An Extended Pattern Search Algorithm for Three-Dimensional Component Layout

Su Yin
e-mail: yin@andrew.cmu.edu

Jonathan Cagan
e-mail: cagan@cmu.edu

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

An extended pattern search algorithm is introduced for efficient component layout optimization. The algorithm is applicable to general layout problems, where component geometry can be arbitrary, design goals can be multiple and spatial constraint satisfactions can be of different types. Extensions to pattern search are introduced to help the algorithm to converge to optimal solutions by escaping inferior local minima. The performance on all of the test problems shows that the algorithm runs one-to-two orders of magnitude faster than a robust simulated annealing-based algorithm for results with the same quality. The algorithm is further extended to solve a concurrent layout and routing problem, which demonstrates the ability of the algorithm to apply new pattern strategies in search and to include different objective functions in optimization.
[S1050-0472(00)01901-2]

1 Introduction

Component layout plays an important role in electrical chip and board design [1] and facility layout design [2,3] as well as in mechanical system and product design [4,5]. The effect of a beneficial layout of a product or system can be seen in reduction of manufacturing costs, improvement in performance, and reductions in product size and transportation costs. Recent advances in layout automation algorithms have shown promise in addressing but fallen short in delivering effective and efficient tools for layout optimization. This paper introduces a new approach to layout optimization that demonstrates one-to-two orders of magnitude improvement in speed over a robust simulated annealing algorithm.

Although there can be different formulations of a layout problem, it usually can be abstracted to a constrained optimization problem. An assignment of geometric coordinates and the orientations of components that satisfies certain requirements of a layout and that minimizes certain cost criteria is sought. A number of cost functions may be of interest in layout, such as minimizing required space occupation, lowering center of gravity, and minimizing assembly and routing costs. These cost measures are minimized with different priorities in different applications. The minimization of the cost functions has to be done under certain constraints imposed by the design, fabrication, and operation requirements to ensure that the results are practically correct. They include restrictions on degrees of freedom for individual components, and the proximity or separation requirements between pairs of components.

The layout algorithm has many variants. Most of them can deal with only a restricted class of layout problems. One of the major research goals in layout is to improve the robustness of layout algorithms. By robustness, we mean that the algorithm should be adaptable to components of arbitrary geometry. It should also allow the incorporation of different cost functions of varying degrees of importance. These can be achieved only at the cost of increased run time or, even worse, of reduced quality of the constructed layout. Despite the rapid evolution of design automation tools, there still lacks an efficient layout method that consistently produces high quality output.

In this paper, we introduce an extended pattern search algorithm to solve the general layout problem efficiently and with consistent quality. Pattern search algorithms [6] are a subset of direct search algorithms. They use exploratory moves along a set of pattern directions and need only direct comparisons of objective function values to determine new state acceptances. The highly structured search methods make pattern search algorithms more efficient than the stochastic algorithms, such as simulated annealing. But the possibility of converging to inferior local minima presents a problem for pattern search approaches when good quality solutions are required. By introducing a set of extensions to pattern search methods, we choose pattern search directions and exploratory moves that can reflect the characteristics of the layout problem, randomize search orders in terms of components and directions to better explore the design space, and combine the localized and global search strategies to help converge to optimally directed designs. The algorithm is able to generate good quality layouts for arbitrary geometry by optimizing multiple design goals while satisfying spatial constraints. The run time of the algorithm on test problems is one-to-two orders of magnitude faster than that of a simulated annealing-based algorithm which has been shown quite effective in solving realistic mechanical and electro-mechanical layout problems.

The rest of the paper is organized as follows: We provide a survey of current layout algorithms in Section 2. This is followed by an overview of pattern direct search algorithms in Section 3. Section 4 presents the extended pattern search algorithm for the layout problem and describes the extensions introduced to the original algorithm. We compare performance of the algorithm with that of a simulated annealing-based algorithm in Section 5. In Section 6, we further extend the algorithm to solve a concurrent layout and routing problem which demonstrates the versatility of the algorithm. Finally, we draw conclusions in Section 7.

