ELSEVIER

PII: S0010-4485(98)00036-0

Computer-Aided Design, Vol. 30, No. 10, pp. 781–790, 1998

Printed in Great Britain
0010-4485/98/$19.00+0.00

A simulated annealing-based algorithm using hierarchical models for general three-dimensional component layout

Jonathan Cagan*, Drew Degentesh and Su Yin

An efficient simulated annealing-based algorithm which optimizes component layout is presented. The efficiency comes in the algorithm's ability to calculate component overlap quickly by taking advantage of a hierarchical decomposition of the model's geometry. The result is an algorithm able to optimize the placement of components of arbitrary geometry inside an arbitrarily shaped container over multiple design goals and subject to inter-component spatial and performance constraints. The algorithm is demonstrated on test problems of known solution, and a variety of industrial problems of *a priori* unknown solution.

Keywords: simulated annealing, layout, configuration, octree

INTRODUCTION

Of the various aspects of design that have been considered for automation, the layout of components within their container has received surprisingly little attention. Component placement in the mechanical and electro-mechanical world is typically done through physical prototyping and placement by hand. Significant time and costs are associated with physically making the models. Further, the negotiation of the often conflicting spatial relations between the components during placement becomes a difficult, error-prone, labor-intensive, and time-consuming task. To date there has been no successful, general approach to computer-assisted layout of arbitrary 3-D components.

We define the general 3-D component layout problem as (adapted from Szykman and Cagan[1] and Kolli *et al.*[2]):

> Given a set of three-dimensional objects of arbitrary geometry and an available space (possibly the space of a container), find a placement for the objects within the space that achieves the design objectives, such that none of the objects interferes (i.e. occupy the same space), while satisfying optional spatial and performance constraints on the objects.

Component layout plays an important role in the design and usability of many engineering products. Engineering artifacts of today are becoming increasingly complicated, yet continuous competitive pressure drives a reduction in design cycle time: consumers prefer compact designs with more features; manufacturers require less time and resources spent in generating design layouts.

In the electrical world there exist significant literature and commercial tools that place and route hundreds to millions of components. Here, components are limited to rectangles aligned in 2-D along orthogonal axes. These computational design tools employ simulated annealing[3] to automate analog and digital circuit layout tasks (e.g. References[4–8]).

Szykman and Cagan[1,9,10] extended the technology from the 2-D electrical world to the 3-D mechanical and electro-mechanical world. They presented an approach able to take components, a container, and spatial constraints on the components with respect to the container and other components, and generate alternative layout configurations. Although theoretically general, in practice their work was limited to simple geometric shapes (i.e. blocks and cylinders), rotations that are multiples of 90° and constraints specified along the principal axes. Kolli *et al.*[2] proposed an extension to the work of Szykman and Cagan which relaxed the limited shapes and rotations for packing problems (i.e. problems without inter-component spatial constraints). They introduced the concept of using hierarchical models to represent the component and container geometries at various levels of resolution compatible with probabilities of acceptance of a simulated annealing-based packing algorithm.

In this paper we present a general extension to the above-mentioned work for layout and packing problems, introducing and demonstrating a simulated annealing-based layout algorithm able to generate consistently good quality layouts for arbitrary geometries. These layouts not only optimize one or more objectives such as packing density or assemblability, but also satisfy spatial constraints between the components and the container and the components with

*To whom correspondence should be addressed. Tel.: 412 268 3713; Fax: 412 268 3348; E-mail:jon.cagan@cmu.edu
Computational Design Lab, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
Paper Received: 30 July 1997. Revised: 26 April 1998. Accepted: 4 May 1998

respect to each other. The algorithm and implementation are readily extendible to other objectives and performance analysis; one example will show layout results in conjunction with a tube routing algorithm introduced by Szykman and Cagan[10]. After a brief review of related efforts in 3-D layout, the general simulated annealing-based algorithm will be described and demonstrated on test problems and realistic design problems.

RELATED WORK

A variety of non-linear programming techniques have been applied to 3-D component layout problems.[11–15] Component layout/packing problems have also been approached using genetic algorithms.[16,17] However, most of these approaches place restrictions on component geometry, orientations and problem complexity.

Beyond these efforts, most of the research on packing problems has been done in the field of operations research (OR). The various OR approaches to packing problems are summarized in survey papers.[18–20] However, a drawback with these methods is that they are generally heuristic methods that are only applicable to a specific class of problems, such as cutting parts from stock of fixed width or packing different sized rectangles into bins of fixed dimensions. Another limitation is that the majority of problems considered in the OR literature involves the use of relatively simple geometries, such as rectangles or blocks[20] and often in two dimensions. Thus, most of these methods do not apply to problems with components or containers with complex geometries, nor do they easily incorporate objectives other than maximizing packing density.

Simulated annealing does not share these shortcomings, and has been used to address the layout and packing problems. Cagan[21] used a simulated annealing-based technique to solve a geometrically constrained knapsack problem, and Kämpke[22] reports a solution to a bin-packing test problem using simulated annealing with results superior to previously published solutions. Simulated annealing has also been used on several classical test problems for other types of layout problems, such as facilities layout.[23] As mentioned, some successful approaches to the electrical layout problem have included simulated annealing. Such work directed the efforts by Szykman and Cagan and Kolli *et al.*, which in turn motivated the current effort.

Smith *et al.*[24] and Hills and Smith[25] presented work in spatial engineering for made-to-order products such as offshore oil rigs. Their efforts also used simulated annealing to produce initial layout configurations for later manipulation by human intervention and knowledge-based systems. They used a cellular decomposition of the entire placement space, modeling components within numerical matrices, with each matrix member representing a unit volume of the placement space. Because of this grid-like layout, non-orthogonal rotations and translations of non-unit factor were not permitted, limiting the permissible packing density.

Dai and Cha[26,27] presented an octree-based heuristic algorithm that could generate packings of 3-D components of somewhat arbitrary geometry. Their approach is a greedy algorithm that places components sequentially in a container. However, their approach is limited in the objective function chosen, the restriction that components can only rotate in multiples of 90°, the quality of solutions with highly concave geometries, and the need to have a substantially larger volume in the packing container than the parts being packed.

