ELSEVIER

Computer-Aided Design 34 (2002) 597–611

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

A survey of computational approaches to three-dimensional layout problems

J. Cagan*, K. Shimada, S. Yin

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

Received 4 August 2000; revised 1 January 2001; accepted 9 February 2001

Abstract

The component layout or packaging problem requires efficient search of large, discontinuous spaces. This survey paper reviews the state-of-the-art in product layout algorithms. The focus on optimization and geometric interference calculation strategies addresses the common aspects of the layout problem for all applications. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Intelligent packaging; Product layout; Optimization; Stochastic optimization

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 3-D 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.

The difficulty in automating the mechanical and electromechanical 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.

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 non-linear and multi-modal, making it vital to identify a suitable algorithm to navigate the space and find good quality solutions.

Fig. 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

* Corresponding author. Tel.: +1-412-268-3713; fax: +1-412-268-3348.
E-mail address: cagan@cmu.edu (J. Cagan).

PII: S0010-4485(01)00109-9

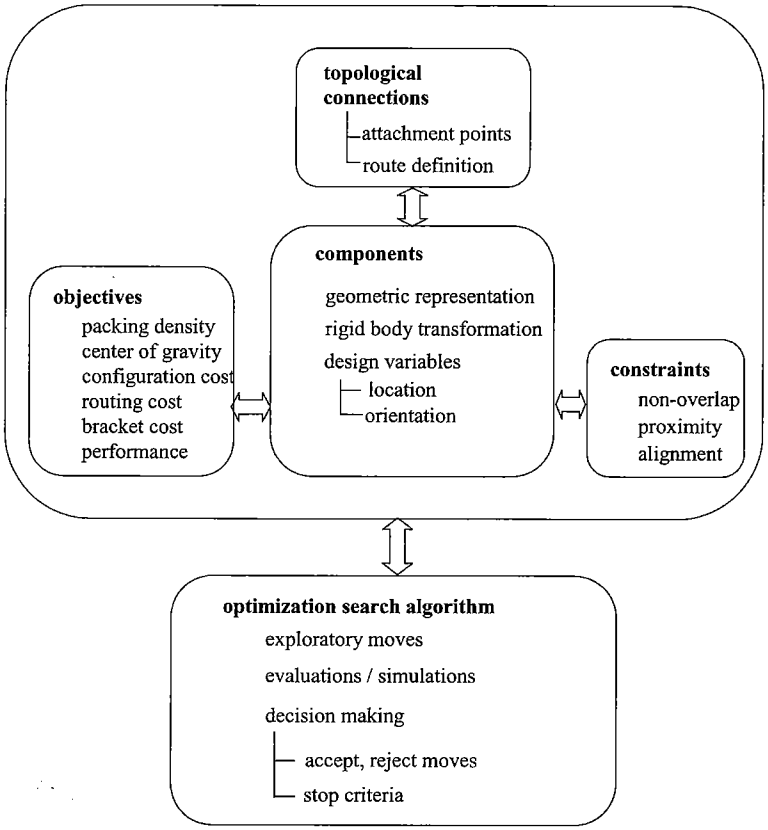

Fig. 1. Major constituent parts for generic layout synthesis.

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 3-D 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. The *layout space* is defined as the mathematical representation of the space of configurations mapped against the cost per configuration. 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 [45,46]. 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 [39]. 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