2 Related Work

A variety of numerical optimization techniques have been approached to tackle component layout and packing problems. They can be classified into four major categories: heuristic methods, gradient descent algorithms, genetic algorithms, and simulated annealing-based algorithms.

Practically all versions of the layout problem as a whole are intractable due to its combinatorial nature. This makes heuristic methods a reasonable resort. Heuristic layout algorithms can be most often found in the operations research field. They are capable of solving specific classes of problems efficiently, however they have difficulty when applied to general layout problem. Typical examples of this kind of algorithms include cutting and bin-packing algorithms [7,8]. Dai and Cha [9] presented an octree-

Contributed by the Design Automation Committee for publication in the ASME JOURNAL OF MECHANICAL DESIGN. Manuscript received Feb. 1998; revised Jan. 2000. Associate Technical Editor: M. A. Gante.

based heuristic algorithm for three-dimensional component packing. But their approach is limited in terms of achievable packing density and representative objective function.

Landon and Balling [10] provided a three-dimensional packing solution by using explicit gradients in the optimization search routines. Kim and Gossard [11] utilized a combination of gradient-based optimization and solid modeling method techniques to address the packing problem. The gradient-based methods are first-order methods. They impose restrictions on the form of the objective function and thus are not suitable for the general layout problem since some of the objective function terms for the general layout problem require simulations that do not have analytical forms.

Genetic algorithms for layout [12–14] are general in the sense of problem representation and generic operators. However, there are often restrictions in the bit string or binary tree encoding in order to discretize the design variables and to make the search space manageable.

Simulated annealing algorithms [15] have been used to solve VLSI layout and routing problems [16] successfully. The algorithms have been recently used to tackle mechanical layout problems as well. Hills and Smith [5] and Smith et al. [17] combined simulated annealing and knowledge-based systems to produce layouts for made-to-order products. Szykman and Cagan [18–20] used simulated annealing to solve 3-dimensional constrained layout and routing problems of blocks and cylinders. Kolli et al. [21] and Cagan et al. [4] made extensions to that work by introducing hierarchical representations of arbitrary geometry, thus making the technology capable of solving the general three-dimensional layout problem.

A thorough discussion of the layout algorithm literature can be found in Cagan et al. [4]. Only simulated annealing-based algorithms have addressed the general three-dimensional layout problem. The stochastic nature of simulated annealing avoids the problem of solutions being trapped in local optima that faces many other algorithms. Although simulated annealing-based algorithms have yielded impressive results when applied to layout problems, the efficiency and run-time behavior still present a problem for their application. Simulated annealing works well only if a huge amount of computing time is invested because it requires both small temperature decrements and long runs at each temperature. If only a small proportion of the required computing time is invested, the partial solution leaves no indication of the feasibility and quality of the final design state. This means that partial optimizations are not good predictors of feasibility of final designs, and no apparent correlation exists between the quality of partially optimized designs and that of final designs. This is undesirable for design prototyping.

The success of simulated annealing-based algorithms also depends on a suitable choice of the cooling schedule [22,23]. However, making choices of the control parameters in a cooling schedule turns out to be a computationally expensive process and quite an art. Furthermore, it may not be problem-independent, which makes the experiment and implementation process challenging.

What is desired for layout algorithms is the ability to conduct search efficiently using information about objective function values only. This is a requirement for general, non-smooth and nonlinear optimization problems. Also desired is a good global behavior of the algorithms, since we need to locate good quality solutions from arbitrary starting points with no prior knowledge of promising areas. Pattern search algorithms have these properties; however the basic algorithms search for any local stationary point and thus require extensions to avoid inferior local optima, thereby regaining the benefit of simulated annealing-based algorithms.

