function can be minimized but may not be avoided entirely. When the number of objects is large and spatial relationships are complex, the computational problems of convergence and efficiency may arise and the constraint management can be difficult.

Gradient-based algorithms are deterministic methods (to the far left on the continuum shown in Figure 6) that can significantly reduce the computation time required to converge to an optimal solution by limiting search to promising directions. However, the solution is only the nearest local optimum with respect to the initial design. Multiple runs from different starting points may be necessary because of the fractalness of the layout space. When explicit gradients are not available, which is the case for many layout problems, forward difference approximations may not be accurate and thus may have misleading effects on the search.

3.1.3 Genetic Algorithms

Genetic algorithms (GAs) are search algorithms based upon natural selection. The first step in using GAs is to code the problem space. The design variables are mapped into chromosomes by a fixed length string of symbols. It is assumed that each individual string represents a unique point in the search space. In each iteration of the search process, a population of strings that represent a family of the current possible solutions is maintained. The selection, mutation and

crossover operators are used to create the new generation of solutions. A fitness function evaluates the designs and decides which will be the survivors into the next generation. Selection is accomplished by copying strings from the last generation into the new generation based on a fitness function value. Mutation is the process of randomly changing one bit of information in the string and it prevents GAs from stagnating during the solution process. Crossover is responsible for introducing most new solutions by selecting two parent strings at random and exchanging parts of the strings.

GAs have been used in VLSI placement and routing (Schnecke and Vornberger, 1996; Cohoon and Paris, 1986) and two-and-a-half dimensional cuboid packing (Wodziak and Fadel, 1994). Dighe and Jakiela (1995) used GAs to pack 2D polygons by "dropping" them into a container. Ikonen et al. (1997) used GAs to pack 3-D non-convex objects into a cylindrical container for rapid prototyping. In his approach, three pieces of information are represented: 1) the order in which parts are placed; 2) the orientation of each part; and 3) how parts spatially relate to each other. Each chromosome is a list of three sublists of integers. The first sublist is a permutation of part numbers being packed. The second sublist is the orientation list, which indicates the rotation of each part. To reduce the search space, the number of allowed rotations is restricted to 45-degree increments around each coordinate axis. The third sublist is the list of pre-defined attachment points on the previous part in the permutation sublist the current part is attached to. Crossovers are used between the same type of sublists and a higher mutation rate is used for the orientation list to prevent some orientation values leading to a non-promising packing from being removed from the gene pool too soon. The fitness function uses a penalty-function method where infeasible solutions are given a penalty to indicate how much the defined constraints are violated. The calculation of intersection between objects is based on the intersection of triangles on the tessellated surfaces of the objects. To reduce the number of triangles to be checked in the intersection calculation, the bounding boxes of the objects and triangles are checked first for intersection.

The search space for layout is highly discontinuous and multi-modal. How well a genetic algorithm does on such a space depends on the evaluation function used and how well it is able to differentiate between promising and poor solutions. It has been observed that even a small change in the weight factors of the evaluation function can make a big difference to the results a genetic algorithm is able to obtain. A small change in a chromosome can also make a great difference in the packing solution. For example, layouts with objects slightly disoriented from a good packing plan can produce a very low score. Also, it is not easy to choose good mutation and crossover

rates to make the algorithm converge properly. The intersection calculation based on pairs of triangles is accurate, but may be very slow when the objects are of complex shape and thus the number of surface triangles is numerous. Further, the process would be more complicated if arbitrary rotations of objects were considered.

GAs are stochastic algorithms that have a lot of randomness built in; thus they are placed close to the random end on the continuum. They are zero-order algorithms that require function values only. They are reliable and can deal effectively with non-smooth and discontinuous functions. The price paid for this generality is that these methods often require a large number of function evaluations to achieve an optimum, even for the simplest of problems. Therefore, they are considered most useful for problems in which the function evaluation is not computationally expensive. The strength of GAs is their robustness, which is mainly caused by the fact that they deal with a sample of candidate solutions at a time and utilize probabilistic transition rules. However, the coding and decoding processes can be very long if the number of objects is large. Also, there are often limitations in the processes to make the search space manageable. The quality of solutions is dependent on the weight factors in the evaluation function and the relative ratio of the selection, mutation and crossover operators, however it is not obvious how to choose the values that can lead to a good solution for a specific problem.

3.1.4 Simulated Annealing Algorithms

Simulated annealing (Kirkpatrick *et al*. 1983) is a generally applicable stochastic technique based on the analogy between simulating the metallurgical annealing process and solving large combinatorial optimization problems. Within the algorithm an initial design state is chosen and the value of the objective function for that state is evaluated. A step is taken to a new state by applying a move, or operator, from an available move set. This new state is evaluated; if the step leads to an improvement in the objective function, the new design is accepted and becomes the current design state. If the step leads to an inferior state, the step 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:

$$P_{accept} = e^{-\frac{\ddot{A}C}{T}}, \qquad (2)$$

where *DC* 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, steps taken 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 steps decreases, those steps tend to get rejected, allowing the algorithm to converge to an optimum once promising areas of the objective function space have been found.