In this paper we describe an approach to solving the general 3-D layout optimization problem. Our work introduces a multi-resolution geometric analysis which, in conjunction with simulated annealing, generates solutions to the generic 3-D layout problem, with components of arbitrary geometry and orientation, in reasonable time. The resulting code has been tested on a variety of examples in industrial settings.

THREE-DIMENSIONAL LAYOUT OPTIMIZATION BY SIMULATED ANNEALING

The 3-D layout space is non-linear, combinatorial and discontinuous, making it difficult to find an algorithm that is both efficient and capable of finding the global, or at least very good local, optimum. Since traditional downhill search algorithms are unable to navigate such spaces, stochastic optimization techniques are required; we have adapted simulated annealing as the search method of choice based on its history in PCB and chip layout. In our approach, the annealing moves components around within a defined space (possibly a container), analyzing each move on its effectiveness toward minimizing a combined objective function. The weighted objective function includes design goals such as to maximize packing density, minimize bracketing costs, and minimize tube routing costs, and penalty terms that evaluate the amount of component overlap, container penetration and spatial constraint violations. Owing to the nature of simulated annealing, large numbers of evaluations must be performed throughout the run (the algorithm can make hundreds of thousands of moves). Thus the evaluation of the objective function terms must be performed quickly. Of the primary terms considered, the single most computationally expensive one to evaluate is interference detection and quantification between components. Thus, to minimize overall run time, we must minimize the complexity associated with this operation.

In conjunction with the simulated annealing algorithm, we utilize a hierarchical decomposition of geometric models for efficient and effective interference analysis; rough analyses are performed at the start of the algorithm (with a low-resolution model) and more refined analyses (with a high-resolution model) are performed towards the end of the algorithm where more accurate evaluations are required. It is the combination of the annealing algorithm and the hierarchy of models for fast evaluation that make this approach unique and effective. Each aspect of the algorithm will now be described in more detail.

Simulated annealing

Simulated annealing is a stochastic optimization technique introduced by Kirkpatrick *et al.*[3]. In summary, 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

$$P_{\text{accept}} = e^{-\frac{\Delta C}{T}} \quad (1)$$

where ΔC is the change in objective function due to the move and T is the current temperature. The temperature starts out high and decreases with time. Initially, 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.

The temperature in a simulated annealing algorithm is controlled by an annealing schedule which consists of an initial temperature, the number of steps taken at each temperature, the amount by which the temperature is decreased for each reduction, and an overall number of temperatures. Various adaptive annealing schedules exist to improve the effectiveness of the simulated annealing algorithm. We take advantage of two of them within our algorithm. The first is from Huang *et al.*[28], called HRSV, which automatically tailors the temperature profile to a given problem based on statistical information about states visited during the optimization. The second is from Swartz and Sechen[29] based on Lam and Delosme[30] which accelerates equilibrium by targeting an acceptance ratio. The choice of the schedule is determined by the user and the implementation readily allows the use of alternative schedules.

We adopt the perturbation-based approach of Szykman and Cagan[9] to define the allowable moves for a component. A 'move set' is defined such that perturbations that can be made to the state of the system are chosen from a predetermined set of discretely sized moves. These moves consist of translation moves, rotation moves and the swapping of positions of two components. For rotation and translation the allowable perturbation move directions for each component consist of a set of orthogonal axes together with plus or minus directions, giving six full degrees-of-freedom (DOF). The minimum number of move directions for a component can be zero, indicating that the component is not subject to any moves. Translation moves are realized through a finite step in one of the orthogonal axes as done by Szykman and Cagan; we extend that work to include rotation moves which are similarly realized through a finite angle about one of the same orthogonal axes. A 4-by-4 transformation matrix is constructed for each component to keep track of its position and orientation in the global coordinate system after the translation and rotation moves. The ability of the algorithm to efficiently handle arbitrary rotations is coupled with our approach to intersection calculations as described below.

In simple simulated annealing algorithms, the probabilities of selecting given moves from the move set are fixed by the user before the algorithm is run. To improve efficiency further, the algorithm uses a move selection strategy introduced by Hustin and Sangiovanni-Vincentelli[31] which dynamically modifies move selection probabilities based on past performance, so that moves which are likely to improve the design have a high probability of being selected and those which are less likely to lead to better designs have a lower probability of being chosen. The selection probabilities are updated periodically as the algorithm runs, since the utility of various moves may change as the optimization proceeds. By decreasing the probability of applying moves that will be rejected, the amount of wasted search is reduced. See Szykman and Cagan[9] for more details on Hustin move sets.

Objectives

Different layout problems have different goals, or objectives. We have formulated several objectives and included them as options within our algorithm. Among them are:

- *Maximization of packing density.* Essentially this means minimizing the total space occupied by a set of components. Mathematically this is represented as the ratio of the sum of individual component volumes to the minimal rectangular volume that contains all components.
- *Minimization of routing costs.*[10] Many problems require routes of either wires or tubes between components, in addition to the placement of the components themselves. The cost of routing can be reduced through the minimal use of tube/wire length, and using as few bends in the routes as possible.
- *Maximization of assemblability.* In this paper we introduce the idea of bracketing to model the costs of manufacturing a product. Although certain layouts may be preferable from a packing viewpoint, they may be inefficient from an assembly perspective. While the algorithm could handle any assemblability analysis that is coded (and preferably efficient), we choose a simple but powerful approach that essentially penalizes a component from moving away from a wall, ceiling or floor based on a structural beam support analysis.
- *Minimization of configuration costs.* Often, layout problems are contingent on selection of the type, size and number of components used. It is desirable to minimize the overall cost of the layout by determining not only optimal positions but also optimal configurations of components. We have implemented an objective function term to select the proper number of components to be used; other types of component selection are certainly possible.
- *Minimization of center of gravity.* In some applications, heavy components need to be placed as low as possible within the container (for purposes of stability, for example). Calculating the center of gravity is straightforward if the density of the components are specified.
- *Performance objectives.* Performance goals can be articulated and included in the optimization. For example Campbell *et al.*[32] describe a model, to be used in conjunction with a stochastic optimization layout algorithm such as ours, which analyzes the layout for thermal properties where the temperature of the layout is to be minimized.