3 Traditional Pattern Direct Search Algorithms

Pattern direct search algorithms are a subset of direct search algorithms introduced by Hooke and Jeeves [24]. The algorithms were used to solve curve fitting problems in their work and showed promise since they either provided solutions to some problems which had been unsuccessfully attacked by classical methods, or provided faster solutions for problems solvable by classical methods. Torczon and Trosset [6] surveyed the history of pattern search methods and provided some practical suggestions for using them. The search algorithms follow a series of exploratory moves defined by pattern matrices to walk through a design space to search for a stationary point. They rely exclusively on the direct comparisons of function values during search, and thus provide a tool suitable for the exploration of a design space that is nonlinear and combinatorial, as is the case of the layout problem. Lewis and Torczon [25,26] have done extensive research on pattern search algorithms. The convergence analysis [27] and the parallel implementation [28] of pattern search algorithms are also introduced.

A basic pattern search proceeds as follows: An initial state is chosen, and the objective function for that state is evaluated. A step is taken to a new state by applying a move along a pattern direction by a specified step length. The objective function is evaluated again at the new state. The two evaluations are compared, and the better one is chosen; the corresponding state is accepted as the current state. The search continues from the current state along the next pattern direction, following the same criteria of the new state acceptance. After an iteration of explorations along all the pattern directions, the step length is either carried into the next iteration if at least one previous move has led to a better new state, or scaled by a factor that is less than 1. The search stops when the step length is smaller than a pre-specified tolerance. An example of how the algorithm works on a simple 2-dimensional problem can be found in Torczon and Trosset [6].

Each pattern direction can be denoted as a vector of n terms, where n is the number of design variables. The set of pattern directions forms a pattern matrix. The construct of pattern matrices as well as the number of pattern directions can be varied.

New classes of pattern search algorithms [25,26] exist in addition to the basic ones. They are characterized by the nature of choosing and updating pattern matrices and the exploratory moves. These algorithms can reduce the worst case cost of an iteration by almost half.

Pattern search methods exhibit several attractive features that suggest they can be the methods of choice for nonlinear and nonsmooth 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.

4 Extended Pattern Search Layout Algorithm

Pattern search methods provide a useful exploratory tool for solving nonlinear and nonsmooth optimization problems. They are capable of locating the general region of a stationary point from any initial design state. However, extensions are necessary to address the requirements of the layout problem.

There are two major issues to be addressed. First, to allow the designer alternative designs to select from, we wish to generate a variety of equally good layouts from multiple runs instead of a single layout solution to a given problem, which makes a certain degree of randomization desirable. Secondly, although no globally optimal solutions are required or enforced, we certainly want to produce good quality solutions only, which means that stationary-point solutions may not be good enough (i.e., in this multimodal space we want the better of the local optima).

In this section, we present an extended pattern search layout algorithm that generates good quality layouts in a fraction of the time needed by a simulated annealing-based algorithm. Although some of these extensions are specific to the layout problem, others are more general extensions to give pattern search a stochastic characteristic.

4.1 Layout Algorithm Using Extended Pattern Search Methods. The algorithm begins by taking as input a number of components and a container as well as constraints describing spatial relations between components and components and the container. Two sets of pattern directions are used, namely the translational and rotational pattern matrices. The simplest pattern is associated with the coordinate search which alternates the search directions along the *X*-axis, *Y*-axis, and *Z*-axis in three-dimensional space. Other orthogonal or nonorthogonal pattern directions can be chosen as well. Usually, not all the components have the same set of moving directions because there can be constraints applied to individual components that dictate or prohibit certain movements. The pattern matrices are thus chosen to reflect the permitted move directions for each component as well as the generally preferred search strategy.

The initial step size for translations is specified with respect to the size of the container. A common choice is to make the initial step size several times smaller than that of the largest dimension of the container, while maintaining a large enough move to enable any component to span the container space in a reasonable number of moves. The initial step size for the rotations is chosen and updated independent of that for the translations.

A generic extended pattern search layout algorithm is outlined in Fig. 1. From an arbitrary initial state of the components, translations are taken as the first set of exploratory moves. Components are randomly sorted, and the first one in the list is selected. A series of translations take the component to new states according to the predefined pattern directions and step size, accepting new states as the current states whenever an improvement in the objective function occurs, or keeping original states if no such improvement is found. This process is repeated for all the components. If there is no improvement for any of the translations attempted, the step size for translations is scaled down by a factor that is less than but close to 1.