Applications of SA algorithms require specifications of three distinct items (De Bont et al. 1988): 1) a concise problem representation; 2) a transition mechanism; and 3) a cooling schedule. The problem representation consists of a configuration representation and an expression for the cost function. The cost function represents the cost effectiveness of different layouts. The transition mechanism generates a new configuration from a current one. The difference in cost between the two configurations must be calculated and a decision is made whether or not the new configuration is to be accepted. The cooling schedule is used to control the temperature in the algorithm, and specify the starting value, decrement function, length of generation and the stopping criterion.

Simulated annealing algorithms have shown great success in circuit layout (Sechen, 1988; Wong, et al., 1988; Rutenbar, 1989; Hustin and Sangiovanni-Vincentelli, 1987) and significant work in the mechanical component layout area was motivated by the circuit layout technology. Jajodia, et al., (1992) presented a solution to inter-cell and intra-cell layout problems using SA, which addressed the relative placement of equal-dimensional manufacturing entities within a discrete solution space in an attempt to minimize the total material flow between these entities. The performance of the SA algorithm was compared to other facility layout methods and shown to yield either equal or better quality for each of a set of classical test problems. The quality of the SA approach is insensitive to the initial starting states.

Cagan (1994) presented an approach using a combination of shape grammars (Stiny, 1980) and simulated annealing to solve the constrained geometric knapsack problem. A 2-D packing of half-hexagons into knapsacks of various shapes is solved in polynomial time and space complexity.

Szykman and Cagan (1995, 1997) extended the technology from 2-D VLSI to 3-D mechanical and electro-mechanical layout. Their approach can deal with blocks and cylinders with rotations constrained to multiples of 90 degrees. A perturbation-based approach was used in which infeasible states with component overlap and constraint violations were allowed and penalized. The move set includes translation, rotation and swap moves. An adaptive annealing

schedule (Huang et al., 1986) was used to control the temperature, and a probabilistic move selection strategy (Hustin and Sangiovanni-Vincentelli, 1987) was used to choose moves based on their prior performance. The objective function consists of a weighted sum of multiple performance measures. Several types of spatial constraints that are characteristic of layout problems are defined. These include the global and relative constraints on component locations and orientations.

An integrated approach to 3-D layout and routing was introduced in Szykman and Cagan (1996) and Szykman et al. (1998). Routing problems are abundant in engineering applications such as routing of pipes, wires and air ducts. The routing cost can be influential in the manufacturing of certain products such as HVAC (Heat, Ventilation and Air Conditioning) products. Taking the routing cost into account during the component placement stage of the layout could significantly improve the quality and reduce the cost of the product layout.

The specification of a routing task consists of locations of a pair of terminals that must be connected for each route. The routing moves include adding, removing, or relocating a bend. The adding move selects a route and inserts a new bend at a random location. The remove move selects a route and randomly deletes one of its bends. The relocation move changes the location of a bend by moving it along a direction by some distance. The layout and routing are carried out concurrently. In addition to the layout cost terms, the routing cost (route length and number of bends) and the component-route intersection penalty terms are included in the objective function. The experiments showed that concurrent layout and routing results in superior solutions over the typical layout-then-route approach.

Kolli et al. (1996) extended the work of Szykman and Cagan by relaxing the restrictions on component geometry and rotations. The component geometry can be imported from commercially available CAD packages and octree representations are used for quick interference evaluation between components of complex shapes. The octree resolution level to be used at a particular stage of the annealing algorithm can be adjusted to make the algorithm more efficient. At higher temperature, the annealing process essentially performs a random walk through the design space and hence does not require a very accurate estimate of the objective function. At low temperatures the probability of accepting an inferior state is much lower and only moves that lead to a better state are likely to be accepted, thus more accurate evaluations are necessary. Several test problems are solved, which include the cube packing and cogwheel packing problems. Cagan et al. (1998) further extended the work to include constraint satisfactions.

Simulated annealing is a general method for a wide variety of combinatorial problems. The method is flexible as well as powerful. Sorkin (1991) has revealed that the behavior of simulated annealing depends heavily on the "energy landscape" associated with the optimization problem. The success of annealing relies on the overall energy difference of collections of states being large compared with the barriers dividing these collections. A large number of evaluations are necessary for SA to converge to good solutions. When the objective function is complex, the computation can be expensive and time-consuming. Certain approximations of the actual analyses may be necessary in order to obtain solutions in reasonable time. Campbell et al. (1997) used a hierarchical heat transfer analysis to reduce the computational time of the placement of heat generating electronic components.

SA algorithms are essentially random during the initial stages of the search and become more deterministic as the temperature decreases. SA algorithms can be placed near to but to the left of GAs on the continuum.