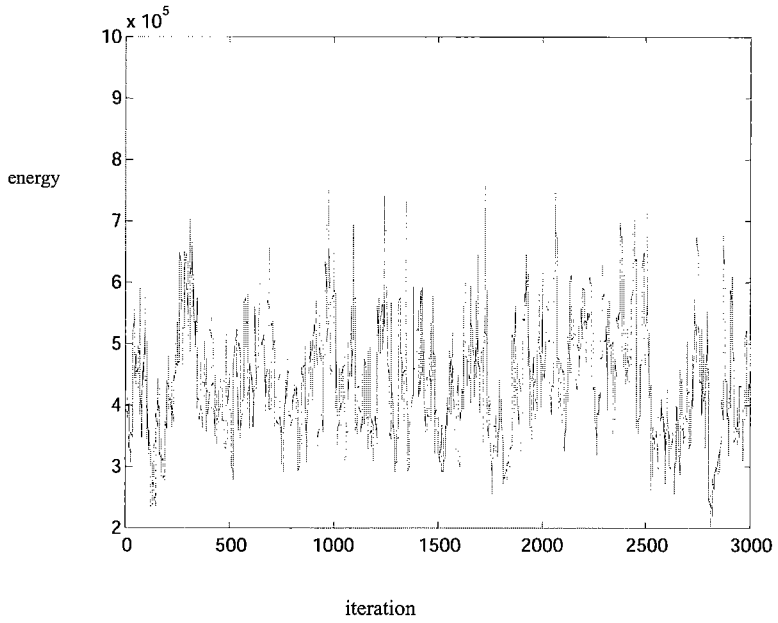

Fig. 2. Energy vs time-series plot of a random walk on a layout space.

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:

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

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'. Eq. (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 [45]. A random walk is taken on the 8-cube packing problem, where eight identical cubes are packed into a container of eight times the size of the individual cubes [6]. A random walk is a sequence of points $X(t)$ $(t = 0,1,2,\ldots)$ where $X(t+1)$ is generated by a random move from $X(t)$. The walk is meaningful in that the any two sequential configurations are differentiated by a single move from the available set of moves. 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 Fig. 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 Fig. 3.

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 (Fig. 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 $1/f^2$ holding down to the frequency of about 10 Hz. 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. Because the plot in Fig. 3 is of negative slope and relatively low standard deviation, it can be said that the layout space is 'fractal-like'.

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 Figs. 4 and 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 (Fig. 3), and the power-law relation as shown in Eq. (1) is satisfied.

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 Fig. 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

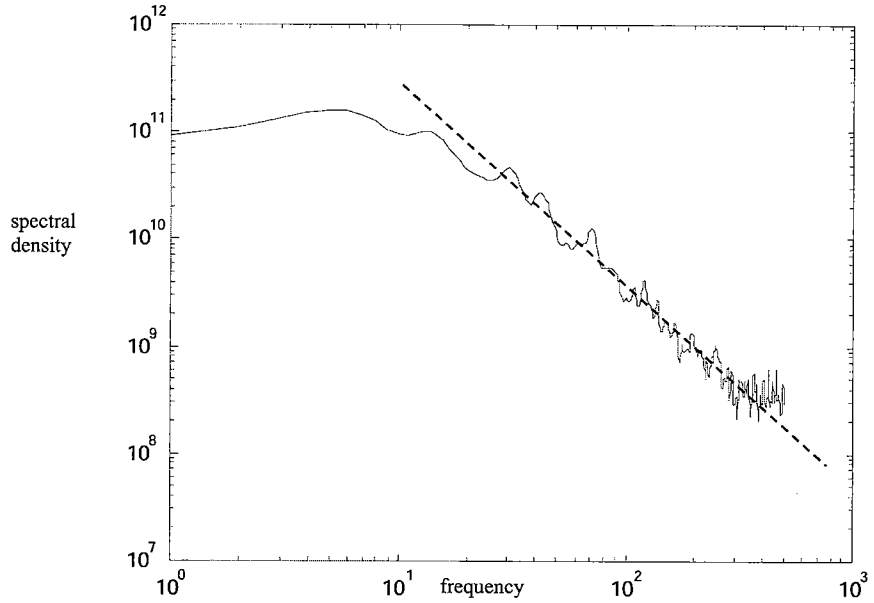

Fig. 3. Spectral density of the layout space energy time-series.