The rotations are then taken as the next set of exploratory moves. Again, the components are randomly sorted, and the process repeats for all the components along all the rotational pattern directions. The same rules apply for the new state acceptance and the step size updating.

After a cycle of translations and rotations, one or more swaps may take place between pairs of randomly chosen components if none of the previous translations and rotations in the cycle has led to a better state. By swapping, we mean that the locations of the two components are switched, but the orientations remain unchanged. Again, the new state is accepted only if it is better than the original one.

The translational, rotational and swapping moves repeat until the stopping criteria are met. Here the criteria are that the step sizes of both translations and rotations are smaller than prespecified tolerances. Note that the choice of translational moves prior to rotational moves is arbitrary, as well as when swapping moves are used. Future research may indicate a preferred order of moves, however the current implementation performs quite well and initial investigation has indicated the order to be irrelevant.

The objective function is formulated as the weighted sum of multiple objectives representing multiple goals, as done by Szykman and Cagan [20]. The constraints in general are added to the

```
Generate an initial layout $L_0$
Evaluate $f(L_0)$
while stopping criteria are not met
    for translatable components along their pattern directions
        Explore pattern translations to generate layout $L_1$
        if $f(L_1) < f(L_0)$
                Accept $L_1$ as the current layout ($L_0 = L_1$)
        else
                Revert to the previous layout
        end if
    end for
    if none of the moves is accepted in the above for loop
        Reduce translation step size
        if step jumping allowed and the criteria met
                adjust translation step size accordingly
        end if
    end if

    for rotatable components along their pattern directions
        Explore pattern rotations to generate layout $L_1$
        if $f(L_1) < f(L_0)$
                Accept $L_1$ as the current layout ($L_0 = L_1$)
        else
                Revert to the previous layout
        end if
    end for
    if none of the moves is accepted in the above for loop
        Reduce rotation step size
        if step jumping is allowed and the criteria are met
                adjust rotation step size accordingly
        end if
    end if

    if no acceptable moves in the above two loops and
        swapping is allowed
        Explore swaps to generate layout $L_1$
        if $f(L_1) < f(L_0)$
                Accept $L_1$ as the current layout ($L_0 = L_1$)
        else
                Revert to the previous layout
        end if
    end if
end while
```

Fig. 1 A generic extended pattern search layout algorithm

objective function as penalty terms. The exception is those constraints that restrict the feasible move directions of individual components, which are reflected and enforced within selected patterns.

4.2 Extensions and Their Justifications. In the extended pattern search layout algorithm described above, five extensions are introduced to address the characteristics of layout problems and to help converge to good quality solutions. They are discussed in more detail in this section, and justifications are provided. No global optimality is guaranteed, however, consistently good local optimal solutions and at times the exact global optimal solutions are found after the introduction of the extensions.

- **Randomized search orders**

By randomly picking the order of components to perform the exploratory moves, we have a better chance to get alternative solutions with varied appearances. The order of selecting search directions for individual components is also randomized, making the move sequence of each component less deterministic to avoid getting trapped in certain local minima. Even with the randomization, the extended pattern search layout algorithm proceeds in a much more disciplined manner than the simulated annealing-based algorithms which make random moves prior to each acceptance choice.

• **Constraint related search directions**

The construct of the search directions not only reflects designers' intentions to apply preferred search strategies, but it also takes into account the constraints applied to individual components. This makes the spatial constraint satisfactions easier and at the same time restricts the feasible search space to a smaller one, thus making the search more efficient.

• **Occasionally allowed step-jumps**

While the general trend of the down scaling of the step sizes is preserved, occasionally the step sizes may take a jump which means that abrupt increases in the step sizes are permitted. Step-jumping delays the satisfaction of the stopping criteria, thus allowing the algorithm to run longer and to take more exploratory moves. It favors a more thorough exploration of the search space and reflects a trade-off between run time and layout quality. The step jumps are used only in complicated problems where the number of movable components is large.