3.1.5 Extended Pattern Search Algorithms

Pattern direct search algorithms are a subset of direct search algorithms introduced by Hooke and Jeeves (1961). Direct search methods are defined as the sequential examination of trial solutions involving comparison of each trial solution with the "best" obtained to that time together with a strategy for determining (as a function of earlier results) what the next trial solution will be. Torczon and Trosset (1997) surveyed the history of pattern search methods and provided some practical suggestions for utilizing them. The search algorithms follow a series of exploratory moves defined by pattern matrices to walk through the design space and search for a stationary point. They rely exclusively on direct comparisons of function values during the search and thus provide a tool suitable for the exploration of a design space that is nonlinear and discontinuous, as is the case of the layout problem.

A basic pattern search method proceeds as follows: An initial state is chosen, and the objective function for that state is evaluated. A step is taken to a new state by applying a move along a pattern direction by a specified step length. The objective function is evaluated again at the new state. The two evaluations are compared, and the better one is chosen; the corresponding state is accepted as the current state. The search continues from the current state along the next pattern direction, following the same criteria of the new state acceptance. After an iteration of explorations along all the pattern directions, the step length is either carried into the next iteration if at least one previous move has led to a better new state, or scaled by a factor that is less than *1*.

The search stops when the step length is smaller than a pre-specified tolerance. Individual pattern search methods are characterized by the choices of exploratory moves by which the search directions and step size are defined.

Pattern search methods exhibit several attractive features that suggest they can be the methods of choice for nonlinear and non-smooth optimization problems. First, they are gradient-related methods, but they do not rely on the evaluation of derivatives. This is desirable for the cases where derivatives are either unavailable or unreliable. Secondly, pattern search methods have good global behavior: a stationary point can be located by starting from an arbitrary initial point. Finally, pattern search methods are straightforward and easy to use, which makes implementation and parameter tuning a simple task.

Compared to the probabilistic hill-climbing methods of simulated annealing, pattern search methods explore the design space in a more restrictive manner. The moves are allowed only along the pattern directions. The step sizes are updated according to certain rules, with large steps used early in the search and scaled down gradually during the search. These enable pattern search methods to converge with fewer evaluations than simulated annealing does. But the greedy search methods of pattern search cannot prevent the solution from being trapped in the inferior local optima. Extensions need to be introduced to make the algorithm stochastic and enable it to escape from inferior optima for a better design.

Five extensions (Yin and Cagan, 2000a) were introduced to the basic pattern search algorithm to address the characteristics of the layout problem and help convergence to good quality solutions. The extensions include: randomized search orders, constraint related search directions, occasionally allowed step-jumps, strategically used swapping moves, and judiciously chosen hierarchical models. The extended pattern search layout algorithm was applied to a series of test cases and industrial applications (Yin et al., 1999) and shown one-to-two orders of magnitude improvement in speed over a robust SA algorithm for solutions of equivalent quality. Various pattern search heuristics have been incorporated into the algorithm to guide the search along promising directions (Yin and Cagan, 2000b).

The extended pattern search algorithm moves components through the design space, evaluates the design states, and makes decisions about the exploratory moves. The moves are allowed only along certain directions defined by a pattern direction set. The pattern directions can be updated according to the effectiveness of previous moves. The translation moves and rotation

moves are interlaced. The swap moves are used only if a loop of translations and rotations at a given step size fails to generate any improved state, creating a better chance for the search to continue from a better design state. The order of selecting components for exploratory moves and the search directions to move along is randomized, making the sequence of the moves less deterministic to avoid getting trapped into certain local minima. The step size starts to be large and is generally decreased over time. The search stops when a pre-specified minimal step size is reached. Step jumps may be used to allow occasional increase in the step size that essentially combines the local and global explorations and makes the search more exhaustive by delaying the satisfaction of stop criteria. New design states are accepted as the current states whenever an improvement in the objective function occurs, and the original state is retained if no such improvement is found.

The extended pattern search algorithm is more deterministic than SA algorithms in terms of the move direction, step size, and new state acceptance. However, it has stochastic elements that help navigate the fractal space. It places no restrictions on objective function formulation and thus is a generally applicable method, found left of center on the continuum.

3.1.6 Hybrid Approaches

Hybrid approaches take advantage of two or more search algorithms and combine them in the problem solving. Dai and Cha (1994b) presented a hybrid approach of heuristic rules and neural network algorithms to solve the two-dimensional rectangular layout. The 2-D packing problem is mapped to a Hopfield neural network in which the heights and widths of the rectangles represented by neurons and the unused area represented by an energy function. The heuristic rules are based on the algorithm proposed by Coffman and Short (1990) to decide the packing order and the orientation of the rectangles. Dai and Cha's hybrid method was compared to three other approaches: a pure heuristic method, a mixed discrete optimization method, and a simulated annealing method. The pure heuristic algorithm has high stability, needs little computing time, and is very robust in finding feasible solutions. However, it rarely gets superior quality solutions. The performance of the mixed discrete optimization algorithm is poor and time-consuming because of the existence of a large number of local minima for the packing problem. The simulated annealing algorithm has the potential to get arbitrarily close to a global optimum, but is terribly slow (note, however, that the authors did not discuss the type of annealing schedule they had used; our conjecture is that they used a simple vanilla annealing schedule that would not perform well). The hybrid method can find a good solution in reasonable time. The disadvantages

of using artificial neural networks include their strict demands on the format of the energy function and the difficulty in network parameter adjustment.