Multiple objectives can be considered concurrently; each goal is added to a weighted multi-objective function. Objectives represent desires in that each goal should be minimized as much as feasible. However, in so doing, there are design constraints that must be satisfied. In our formulation, constraints are added to the objective function as penalty terms. Like objectives, any constraint that can be articulated (again, preferably efficiently) can be included in the problem formulation. In the current implementation, only spatial constraints of component–component overlap, component-outside-of-container penetration, and positional

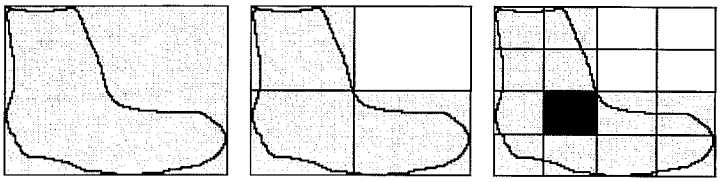

Figure 1 Three levels of decomposition of model in 2D

constraint violation are considered. From an annealing perspective, once included in the weighted objective function, each penalty term is just considered another term to be minimized. Details of the spatial constraint formulations and calculations are described in the next sections.

Multi-resolution modeling

During the annealing process, components are permitted to overlap under the assumption that moving through infeasible designs can lead to superior feasible designs. Recall that evaluating the volume of intersection of components is the most time-consuming part of the core objective function; therefore, full polyhedral interference checking and volume measurement of numerous and complex components is prohibitive owing to the potentially large number of design states visited. To obtain interference calculations in reasonable time between two arbitrarily oriented and shaped objects, we use a multi-resolution modeling scheme, based upon octrees, to approximate the geometry of components. This approach allows us to trade-off accuracy of evaluation with the time taken to calculate the overlap objective. Of course, overall solution quality must not suffer. Fortunately, simulated annealing has a unique characteristic; when combined with multi-resolution models, it forms a powerful and efficient tool.

Early in the annealing process, nearly all design state perturbations are accepted as new design states (the algorithm essentially performs a random walk through the solution space) and thus the algorithm does not require a very accurate estimate of the objective function. Conversely, later in the process, design perturbations are smaller, and more accurate evaluations of the objective function are necessary. By changing the resolution level at which intersections are calculated, a more or less accurate evaluation of the amount of interference is determined; at lower levels, less accurate calculations are obtained with minimal computation, while at higher levels, more accurate evaluations take longer to calculate. We are thus able to specify the quality of intersection calculations needed for each part of the annealing run, significantly saving computational time. Several heuristics can be used to determine which resolution level is to be used at a particular stage of the annealing algorithm. The simplest one is to increase the accuracy of the calculations as a function of decreasing temperature.

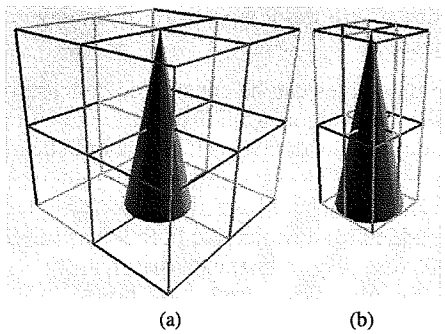

Figure 2 Difference in voxel sizes between a traditional octree (a) and our representation (b)

Modeling components and calculating intersections

Our models are very similar in nature to the octree decomposition model which finds its roots in graphics research (e.g. References [33–35]). An octree is an approximation of an object at various levels of resolution based on successive binary division along each coordinate direction (each division in 3-space divides a cube into eight cubes). We classify each cube at each level of division as white, black and gray—white cubes indicate that no part of the actual object sits inside that cube, black indicates that the entire cube rests inside the original object and gray indicates a partial intersection. We can take advantage of this classification for efficient intersection testing, as discussed below.

Our models differ from octrees in two ways. First, and most important, is the amount of data retained. True octrees do not further decompose non-gray nodes while our representation does include this decomposition. *Figure 1* illustrates three levels of decomposition of a model in 2-D. Second, in the generic notion of an octree, a component is modeled within an 'octree box', which presumably contains the entire component, and is generally a cube with equal side lengths. In other words, the first level of an octree is always a gray cube (unless the component being modeled is a cube, in which case the first and only level would be a black cube), and all voxels are cubes. Our models, however, use the minimal rectangular solid that completely contains the component as its first level, and the voxels are rectangular solids with proportionally consistent dimensions upon successive levels (*Figure 2*). Although not significant differences, they improve the efficiency of intersection calculations for our purposes.

Interference detection is the process by which one can determine if two components are intersecting with each other. Interference evaluation quantifies the 'degree' of intersection, so that appropriate penalty terms can be devised. At each iteration in the optimization process, the annealing algorithm perturbs the positions of components and requires interference detection and evaluation. As noted earlier, the overall speed of the algorithm critically depends on the speed of the interference detection and evaluation code. Lin[36] and Garcia-Alonso *et al.*[37] suggest two such interference detection schemes based on a hierarchy of geometric approximations. In our use of these approaches, as the annealing algorithm progresses, the models move down in the hierarchy for more accurate analysis. At each level, the amount of overlap is calculated as the sum of the volume of cubes that overlap at that level of resolution, giving an accurate evaluation at a specified level of resolution. The final intersection accuracy is dependent only on the

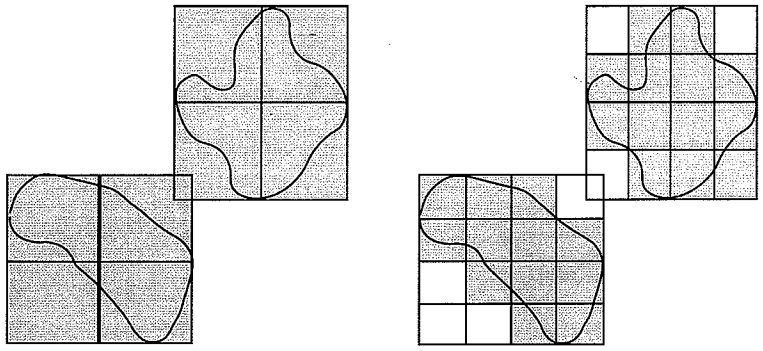

Figure 3 Overlap of 2 objects at level 1 (a); no overlap of same objects at level 2 (b)

user-specified lowest resolution. *Figure 3* illustrates a configuration of two objects, whose bounding box has a 'volume' of unit 1. In *Figure 3a* the objects are represented at level 1 and overlap with a volume of 1/4. *Figure 3b* illustrates the same objects but represented at level 2; here there is no overlap observed and the volume of overlap is 0. In general, as the levels increase the accuracy of volume calculations increase as well. Because of the efficiency and generality of this approach to intersection evaluation, intricately shaped components can be arbitrarily translated and rotated within a simulated annealing algorithm. The approach leads to the first powerful and fast approach to solving the general 3-D layout problem.