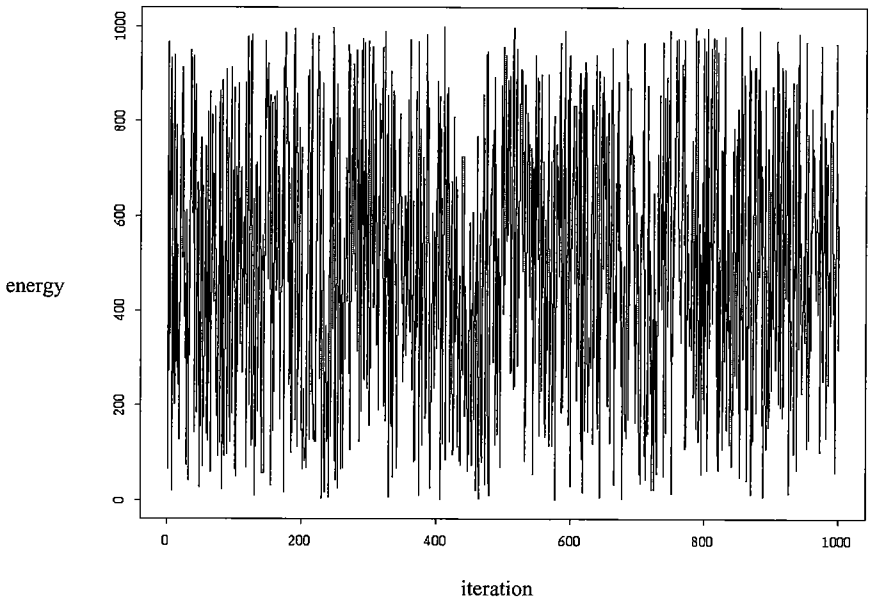

Fig. 4. Energy vs time-series plot of a random walk on a linear space.

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 Fig. 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.

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

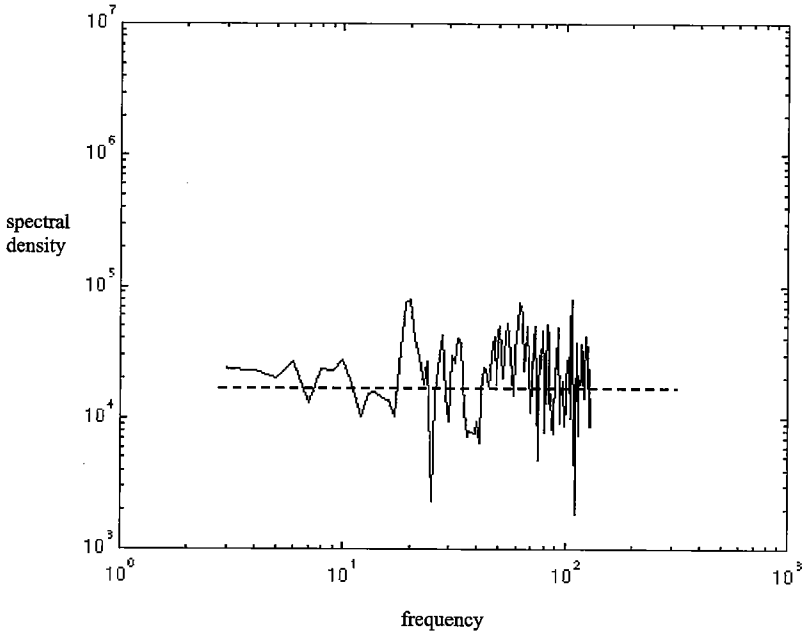

Fig. 5. Spectral density of the linear space energy time-series.

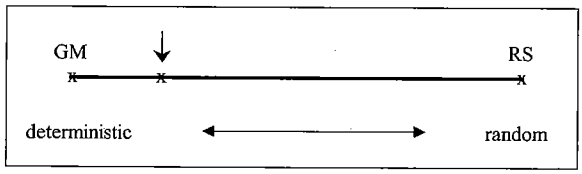

Fig. 6. A brief continuum of layout search algorithms.