Smith et al. (1996) described an application using a combination of a simulated annealing method and a knowledge-based system technique for spatial layout. Conflicting requirements such as the usage of space, routings and adjacencies need to be negotiated in the design. The knowledge-based system represents the design engineer's expertise to formulate problem specifications and evaluate candidate solutions. The SA algorithm is used to generate initial layout configurations for later manipulation by the knowledge-based system. A cellular decomposition of the entire placement space is used. Because of this grid-like layout, non-orthogonal rotations and translations of non-unit factor are not permitted, limiting the permissible packing density.

The hybrid methods are placed around the center on the continuum: the neural network algorithm and the SA algorithm introduce randomness into the layout process while the heuristic rules and the knowledge system make the methods less stochastic.

3.1.7 Summary

Various layout algorithms are reviewed in this section. The continuum appeared in Figure 6 is shown again in Figure 7, with the positions for different algorithms marked.

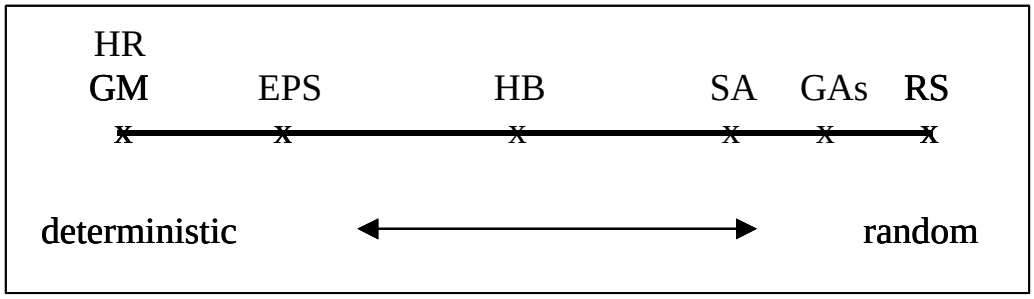

Figure 7. A detailed continuum of the layout search algorithms.

As a recapitulation, heuristic rule (HR) based algorithms and gradient methods (GM) are deterministic methods that are efficient for specific types of problems, but the solutions are likely to be inferior local optima. Simulated annealing algorithms (SA) and genetic algorithms (GAs) are stochastic methods that are applicable to a wide variety of problems. However, the large amount of randomness makes the methods inefficient. Hybrid methods (HB) use stochastic

methods to navigate the space and heuristic rules or knowledge-based systems to reduce the search space, thus they are placed in the center on the continuum. The extended pattern search (EPS) algorithm incorporates stochastic elements into an otherwise deterministic pattern search method and thus is placed towards the deterministic end on the continuum.

3.2 Geometric Representation and Interference Detection

For the general three-dimensional layout/packaging problem, components can be of complex geometry and the container space may not be substantially larger than the combined spatial occupation of the components to be contained. For better exploration of the search space, it is often desirable to allow objects to move through each other and penalize the degree of overlap to drive the design into a feasible region. Since the interference calculation is performed in each iteration and a large number of iterations are necessary for convergence to a good solution, it is vital to choose an appropriate geometric representation scheme and an efficient intersection detection algorithm.

Interference detection is a key issue to many applications such as computer animation, virtual reality and robotic path planning. The solution strategy for this difficult problem has to be decided considering a trade-off between accuracy and speed. In this section, the generic representation schemes of 3-D geometry that are amenable in the layout and packaging applications are reviewed. These include the boundary representation, the constructive solid geometry, and the bounding volume and multi-resolution models. The topic in this section belongs to the component representation, which is a part of the building block of the components in Figure 1.

3.2.1 Boundary Representation

A boundary model represents the boundary surface of a three-dimensional object as a collection of geometric entities such as vertices, edges, and faces. The data structure is based on graphs that represent connectivity of the entities. The interference detection is through the pair-wise intersection tests between lines and faces on the tessellated boundary surfaces. Boundary models are the most popular internal representation of three-dimensional geometry in commercial solid modeling systems. However, we often need to convert a boundary model to another representation suitable for efficient intersection detection. The popular boundary representations include vertex-based polygon models, edge-based boundary models, and winged-edge models.

The simplest representation of a solid surface is a vertex-based polygonal model consisting of lists of vertices and polygonal faces. If a face has holes, it has to be subdivided into a set of simpler polygons with no holes. This representation has been widely used in many computer graphics file formats. It stores an unordered list of vertices and an unordered list of faces, and each face is represented as an ordered list of boundary vertices. It is important that boundary vertices are ordered consistently, i.e., either counter-clockwise or clockwise, so that an intersection detection algorithm can tell which side of a face is outside. This vertex-based model, the simplest representation of solid objects, gives sufficient information to calculate intersections between two solids. In the latest version of the virtual reality modeling language, VRML2, collision detection is automatically performed between two solids represented using IndexedFaceSet, which is essentially a vertex-based polygon model.