Constraint modeling and satisfaction

The approach discussed thus far permits the packing of components into a container. The more general layout, or component placement, problem differs in that components are constrained with respect to the container and each other. As discussed in Szykman and Cagan,[1] this additional aspect of the problem adds a significant level of complexity to the algorithm. We follow the approach of Szykman and Cagan to include both equality and inequality positional constraints in our system.

Hard equality constraints have been implemented by restricting the move set of a particular component in a given direction. If a given component is to have a particular absolute orientation or position with respect to a linear combination of global coordinate axes, it is simply placed in a feasible initial position, and its move set is restricted such that it cannot be moved to an infeasible point or be rotated along an infeasible axis. This helps the efficiency of the annealing.

Inequality constraints, however, are formulated as penalty functions relative to the global coordinate axes. Here the constraint may be violated under the assumption that allowing the annealing to 'pass through' infeasible designs can, in fact, lead to better feasible designs. Constraints are violated towards the beginning of the algorithm and designs are pushed into the feasible range (via penalty functions) as the algorithm progresses. Unlike the Szykman and Cagan work that was restricted to cuboids and cylinders along orthogonal axes, our constraint satisfaction analysis works in conjunction with our multi-resolution modeling framework to permit arbitrary orientations and rotations of arbitrary components. The benefit is a more general description of constraints; the disadvantage is that it is harder to specify specific points on the components within the constraints.

Our current implementation permits two types of spatial inequality constraints: constrained by centers and constrained by extents. In the first, constrained by centers, a constraint may dictate position based upon the origin of component coordinate systems. For example, given two components A and B, one may specify that the x position of A should be greater than the x position of B, where 'x position' refers to the x coordinate of the origin of each component's coordinate system within any specified reference frame (*Figure 4a*). This is the primary constraint type. The second type of constraint, constrained by extents, is similar to the first; however, here all points of the component are included in the analysis rather than its origin alone (*Figure 4b*). With the same components A and B, one could specify that all of A should be at greater x position than all of B, again in any specified reference frame. These constraints are currently implemented by first separating the constraint into x, y, z components for translation, and similar components of x, y, z rotations; the violation of the inequality constraint for each component is then calculated and a penalty function is created that penalizes the objective function for the constraint violation. Note that the more general notion that a specific point on the surface must be aligned relative to a point on a different object is currently not implemented, although the algorithm can readily be extended to include such spatial relations by defining specific geometric relations among classes of constraints.

TEST CASES

To validate our algorithm, we consider problems to which the global optimum is known, as specified in References [9,2]. The first example is the packing of 8 equally sized cubes into a cube of 8 times greater volume. The problem is run in two different cases, changing the maximum model resolution level used from 3 to 4 (level 4 being of higher resolution). Note that by allowing the models to proceed to the fourth level, final configuration quality certainly improves, but at the expense of computation time. The second example also uses the 8-cube problem but adds two positional inequality constraints that impose certain cubes to be in front of other cubes in certain directions. Again, this problem is run up to level 3 and level 4, with the quality improving at the higher resolution but the algorithm taking a longer time to run. In the constrained case the constraints are satisfied in every run of the algorithm. The wide range of solution times for the 8-cube cases comes from the algorithm's ability to halt the annealing process prematurely

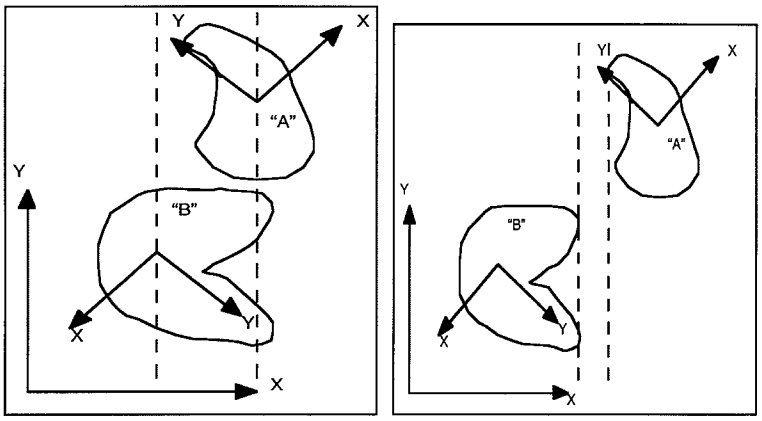

Figure 4 Example of constraint by centers (a), and by extents(b)

if no design improvements are detected over a sequence of iterations.

The unconstrained 8-cube problem is then extended to the packing of 64 cubes into a cube of 64 times greater volume (using models up to level 4 in resolution). Finally, to demonstrate the validity of decomposition modeling, 16 cogwheels are packed into a container (using models up to level 5 in resolution). The cogwheels are sized such that they can only fit into the container if their teeth intermesh. Further, they are constrained to rotations along their length but with arbitrary translations. A summary of the test cases is shown in *Table 1*; for all of these test runs the Lam and Delosme schedule was used.