• **Strategically used swapping moves**

In the extended pattern search layout algorithm, translations and rotations are given priority to explore in the neighborhood of the current design and to push the design towards a better state as judged by the objective function. They are allowed to go as far as they can at each step size level. Once they get stuck (possibly in some local minimum), the swaps will take over to allow some dramatic state changes before another cycle of translations and rotations proceeds at smaller step sizes. Swapping provides a chance of doing some global search, enables the algorithm to escape inferior local optima and helps the algorithm to converge to layouts of better quality.

Swapping is used as a perturbation method in the simulated annealing-based algorithm as well. There the move is a choice of equal importance with dozens of other translations and rotations in a move set, while in our extended pattern search algorithm the move is used more strategically.

• **Judiciously chosen hierarchical models**

In our implementation, we take advantage of multiresolution models based on a hierarchical octree decomposition to represent components as described in Cagan et al. [4]. The resolution level directly influences the accuracy of intersection evaluation, and thus affects the quality of the layout. It is desirable to use higher resolution level models in order to generate good quality layouts, but substantially longer time is needed for the evaluation.

To speed up the run time without compromising the quality of the layout, a multiresolution modeling scheme is used in the extended pattern search algorithm. Lower resolution levels are used early in the search and higher resolution levels are used later in the search. The resolution level is chosen such that it is directly related to the translational step size. This makes sense because lower resolution level models are good enough for picking a better state from two consecutive ones when the step size is large. It takes higher resolution models to tell the difference between the amount of intersection in an original state and that in a new state when the step size is small. By moving to a higher resolution level whenever it is needed and only when it is needed, we can best achieve efficiency and solution quality.

In contrast, the simulated annealing-based layout algorithm uses hierarchical models with less efficiency. Although some heuristics can be used to choose resolution levels at different stages of the annealing process, such as to represent it as a function of the temperature, there is no justification for the choices made. Also, due to the different resolution levels used, substantially different objective function evaluations for the same state may result, which can influence the probability of accepting a new state after the resolution level changes and thus disturb the overall accuracy and consistency of the statistical information obtained.

5 Performance

The extended pattern search layout algorithm is tested on a set of benchmark problems originally solved by a simulated annealing-based algorithm. The layout problems are listed below, and the detailed problem descriptions can be found in Cagan et al. [4] and Kolli et al. [21]:

• 8-cubes packing, with the unconstrained version of packing 8 equally sized cubes into a cube of eight times volume, and with the constrained version satisfying two spatial constraints in addition to packing;

• 16-cogwheels packing, where 16 equally sized cogwheels are packed into a box, with the cogwheels sized such that they can fit into the box only if their teeth intermesh. This example is used to test how the algorithm performs when components are of concave shape and in close proximity;

• 64-cubes packing, similar to the 8-cubes packing problem, but with 64 cubes packed into a cube of 64 times volume;

• Saturn car engine compartment layout, a more realistic problem with models of selected major parts in the engine compartment as the layout components, and the compartment itself as the container. The problem has 11 components and 26 constraints relating to the components. The constraints include the alignment of pairs of components and the accessibility of the fluid container openings.

The timing comparisons for each layout problem solved by the two algorithms are illustrated in Tables 1–4, with the number of evaluations and the run time averaging over 10 runs each. Both algorithms are coded in C++ and tested on a Silicon Graphics Indigo (195 MHz, MIPS R10000 CPU, 128 MB RAM). The simulated annealing-based algorithm uses an adaptive annealing schedule and a dynamically modified perturbation-based move strategy as described in Cagan et al. [4].