A vertex-based polygonal model does not have edge entities explicitly in its data structure. This is because we assume that all the edges are straight lines and that all the faces are planar faces. If these assumptions do not apply, edges should be explicitly represented in the data structure so that the geometric information of a curved edge can be associated with each edge entity in the database.

An edge-based boundary model consists of: 1) an unordered list of vertices, 2) an unordered list of edges, each of which has an ordered pair of two end vertices, and 3) an unordered list of faces, each of which has an ordered list of edges.

While the vertex-based models and edge-based models are sufficient and convenient for a simple rendering system, higher level geometric operations such as Boolean set operations and intersection detection can benefit from more information on the topological connectivity among the geometric entities. Although this same information can be derived from a vertex-based polygon model or an edge-based boundary mode, if the information is required repeatedly in performing geometric operations we can save computational time for the operations by explicitly representing this information on the topological connectivity.

How much topological information should be represented? If we represent too much information, maintaining and updating such information can become a computational bottleneck. The optimal choice of how much information should be stored in the data structure depends on the types of geometric operations we perform, but in many geometric operations like intersection

detection the so-called “winged-edge” data structure and its variations are known to be good choices.

Baumgart first introduced the winged-edge data structure in his Ph.D. thesis (Baumgart, 1974) as a convenient data structure for polyhedral objects for computer vision. It is a kind of edge-based data structure, but each edge entity has more pointers to: 1) the two end vertices; 2) four adjacent edges; and 3) two faces that share the edge.

3.2.2 Constructive Solid Geometry Representation

In constructive solid geometry (CSG), a solid is represented as a combination of primitives, or building blocks, such as cuboid, cones, cylinders, spheres, tori, prisms, and so on. The motivation for decomposing a component into a set of primitives is that the intersection tests between the simple-shaped primitives are much faster than the tests between the complex original geometry. Three essential elements of CSG besides primitives are: 1) rigid transformation, 2) regularized Boolean set operations, and 3) binary tree structures (Mortenson, 1997).

Primitives are first scaled based on specified dimensions, then transformed by a rigid motion, or a combination of translations and rotations, and finally merged together by regularized Boolean set operations: 1) union, 2) intersection, 3) difference, and 4) complement. Because the final geometry changes depending on the order of the operations performed, this order of operations is stored in a binary tree structure. CSG is conceptually straightforward and many solid modeling systems use the representation as one of the modeling user interfaces.

3.2.3 Bounding Volume and Multi-Resolution Representation

The exact interference calculations using the boundary representation or CSG representation may not be computationally efficient since the number of pairs of primitives on the tessellated surfaces or primitives in CSG representation can be huge if the components are of complex geometry and in close proximity. The bounding volume and multi-resolution models are often used to approximate the actual geometry and perform the intersection detection quickly without significant compromise in accuracy in order to solve the problem in reasonable time. Further, hierarchical models work well in stochastic layout methods in that more accurate interference analysis can be done at lower levels of the hierarchy while rough analysis useful in the early stages of these algorithms can be done at higher levels. The most often used bounding volume and multi-resolution representations include sphere tree, octree, and OBB tree.

The key to the sphere tree representation is that the symmetries of an object around its skeleton suggest ways to place spheres for a tight approximation. Hubbard (1995) used sphere hierarchies to compute time-critical intersection detection in a virtual environment. The first step in the sphere-tree construction is to build the medial-axis surface (Goldak et al., 1991) or skeleton of the object. The method distributes points over the surface of the object and builds Voronoi diagrams for the points. The corners of the Voronoi cells define the centers of spheres that closely fit the object. Repeatedly merging adjacent pairs of spheres reduces the number of spheres for a desired level of detail. For each pair of objects whose bounding spheres have intersection, the intersection detection method descends the pair's hierarchies and tests only child spheres whose parents intersect.

Using spheres as the basis of approximation (Hubbard, 1996; Palmer and Grimsdale, 1995) works well in a wide range of situations. Spheres are rotationally invariant, which makes the testing computationally efficient. However, broad, flat objects may pose problems in that many spheres are necessary to approximate such objects. Also, the pre-process that builds the medial-axis surface is difficult to implement and slow to run.

Octrees represent a solid using spatial-occupancy enumeration (Mortenson, 1997). The representation starts with an axial-aligned bounding box of the original object, and the box is recursively subdivided into eight octants. If any of the resulting octants is full or empty, there is no need to subdivide it further. The partially full octants are recursively subdivided until the resulting octants are full or empty, or until a pre-specified level of resolution is reached.

The cost of testing a pair of octants for interference is small and the degree of interference is easy to quantify, which is desirable for fast interference evaluation. However, for some geometry, such as a set of long-thin oriented polygons, it may take more levels of octrees to fit the object tightly.

An oriented bounding box (OBB) tree (Barequet, et al., 1996) is a rectangular bounding box at an arbitrary orientation. The primary motivation for using OBB is that, by virtue of their variable orientation, they can bound a geometry more tightly than octrees and sphere trees. Therefore, fewer levels of an OBB tree need to be traversed to process the collision detection for objects in close proximity. Gottschalk et al. (1996) developed efficient algorithms for computing the hierarchies of tight-fitting OBBs for unstructured models and for the rapid checking of the overlap between two OBB trees.