Note that in all cases, the amount of intersection in the final configuration is quite low (the percentage is a ratio of total intersection volume to the sum of all component volumes). Values for the proportion of components placed outside the container is somewhat higher, particularly in the 8 cube cases. However, the container is exactly the same size as the 8 cubes, with no leeway. Because of the cellular qualities of our models, and the discrete sizing of moves in the move set, finding a placement such that no intersection

Table 1 Summary of test case results

20 Runs each	% Outside container	% Intersection	Constraint violation	Inverse packing density	Time to solve (s)
8 cubes, level 3					
Mean	9.50	1.32		1.14	30.42
SD	1.82	1.83		0.14	9.04
Min	7.94	0.00		1.03	15.48
Max	15.52	6.58		1.63	42.51
Best of 3 mean	8.90	0.43		1.10	30.02
8 cubes level 4					
Mean	5.80	0.06		1.08	68.17
SD	1.90	0.14		0.07	38.43
Min	4.24	0.00		1.03	28.14
Max	9.09	0.39		1.26	180.06
Best of 3 mean	4.63	0.00		1.05	74.72
8 cubes, constraints, level 3					
Mean	9.58	1.20	0	1.14	39.74
SD	2.56	1.74	0	0.09	37.12
Min	4.62	0.00	0	1.03	10.55
Max	14.26	6.78	0	1.35	193.99
Best of 3 mean	8.34	0.00	0	1.09	58.71
8 cubes, constraints, level 4					
Mean	7.68	0.16	0	1.13	80.90
SD	2.63	0.47	0	0.11	50.97
Min	4.24	0.00	0	1.04	35.01
Max	15.97	2.06	0	1.46	209.98
Best of 3 mean	5.35	0.06	0	1.14	100.22
64 cubes					
Mean	3.63	0.66		1.08	3,765.71
SD	0.60	0.49		0.02	118.09
Min	2.39	0.10		1.05	3,584.54
Max	4.83	1.65		1.13	4,044.71
Best of 3 mean	3.01	0.24		1.06	3,755.09
Cog wheels					
Mean	1.31	1.03		1.83	817.16
SD	0.43	0.44		0.07	292.71
Min	0.74	0.35		1.72	448.21
Max	2.57	2.01		2.06	1,328.69
Best of 3 mean	1.05	0.92		1.78	808.67

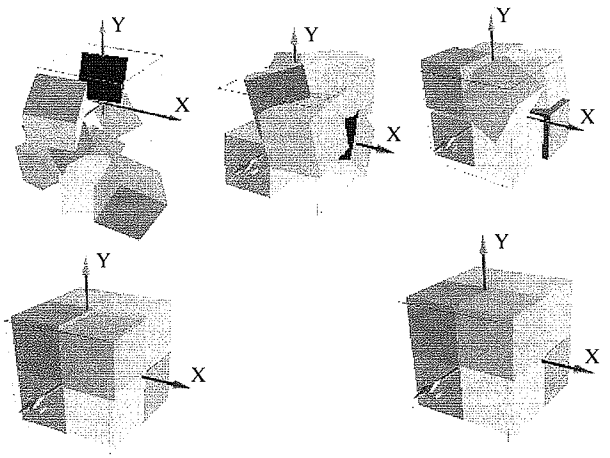

Figure 5 Snapshots of 8-cube packing example at sequences up to the final configuration at 29,000 iterations

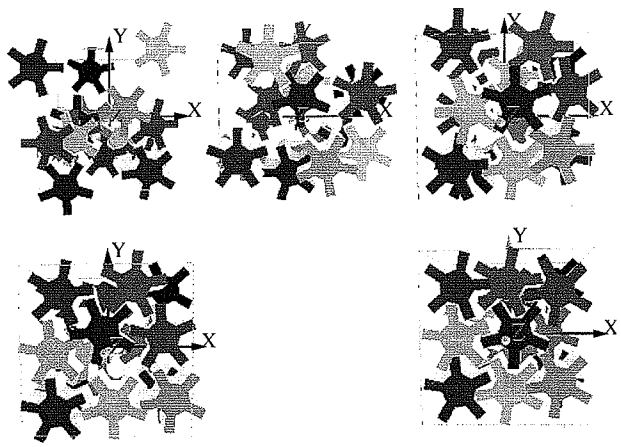

Figure 6 Snapshots of 16-cogwheel-to-cube packing example with sequences up to the final configuration at 75,000 iterations

and no container violation exists is unlikely; rather the overlap is within a user-specified resolution. Data from the cog wheel case show that as the algorithm is given more room within which to place components (in this case, the container has a volume greater than the maximum possible packing density of 1.0), the amount placed outside of the container is significantly reduced.

Results of the inverse packing density (i.e. the ratio of the volume of the bounding box of all components to the sum of the volumes of all components) in all cases is quite good, particularly when the algorithm is given higher resolution levels to explore. The optimal inverse density for the 8 and 64 cube cases is 1; for the cog wheel case it is unknown, though approaches 1.

Owing to the stochastic nature of the algorithm, a heuristic often used with simulated annealing is to run the algorithm three times or more and select the best solution; this approach helps to make sure that if a run settles into a poor local optimum it will not influence the chosen design solution. The last line in each example in *Table 1* lists the results of taking the best of three consecutive runs with the same data. The approach does improve the overall quality of the solutions, although owing to random groupings the mean of the groupings may not be better than the overall mean of all the runs.

The algorithm is coded in C++ and runs on all major UNIX platforms. The code can also provide real-time visualization of the optimization process on Silicon Graphics workstations. Snapshots of the 8-cube and cogwheel problems are shown in *Figures 5 and 6*, illustrating the convergence of the stochastic process. All the timings indicated in *Table 1* are measured on a Silicon Graphics R10000 195 mHz Indigo.

APPLICATIONS

The test cases above confirm the ability of the algorithm to obtain global optima for known solutions. The real power of the approach comes in design layout generation when the design has many DOF yet must satisfy spatial constraints. In this section three applications are described. The first uses geometric models of a Saturn car engine compartment to generate alternative rough layouts. The second is the design of a heat pump based on a problem formulation described in Reference [38], which optimizes several objectives including tube routings and assemblability. The third example is an illustration of an order configuration problem for a truck manufacturer, where the chassis of a truck is configured for various wheelbase lengths. Here the algorithm is modified to allow for the selection of different components. In each of these applications the HRSV schedule was used.