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 [13]. 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. Layout strategies extend the core optimization technique by identifying acceptable moves within the configuration space. As will be discussed, general algorithms typically require some level of (stochastic) perturbation to avoid local optima.

Much of the literature concerns simpler two-dimensional rectangular layout and 3-D 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 [17], 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 [29], simulated annealing approaches [6,34], a hybrid approach using a combination of simulated annealing and expert systems [22], and an extended pattern search approach [59,60] 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 Fig. 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 [15] 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 an 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 [16]. 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 [56] 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 [1] and Israni and Sanders [30] proposed an approach to first nest the pieces into regular modules.

The wall-building approaches [5,19] 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 [18] is used to develop layers.

Dai et al. [10] proposed a heuristic algorithm for the generation of 3-D non-cuboid packing. An octree representation [11] 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° 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 Fig. 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 [41] 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 [4] proposed a linear optimization method with 0/1-variables for the two-dimensional non-guillotine cutting problem of rectangular objects using a Lagrangian 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 non-linear, 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 [35] 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 [32] 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 Fig. 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 [9,42] and two-and-a-half dimensional cuboid packing [57]. Dighe and Jakiela [14] used GAs to pack 2D polygons by 'dropping' them into a container. Ikonen et al. [29] 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° 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 [33] 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:

$$P_{\text{accept}} = e^{-\frac{\Delta C}{T}}, \tag{2}$$

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, 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 [13]: (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 [28,40,43,58] and significant work in the mechanical component layout area was motivated by the circuit layout technology. Jajodia et al. [31] 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 [62] presented an approach using a combination of shape grammars [63] 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 [48,50] 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°. A perturbation-based approach was used in which infeasible states with component overlap and constraint violations were allowed and penalized [47]. The move set includes translation, rotation and swap moves. An adaptive annealing schedule [25] was used to control the temperature, and a probabilistic move selection strategy [28] 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 [49] and Szykman et al. [51]. 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. [34] 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. [6] 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 [45] 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. [7] 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 [23]. 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 [55] 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 non-linear 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 prespecified 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 [59] 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 [61] 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 [60].

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 [12] 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 [8] 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. [44] 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 Fig. 6 is shown again in Fig. 7, with the positions for different algorithms marked. 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.

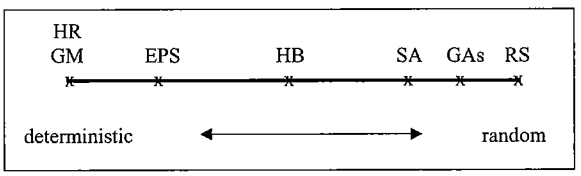

Fig. 7. A detailed continuum of the layout search algorithms.

3.2. Geometric representation and interference detection

For the general 3-D 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 is necessary for convergence to a good solution, it is vital to choose an appropriate geometric representation scheme and an efficient intersection detection algorithm. This section describes the component representation that is one of the component attributes shown in Fig. 1.

Interference detection is a key issue in many other application areas such as computer-aided design and manufacturing (CAD/CAM), computer animation, virtual reality and robotic path planning. The solution strategy for this difficult problem has to be decided considering the specific requirements of each application. In this section, we review three generic representation schemes of 3-D geometry that are most commonly used in layout and packaging applications: (1) boundary representation; (2) constructive solid geometry representation; and (3) bounding volume and multi-resolution representation. Note that this does not cover all possible geometric representations. For example, we omitted parametric surface and implicit surface representations because these representations can be and should be converted by boundary representations, bounding volume, or multi-resolution representation before being utilized in a 3-D layout system. While such an approximation is not suitable in exact interference detection, required for a geometric modeling system, it is acceptable in most practical layout problems because a certain amount of tolerance, or gaps between packed components, is required anyway. For a more comprehensive survey on collision detection methods including other types of geometric representations and on public domain software for collision detection, see Lin and Gottschalk's survey paper [36].

3.2.1. Boundary representation

A boundary model represents the boundary surface of a 3-D 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 3-D 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 PhD thesis [3] 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.