The placement of a tight fitting OBB around a collection of polygons makes use of first and second order statistics summarizing the vertex coordinates. Two of the three eigenvectors of the covariance matrix are used to determine the axes along which the box is aligned. After the tight-fitting OBBs are computed, they are represented hierarchically using a top-down approach, which subdivides the longest axis of a box and partitions the polygons according to the splitting axis. The overlap test between two OBBs uses a separating axis theorem, which projects the boxes onto a set of axes. An axis is called a separating axis if the intervals by the projection of two boxes do not overlap. Two objects are disjoint if at least one separating axis exists. OBBs are efficient for detecting interference, but it is difficult to quantify the degree of interference.

It is clear that no hierarchical representation gives the best performance all the time. The total cost of interference detection varies considerably with the relative placement of the objects. The choice of the primitive shapes to construct a hierarchical tree is governed by two conflicting constraints. On the one hand, the model should fit the original object as tightly as possible. On the other hand, testing two such models for overlap should be as fast as possible. Simple primitives like spheres and octrees do very well with respect to the second constraint. However, they cannot fit long, thin objects tightly. OBBs provide tight fits, but checking for overlap between them is more expensive. Generally speaking, when objects are far apart, hierarchical representations based on spheres and octrees work well in practice, while OBBs are more computationally efficient when two objects are in close proximity.

4 Conclusions

Various optimization search algorithms and their representative layout implementations are summarized in this paper. Heuristic rule-based approaches and traditional optimization techniques can be used to solve a narrow class of layout problems efficiently, but they are not suitable for problems with nonlinear, non-differentiable objective and constraint functions. Stochastic algorithms such as GA and SA algorithms are applicable to the general layout problem. However, a large number of iterations may be necessary to achieve good convergence and the computation can be expensive if the objective function evaluation is time-consuming.

How much randomness should a layout algorithm contain? On the one hand, it is beneficial to make use of the gradient information and go along with the downhill directions until an optimal solution is found. On the other hand, it is very unlikely that a solution so found is of good quality because of the multi-modal and fractal-like design space. A proper balancing of deterministic and stochastic search techniques is necessary for the efficient and effective

exploration of the layout space. The extended pattern search algorithm demonstrated substantial time-savings over a robust SA algorithm while obtaining good quality solutions through such a balancing. The algorithm achieves efficiency by reducing the number of search directions to a minimal size while still maintaining a sufficiently rich set of search directions to capture the direction of steepest descent. The step size adjustment and the swap moves allow some "jumping around" in the layout space to address its fractal characteristics. Domain-specific knowledge and heuristics can be incorporated into the algorithm to help reduce the search space and improve efficiency.

Geometric representation and interference detection are an important part of the design evaluation for the layout of 3-D components of complex geometry. While there is no single representation that performs best for all cases of interference tests in the layout/packaging problem, it is recommended to consider the following observations in choosing a geometry representation most suitable for a specific problem:

- In virtually all cases, the usage of some kind of bounding volumes will help reduce the computational cost of interference tests;

- Unless a component's geometry is identical to that of the bounding volume, box or sphere, use hierarchical versions of bounding volumes;

- If a component's geometry has a strong directionality, or is aligned to the three orthogonal axes for example, as in the case of most automobile components, use octrees for computational efficiency and representation simplicity;

- If a component's geometry is arbitrarily shaped and its orientation is not axis-aligned, use oriented, hierarchical bounding volumes, such as OBB trees and spherical trees;

- When arbitrarily shaped objects need to be tightly packed, it is more efficient to use polygonal models in addition to bounding volumes. A rough interference test should follow using bounding objects, while a detailed interference test should be performed using a polygonal model;

- The levels of detail of polygon models should be decided based on how tightly the objects need to be packed.

Regardless of the types of geometric representations chosen for specific applications, the corresponding interference testing routines can be interchangeable for a general layout algorithm.

A great deal of research efforts has been devoted to the development of computational layout approaches and technology is now available for the automated layout synthesis. Future research directions include the incorporation of domain-specific knowledge into the layout algorithm to make the problem solving process more efficient and the creation of seamless interfaces with solid modelers and analysis packages to facilitate the use of the layout tools in practice.

References

Adamowicz, M., and Albano, A., 1976, "Nesting Two-Dimensional Shapes in Rectangular Modules," *Computer Aided Design*, **8**, 27-33.

Barequet, G., Chazelle, B., Guibas, L., Mitchell, J.S.B., and Tal, A., 1996, "BOXTREE: A Hierarchical Representation for Surfaces in 3D," *Proc. Eurographics '96*, pp. 387-484.

Baumgart, B.G., 1974, *Geometric Modeling for Computer Vision*, Ph.D. thesis, Stanford University.

Beasley, J.E., 1985, "An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure," *Operational Research*, **33**, 49-65.