Saturn car engine compartment layout problem

In this example, alternative layouts of an engine compartment are generated using our algorithm. Eleven major parts in a Saturn car engine compartment are modeled to scale using a commercial solid modeling package. These include the engine, engine coolant reservoir, air cleaner, brake booster, master cylinder and brake fluid reservoir assembly, fuse box, battery, engine cooling fan, radiator, windshield washer fluid reservoir and power steering fluid reservoir. The engine compartment is also modeled and represents the container in which the algorithm will lay out these components subject to various constraints. Since we are modeling only a portion of the components and do not know the actual constraints used by the Saturn designers, this example is used only to illustrate the generation of rough alternative solutions to a realistic problem with complicated shapes; it is not meant for comparison with the actual product design.

The layout objective is to pack all the parts into the container such that bracketing costs for the parts are minimized, component overlap is avoided, and positional constraints that describe functionality are satisfied. These constraints specify that all the fluid reservoirs must remain upright, and certain rotational movements of the engine, fan and radiator are prohibited. A set of inequality constraints specify the positional requirements between pairs of components to ensure proper functioning and service. These include easy access of the fuse box and the fluid reservoir openings, and the relative alignment of the fan and the radiator, and of the brake booster and the master cylinder. Each component originally has 3 translational DOF and 3 rotational DOF. After the hard equality constraints are applied, the total DOF of the problem is reduced from 66 to 38.

Owing to the large number of DOF still available after the necessary constraints are applied, different layout solutions are obtained each time the algorithm is run, generating a

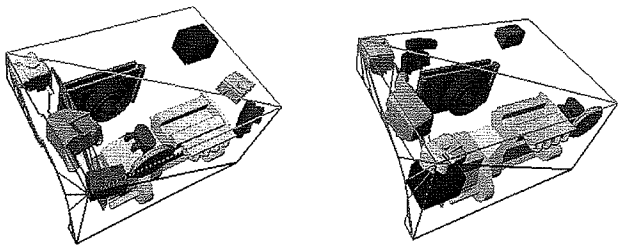

Figure 7 Two alternative layouts for the Saturn engine compartment

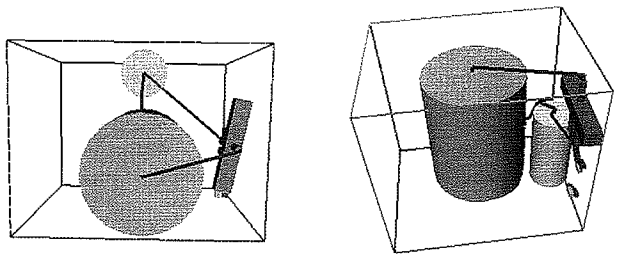

Figure 8 Two alternative layouts of the heat pump problem

series of alternative feasible configurations for designers to choose from; solution snapshots for two runs are shown in *Figure 7*. This clearly illustrates one advantage of this algorithm, namely the generation of comparatively good alternative layouts. Further, by activating, relaxing, or modifying certain constraints, layouts that better reflect the designers' intentions can be generated.

Heat pump layout and routing problem

The integrated layout and routing of a heat pump product using a simulated annealing algorithm was first introduced by Szykman *et al.*[38] where the rotations of the components were limited to multiples of 90°. The problem included packing density and tube routing optimization along with spatial constraint satisfaction. Beyond relaxing the limitation of 90° rotations, this example varies from the previous formulation in several ways: we include a container in which the components should be placed; the minimization of tube length within the container by definition increases packing density and thus we choose not to explicitly maximize packing density; we add the objective function that minimizes bracketing of the components. This realistic example thus illustrates the tradeoff between multiple, conflicting objectives.

Seven components are to be packed into a rectangular container. The main components are the compressor (a large cylinder), accumulator (a medium cylinder), and reversing valve (a small cylinder on top of three other small cylinders). Input and output heat exchanger valves and service valves are specified by small blocks. Dimensions can be found Szykman *et al.*[38].

After the hard equality constraints are applied, the following DOF for individual components are allowed:

- the compressor and the accumulator are free to translate along the horizontal plane, and to rotate around the axis perpendicular to the plane;
- the reversing valve is free to translate along all the three principle axes, and to rotate around the perpendicular axis;
- the coil ports and the service valves are free to move within a vertical plane which is located on the front side of the container.

The inequality constraint requirements specify that the input and output heat exchanger valves should be located with one above the other, and be 100 mm apart.

The tube routing includes six tube routes to be connected, with endpoints of each route as:

- the top of the compressor to the top cylinder of the reversing valve;
- the side of the compressor to the top of the accumulator;
- the top of the accumulator to the bottom middle cylinder of the reversing valve;
- the bottom left cylinder of the reversing valve to the side of the input heat exchanger valve;
- the bottom right cylinder of the reversing valve to the side of the input service valve;
- the side of the output heat exchanger valve to the side of the output service valve.

Component placement, route minimization and bracketing are considered concurrently to minimize the overall cost of the design. Route minimization includes four terms, namely route length, number of bends, component–route intersection, and container–route protrusion.

Figure 8 shows two snapshots of solutions to the problem as described; *Figure 9* shows two snapshots of solutions if the rotations are constrained to 90° as with the original problem. It is interesting to note the difference in the solutions without the 90° restriction. Notice how, without the restriction, the compressor (the largest cylinder) rotates to accommodate a shorter tube route between itself and the accumulator (the medium cylinder), and the reversing valve is more flexibly positioned and oriented to shorten the route length; all the other objective function terms are kept the same. Not only is the route length shortened on average by 11%, but the solutions become more consistent with an improvement in standard deviation by 61%. By relaxing the 90° restriction, then, the quality of the solutions, in terms of the objective function values, is improved.

Truck chassis order configuration problem

A final example of the application of the algorithm is the automated layout of commercial truck chassis. Trucks are created to meet consumer requirements, giving various layouts based upon wheelbase length, required towing capacities, optional equipment, etc. Each truck chassis must be sufficiently rigid (through the use of cross members), must translate power from the transmission to the rear axle with minimum vibration (through the use of drive lines), and must support all required equipment (via fixturing).

The objectives in truck chassis layout are minimizing cost by using as few components as possible, minimizing vibration by properly sizing drive line segments, maximizing serviceability by making components easily accessible, and maximizing aesthetic appeal by evenly spacing cross members. Constraints are placed on the minimum and maximum spacing of cross members, the support of universal joints (i.e., CV joints) between drive line segments, and the non-overlap of components.