The octree decomposition model is an approximation method used for efficient intersection evaluation between components of

Table 1 Performance comparison for 8-cube packing problem

	Extended Pattern Search Layout			Simulated Annealing-Based Layout		
Resolution & Constraints	Evals (x1000)	Time (sec)	obj	Evals (x1000)	Time (sec)	obj
Level 3, Uncon	4.1	1.7	1790.2	167.3	57.4	1799.1
Level 3, Con	4.2	1.8	1777.5	162.6	56.9	1779.3
Level 4, Uncon	4.4	9.0	1789.3	165.9	304.8	1781.9
Level 4, Con	4.6	9.3	1780.4	169.6	305.3	1786.0

Table 2 Performance comparison for 16-cogwheel packing problem

	Extended Pattern Search Layout			Simulated Annealing-Based Layout		
Resolution	Evals (x1000)	Time (sec)	obj	Evals (x1000)	Time (sec)	obj
Level 3	36.7	18.9	1909.9	1283.6	558.1	2093.1
Level 4	39.5	67.7	1771.3	705.3	1201.6	1948.5

Table 3 Performance comparison for 64-cube packing problem

	Extended Pattern Search Layout			Simulated Annealing-Based Layout		
Resolution	Evals (x1000)	Time (sec)	obj	Evals (x1000)	Time (sec)	obj
Level 3	382.9	169.2	1996.3	8267.3	2752.4	1999.5
Level 4	396.7	696.0	1977.9	8148.7	9972.3	2008.4

Table 4 Performance comparison for Saturn car engine compartment layout problem

	Extended Pattern Search Layout			Simulated Annealing-Based Layout		
Resolution	Evals (x1000)	Time (sec)	obj	Evals (x1000)	Time (sec)	obj
Level 3	28.8	5.1	3329.4	470.6	76.5	3367.4
Level 4	28.4	16.9	4844.0	347.2	277.7	4872.2

complex geometry. It starts from an axis-aligned bounding box of an object, and uses successive binary division along each coordinate direction to divide each box into eight sub-boxes. Each sub-box is marked as white, gray or black, respectively, if it is not, partially, or entirely filled by part of the actual object. The higher resolution level means better approximation of the actual geometry, with resolution level 1 having 8 voxels, level 2 having 64 voxels, etc. The quality of the final layouts for both algorithms is reflected by the objective function value and is comparable when the same resolution level of the octree models is used. A fixed resolution level is used throughout each run for both algorithms to compare the speed of the algorithms.

From the performance of the 8-cube packing problem as shown in Table 1, we can see that there are no significant differences in both the number of evaluations required for constrained or unconstrained versions, and that required for different resolution levels of octree models. In terms of run time needed, when the resolution of octree models goes one level higher, each voxel is divided into eight sub-voxels, thus making the most time-consuming evaluation—the intersection evaluation process—eight times longer in the worst case. Since the evaluation time for the other objective function terms is not influenced by the resolution level of octree models, the average run time for a one-level higher resolution model over that of a lower level resolution model is less than 8. For this problem it is about 5.3 times for both algorithms. However, the extended pattern search layout algorithm consistently generates layouts of comparable quality more than 30 times faster, requiring on average equally fewer objective function evaluations.

The quality of the layouts from the two algorithms is reflected by the respective objective function values. The objective values shown in Table 1, as well as in other tables to follow, are the comparison of the sum of multiple objective terms without considering the weights. Since a different set of weights is used for each algorithm to achieve better convergence, this is a concise way to evaluate the performance without listing all objectives and comparing them term-wise.

Table 2 shows the timing comparison for the 16-cogwheel packing problem. The run time of the extended pattern search algorithm is about 18–30 times faster than that of the simulated annealing-based algorithm for results with the same quality. This problem was designed with the cogwheels sized so that they can fit into the container only if their teeth intermesh. Previously, 16 cogwheels were chosen based on the effectiveness of the simulated annealing-based algorithm as described in Kolli et al. [21] by choosing octree resolution levels to generate quality layouts with reasonable run time. One snapshot of a final layout by the simulated annealing-based algorithm is shown in Fig. 2, with a two-dimensional view in Fig. 2(*a*) and a three-dimensional view in Fig. 2(*b*). A slight overlapping between the cogwheel teeth can