Bischoff, E.E. and Marriott, M.D., 1990, "A Comparative Evaluation of Heuristics for Container Loading," *European Journal of Operational Research*, **44**(2), 267-276.

Cagan, J., Degentesh, D., and Yin, S., 1998, "A Simulated Annealing-Based Algorithm Using Hierarchical Models for General Three-Dimensional Component Layout," *Computer Aided Design*, Vol.30, No. 10, 781-790.

Campbell, M.I., Amon, C.H, and Cagan, J., 1997, "Optimal Three-Dimensional Placement of Heat Generating Electronic Components," *ASME Journal of Electronic Packaging*, **119**(2): 106-113.

Coffman, E.G. Jr., and Short, P.W., 1990, "Average-Case Analysis of Cutting and Packing in Two Dimensions," *European Journal of Operational Research*, **44**, 134-144.

Cohoon, J.P., and Paris, W.D., 1986, "Genetic Placement," *Proc. of IEEE Int. Conf. On CAD,* pp. 422-425.

Dai, Z., Cha, J., and Yuan, J., 1994, "An Octree Based Heuristic Algorithm for 3-D Packing," *Advances in Design Automation 1994: Proceedings of the 20th ASME Design Automation Conference*, **2**:125-133.

Dai, Z., and Cha, J., 1994a, "An Octree Method for Interference Detection in Computer Aided 3-D Packing," *Advances in Design Automation 1994: Proceedings of the 20th ASME Design Automation Conference*, **1**:29-33.

Dai, Z., and Cha, J., 1994b, "A Hybrid Approach of Heuristic and Neural Network for Packing Problems," *Advances in Design Automation 1994: Proceedings of the 20th ASME Design Automation Conference,* **2**:117-123.

De Bont, F.M.J., Aarts, E.H.L., Meehan, P. and O'Brien. C.G., 1988, "Placement of Shapeable Blocks," *Philips Journal of Research*, **43**:1-22.

Dighe, R., Jakiela, M. J., 1995, "Solving Pattern Nesting Problems with Genetic Algorithms Employing Task Decomposition and Contact Detection," *Evolutionary Computation*, Cambridge, Massachusetts, MIT Press, **3**(3):239-266.

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

Dowsland, W.B., 1991, "Three-Dimensional Packing – Solution Approaches and Heuristic Development," *International Journal of Production Research,* **8**:1673-1685.

Dyckhoff, H., 1990, "A Typology of Cutting and Packing Problems," *European Journal of Operational Research* **44**, 145-59.

Gehring, H., Menschner, K. and Meyer, M., 1990, "A Computer-Based Heuristic for Packing Pooled Shipment Containers," *European Journal of Operational Research,* **44** (2), 277-289.

George, J.A., and Robinson, D.F., 1980, " A Heuristic for Packing Boxes into a Container," *Computers and Operational Research,* **7**:147-156.

Goldak, J.A., Yu, X., Knight, A., and Dong, L., 1991, “Constructing Discrete Medial Axis of 3-D Objects,” *International Journal of Computational Geometry and Applications*, 1(3):327-339.

Gottschalk, S., Lin, M.C., and Manocha, D., 1996, “OBBTree: A Hierarchical Structure for Rapid Interference Detection,” *Proc. SIGGRAPH ’96*, pp. 171-180.

Hills, W., and Smith, N., 1997, "A New Approach to Spatial Layout Design in Complex Engineered Products," *Proceedings of the International Conference on Engineering Design (ICED 97)*, Tampere, Finland, August 19-21.

Hooke, R., and Jeeves, T.A., 1961, “Direct Search Solutioin of Numerical and Statistical Problems,” *Journal of the Association for Computing Machinery*, **8**(2):212-229.

Hopfield, J.J. and Tank, D.W., 1985, “Neural Computations of Decisions in Optimization Problems”, *Biol. Cybern.*, **52**, 141-152.

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.

Hubbard, P.M., 1995, “Real-Time Collision Detection and Time-Critical Computing,” *Workshop on Simulation and Interaction in Virtual Environment*.

Hubbard, P.M., 1996, “Approximating Polyhedra with Spheres for Time-Critical Collision Detection,” *ACM Transactions on Graphics*, **15**(3):179-210.

Hustin, M.D. 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*.

Ikonen, I., Biles, W., Kumar, A., Ragade, R.K., and Wissel, J.C., 1997, “A Genetic Algorithm for Packing Three-Dimensional Non-Convex Objects Having Cavities and Holes,” *Proceedings of 7th International Conference on Genetic Algorithms*.

Israni, S. and Sanders, J.L., 1985, “ Performance Testing of Rectangular Parts-Nesting Heuristics,” *International Journal of Production Research*, **23**, 437-456.

Jajodia, S., Minis, I., Harhalakis, G., and Proth, J.M., 1992, "CLASS: Computerized LAyout Solutions Using Simulated Annealing," *International Journal of Production Research*, **30**(1):95-108.

Kim, J.J., and Gossard, D.C., 1991, "Reasoning on the Location of Components for Assembly Packaging," *Journal of Mechanical Design*, **113**: 402-407.