This example further expands the power of the algorithm by allowing it to configure as well as layout. That is, the

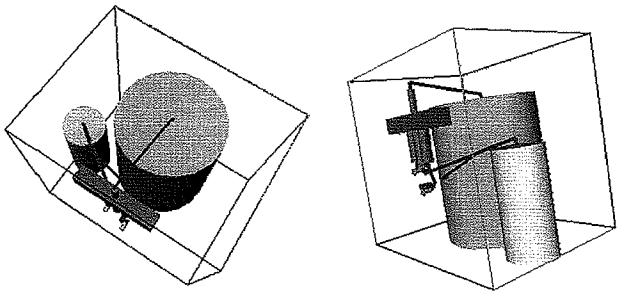

Figure 9 Two alternative layouts for the same problem in *Figure 8* but with the added restriction of only 90° rotations

algorithm chooses how many fixtures, drive line segments, and cross members to use, as well as where they should be placed. Note, for example, the use of two CV joints and five cross members in *Figure 10a*, a chassis with a wheelbase of 7200 mm, while the chassis in *Figure 10b*, with a wheelbase of just 5500 mm, uses only one CV joint and four cross members. This is achieved by determining the maximum number of each component that could be used through given hard constraints, instantiating that many of each component, and allowing each component to be activated or deactivated (added or removed from the design) through integer variables. This action simply becomes an additional move in the move set, and is easily incorporated by modifying the Hustin move set to include the integer variables.

CONCLUDING REMARKS

An approach to 3-D component placement is presented and illustrated through various test cases and applications. The approach uses simulated annealing to search for optimal layouts and hierarchical models of components to efficiently approximate intersections of complex geometric shapes. The simulated annealing and hierarchical models are integrated seamlessly in that the levels of hierarchy correspond to the probabilistic strategy of the annealing algorithm. The result is a very general algorithm able to generate alternative, equivalent-quality layout solutions to design problems in reasonable time. Multiple design objectives, such as packing density, center of gravity, tube routing and bracketing, can be optimized simultaneously, while satisfying both equality and inequality constraints. The algorithm has been applied to a variety of problems in industrial settings with success. The implications of this technique are that general layout problems can be solved faster with greater consideration of alternatives and better up front prediction of performance, layout costs and production feasibility.

ACKNOWLEDGEMENTS

The authors would like to thank Ashish Kolli and Rob Rutenbar for their discussions on this work and Munish Suri for making the geometric models of the Saturn car engine components. They would also like to thank the National Science Foundation under grants MIP-9420372, DDM-9258090, and EID-9256665 for supporting this work.

REFERENCES

1. Szykman, S. and Cagan, J., Constrained three dimensional component layout using simulated annealing. *ASME Journal of Mechanical Design*, 1997, **119**(1), 28–35.
2. Kolli, A., Cagan, J. and Rutenbar, R. A., Packing of generic, three dimensional components based on multi-resolution modeling. In *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference: Design Automation Conference, 96-DETC/DAC-1479*, Irvine, CA, 18–22 August 1996.
3. Kirkpatrick, S., Gelatt, C.D. Jr. and Vecchi, M.P., Optimization by simulated annealing. *Science*, 1983, **220**(4598), 671–679.
4. Cohn, J.M., Garrod, D.J., Rutenbar, R.A. and Carley, L.R., KOAN/ANGRAM II: new tools for device-level analog placement and routing. *IEEE Journal of Solid-State Circuits*, 1991, **23**, 330–342.
5. Sechen, C., *VLSI Placement and Global Routing Using Simulated Annealing*. Kluwer, Boston, 1988.
6. Sechen, C. and Sangiovanni-Vincentelli, A., The TimberWolf placement and routing package. *IEEE Journal of Solid-State Circuits*, 1985, **20**(2), 510–522.
7. Sherwani, N., *Algorithms for VLSI Physical Design Automation*. Kluwer, Boston, 1992.
8. Wong, D. F., Leong, H. W. and Liu, C. L., *Simulated Annealing for VLSI Design*. Kluwer, Boston, 1988.
9. Szykman, S. and Cagan, J., A simulated annealing approach to three-dimensional component packing. *ASME Journal of Mechanical Design*, 1995, **117**(2(A)), 308–314.
10. Szykman, S. and Cagan, J., Synthesis of optimal non-orthogonal routes. *ASME Journal of Mechanical Design*, 1996, **118**(3), 419–424.
11. Fujita, K., Akagi, S. and Hase, H., Hybrid approach to plant layout design using constraint directed search and an optimization technique. In *Advances in Design Automation 1991: Proceedings of the 17th ASME Design Automation Conference*, 1991, Vol. 1, pp. 131–138.
12. Kim, J.J. and Gossard, D.C., Reasoning on the location of components for assembly packaging. *Journal of Mechanical Design*, 1991, **116**, 375–381.
13. Landon, M.D. and Balling, R.J., Optimal packaging of complex parametric solids according to mass property criteria. *Journal of Mechanical Design*, 1994, **116**, 375–381.
14. Sandgren, E. and Dworak, T., Part layout optimization using a quadtree representation. In *Advances in Design Automation 1988: Proceedings of the 14th ASME Design Automation Conference*, Kissimmee, FL, 25–28 September 1988, pp. 211–219.
15. Udy, J.L., Balling, R.J., Benzley, S.E. and Landon, M.D., Computation of interferences between three dimensional objects and the optimal packing problem. *Advances in Engineering Software*, 1988, **10**(1), 8–14.

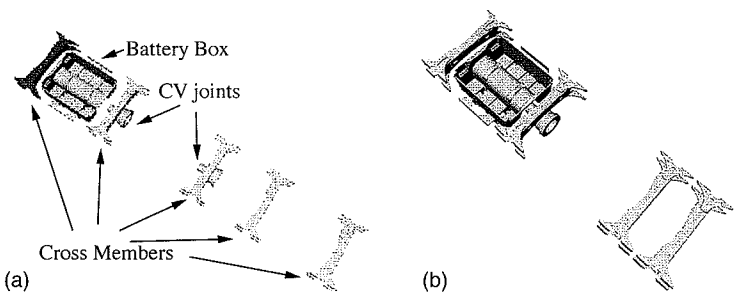

Figure 10 Truck chassis configurations of (a) 7200 mm wheelbase and (b) 5500 mm wheelbase

16. Corcoran, A. L. III and Wainwright, R. L., A genetic algorithm for packing in three dimensions. In *Applied Computing: Technological Challenges of the 1990's—Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, Kansas City, KS, 1992, pp. 1021–1030.
17. Kawakami, T., Minagawa, M. and Kakazu, Y., Auto tuning of 3-D packing rules using genetic algorithms. In *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS '91*, 1991, Vol. 3, pp. 1319–1324.
18. Coffman, E. G., Jr, Garey, M. R. and Johnson, D. S., Approximation algorithms for bin-packing-an updated survey. In *Algorithm Design for Computer System Design*, ed. G. Ausiello, M. Lucertini and P. Serafini. Springer, New York, 1984, pp. 49–106.
19. Dowsland, K.A. and Dowsland, W.B., Packing problems. *European Journal of Operational Research*, 1992, **56**, 2–14.
20. Dyckhoff, H., A typology of cutting and packing problems. *European Journal of Operational Research*, 1990, **44**, 145–159.
21. Cagan, J., A shape annealing solution to the constrained geometric knapsack problem. *Computer-Aided Design*, 1994, **28**(10), 763–769.
22. Kämpke, T., Simulated annealing: use of a new tool in bin packing. *Annals of Operations Research*, 1988, **16**, 327–332.
23. Jajodia, S., Minis, I., Harhalakis, G. and Proth, J.M., CLASS: Computerized LAyout Solutions using Simulated Annealing. *International Journal of Production Research*, 1992, **30**(1), 95–108.
24. Smith, N., Hills, W. and Cleland, G., A layout design system for complex made-to-order products. *Journal of Engineering Design*, 1996, **7**(4), 363–375.
25. Hills, W. and Smith, N., A new approach to spatial layout design in complex engineered products. In *Proceedings of the International Conference on Engineering Design (ICED 97)*, Tampere, Finland, 19–21 August 1997.
26. Dai, Z. and Cha, J., An octree method for interference detection in computer aided 3-D packing. In *Advances in Design Automation 1994: Proceedings of the 20th ASME Design Automation Conference*, 1994, Vol. 1, pp. 29–33.
27. Dai, Z. and Cha, J., An Octree based heuristic algorithm for 3-D packing. In *Advances in Design Automation 1994: Proceedings of the 20th ASME Design Automation Conference*, 1994, Vol. 2, pp. 125–133.
28. Huang, M. D., Romeo, F. and Sangiovanni-Vincentelli, A., An efficient general cooling schedule for simulated annealing. In *ICCAD-86: IEEE International Conference on Computer Aided Design—Digest of Technical Papers*, Santa Clara, CA, 11–13 November 1986, pp. 381–384.
29. Swartz W., and Sechen, C., New algorithms for the placement and routing of macro cells. In *IEEE Proceedings: Cat No. 90CH2924-9, IEEE Conference on Computer-Aided Design*, Santa Clara, CA, 11–15 November 1990, pp. 336–339.
30. Lam, J. and Delosme, J., Performance of a new annealing schedule. In *Proceedings of the 25th ACM/IEEE Design Automation Conference*, 1988, pp. 306–311.
31. Hustin, S. and Sangiovanni-Vincentelli, A., TIM, a new standard cell placement program based on the simulated annealing algorithm. Presented at IEEE Physical Design Workshop on Placement and Floorplanning, Hilton Head, SC, April 1987.
32. Campbell, M.I., Amon, C.H. and Cagan, J., Optimal three-dimensional placement of heat generating electronic components. *ASME Journal of Electronic Packaging*, 1997, **119**(2), 106–113.
33. Aref, W.G. and Samet, H., An algorithm for perspective viewing of objects represented by octrees. *Computer Graphics Forum*, 1985, **14**(1), 59–66.
34. Bloomenthal, J., Polygonization of implicit surfaces. *Computer Aided Generation*, 1988, **5**, 341–355.
35. Meagher, D., Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 1982, **19**(2), 129–147.
36. Lin, M.C., Efficient collision detection for animation and robotics. PhD Thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1991.
37. Garcia-Alonso, A., Serrano, N. and Flaquer, J., Solving the collision detection problem. *IEEE Computer Graphics and Applications*, 1994, **14**(3), 36–43.
38. Szykman, S., Cagan, J. and Weisser, P., An integrated approach to optimal three dimensional layout and routing. *ASME Journal of Mechanical Design*, 1997, **120**(3), 510–512.

Jonathan Cagan, the George Tallman and Florence Barrett Ladd Development Professor in Engineering, is an Associate Professor of Mechanical Engineering at Carnegie Mellon University, Director of the Mechanical Engineering Computational Design Laboratory, and holds appointments in Computer Science and Biomedical Engineering. He received his BS and MS degrees from the University of Rochester in 1983 and 1985, and his PhD from the University of California at Berkeley in 1990, all in Mechanical Engineering. Professor Cagan's research is in the area of computational design and integrated product development. He is a registered Professional Engineer in the Commonwealth of Pennsylvania and the State of California.

Drew Degentesh is a research assistant and PhD candidate at Carnegie Mellon University, having completed his undergraduate work at Purdue in 1994. While at CMU he has researched stress analysis of space frames, the optimization of component packing, and product design for the elderly. His current work focuses on the use of the Internet by the elderly. He resides in the Shadyside section of Pittsburgh, not far from his Southwestern Pennsylvanian childhood home.

Su Yin is a research assistant and PhD candidate in the Mechanical Engineering Department at Carnegie Mellon University. She received her BS degree in Mechanical Engineering from the University of Science and Technology of China in 1992, her MS degree in Optical Instrumentation from Shanghai Institute of Optics and Fine Mechanics in 1995 and her MS degree in Mechanical Engineering from California Institute of Technology in 1996. Her research focuses on computational algorithms for product layout optimization.