Kirkpatrick, S., Gelatt, C.D., Jr., and Vecchi, M.P., 1983, "Optimization by Simulated Annealing," *Science*, **220** (4598): 671-679.

Kolli, A., Cagan, J., and Rutenbar, R.A., 1996, "Packing of Generic, Three Dimensional Components Based on Multi-Resolution Modeling," *Proceedings of the 22nd ASME Design Automation Conference* (DAC-1479), Irvine, CA, August 19-22.

Landon, M.D. and Balling, R.J., 1994, "Optimal Packaging of Complex Parametric Solids According to Mass Property Criteria," *Journal of Mechanical Design*, **116**:375-381.

Mortenson, M.E., 1997, *Geometric Modeling*, John Wiley & Sons, Inc.

Palmer, I.J. and Grimsdale, R.L., 1995, "Collision Detection for Animation Using Sphere-Trees," *Computer Graphics Forum*, **14**(2):105-116.

Peitgen, H.O. and Saupe, D., 1988, *The Science of Fractal Images*, Springer-Verlag, New York.

Rutenbar, R.A., 1989, "Simulated Annealing Algorithms: An Overview," *IEEE Circuits and Devices Magazine*, Vol. 5, No. 1, pp. 19-26.

Scheithauer, G. and Terno, J., 1995, "A Branch & Bound Algorithm for Solving One-Dimensional Cutting Stock Problems Exactly," *Aplicationes Mathematicae*,**23**:2, 151-167.

Schnecke, V., and Vornberger, O., 1996, "An Adaptive Parallel Genetic Algorithm for VLSI-Layout Optimization," *4th Int. Conf. on Parallel Problem Solving from Nature*.

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

Smith, N., Hills, W. and Cleland, G., 1996, "A Layout Design System for Complex Made-to-Order Products," *Journal of Engineering Design*, Vol. 7, No. 4, 363-375.

Sorkin, G.B., 1991, “Efficient Simulated Annealing on Fractal Energy Landscapes,” *Algorithmica*, **6**:367-418.

Sorkin, G.B., 1992, *Theory and Practice of Simulated Annealing on Special Energy Landscapes*, Ph.D Thesis, University of California at Berkeley.

Szykman, S., 1995, *Optimal Product Layout Using Simulated Annealing*, Ph.D Thesis, Carnegie Mellon University.

Szykman, S., and Cagan, J., 1995, “A Simulated Annealing Approach to Three-Dimensional Component Packing,” *ASME Journal of Mechanical Design*, **117**(2A):308-314.

Szykman, S., and Cagan, J., 1996, "Synthesis of Optimal Non-Orthogonal Routes,” *ASME Journal of Mechanical Design*, **118**(3):419-424.

Szykman, S., and Cagan, J., 1997, "Constrained Three Dimensional Component Layout Using Simulated Annealing,” *ASME Journal of Mechanical Design*, **119**(1):28-35.

Szykman, S., Cagan, J., and Weisser, P., 1998, "An Integrated Approach to Optimal Three Dimensional Layout and Routing,” *ASME Journal of Mechanical Design*, **120**(3):510-512.

Talukdar, S.N. and deSouza, P., 1994, “Insects, Fish and Computer-Based Super-Agents,” *Systems and Control Theory for Power Systems*, Vol. 64.

Torczon, V., 1992, “PDS: Direct Search Methods for Unconstrained Optimization on Either Sequential or Parallel Machines,” *CRPC Technical Report: CRPC-TR92206*, Center for Research on Parallel Computing Rice University, Houston, TX

Torczon, V., 1997, “On the Convergence of Pattern Search Methods,” *SIAM Journal on Optimization*, **7**(1):1-25.

Torczon, V. and Trosset, M., 1997, “From Evolutionary Operation to Parallel Direct Search: Pattern Search Algorithms for Numerical Optimization,” *Computing Science and Statistics*, Vol. 29.

Wang, P.Y. (1983), “Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems”, *Operational Research* 31, 573-586.

Wodziak, J.R. and Fadel, G.M., 1994, “Packing and Optimizing the Center of Gravity Location Using a Genetic Algorithm,” *Journal of Computers in Industry*.

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

Yin, S., and Cagan, J., 2000a, “An Extended Pattern Search Algorithm for Three-Dimensional Component Layout,” ASME *Journal of Mechanical Design*, **122**(1):102-108.

Yin, S., and Cagan, J., 2000b, “Exploring the Effectiveness of Various Patterns in an Extended Pattern Search Layout Algorithm,” to appear in *Proceedings of the 2000 Design Engineering Technical Conferences*, ASME, DETC2000/DAC-14254, Baltimore, MD, September 10-13.

Yin, S., Cagan, J., Hodges, P., and Li, X., 1999, “Layout of an Automobile Transmission Using Three-Dimensional Shapeable Components,” submitted to *ASME Journal of Mechanical Design*, also in *Proceedings of the 1999 Design Engineering Technical Conferences*, ASME, DETC99/DAC-8564, Las Vegas, NV, September 12-15.