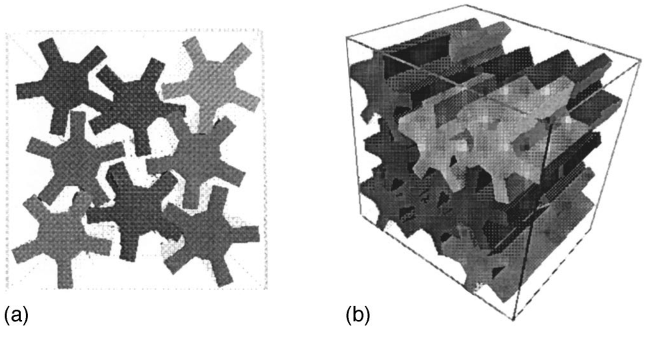

Fig. 2 (*a*) 16-Cogwheel packing by simulated annealing-based layout algorithm; (*b*) three dimensional view of two layers

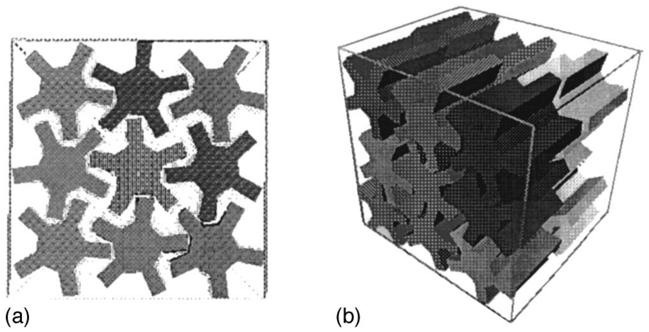

Fig. 3 (*a*) 18-Cogwheel packing by extended pattern search layout algorithm; (*b*) three dimensional view of two layers

be detected in Fig. 2(*a*), and that is because of the resolution level of the octree models used. It can be eliminated by using higher resolution models as is shown in Fig. 3(*a*).

We also applied a random search algorithm and a gradient-based algorithm to the same 16-cogwheel problem in order to compare these extremes with our current approaches. The random search algorithm uses the same number of evaluations required by the simulated annealing algorithm and it only accepts better states during search. Solutions are generated with 14 percent component volume intersection and 19 percent container protrusion, averaged over ten runs. The gradient-based algorithm is used with 20 random starting points. The best solution has 11 percent component intersection and 15 percent container protrusion. Compared to the solutions with no component intersection and 1 percent container protrusion by the extended pattern search algorithm and simulated annealing algorithm, these results support the need for such stochastic techniques to solve general layout and packing problems.

To illustrate the reduced run time from using hierarchical models, we compare the speed of using single resolution level models with that of using hierarchical models. The 16-cogwheel problem is solved by the extended pattern search layout algorithm. It takes 271.3 s using level 5 models and 173.5 s using hierarchical models which starts with level 3 models and moves gradually to level 5 models. A 36 percent saving on run time is achieved by using hierarchical models, while the layout quality is maintained the same.

Since the extended pattern search layout algorithm requires far fewer evaluations to converge and thus runs much faster than the simulated annealing-based algorithm, we can afford to go to a higher resolution level of octree models to approximate the cogwheel shapes more closely, while still keeping the run time at the same level. The higher resolution level contributes to a more accurate intersection evaluation, which in turn may lead to higher packing density. As a result, we used the same amount of run time as used by the simulated annealing-based algorithm and successfully packed 18 cogwheels of the same size into the same container by running the extended pattern search algorithm at a higher resolution level. Snapshots of the 2-dimensional and 3-dimensional views of a final packing are shown in Figs. 3(*a*) and 3(*b*).

The 64-cube packing problem is set up to test the scalability of the algorithm. As is shown in Table 3, the speed of the extended pattern search algorithm is more than one-order of magnitude faster than that of the simulated annealing-based algorithm, even though the step-jumps are allowed in this problem of larger size to slow down the convergence of the extended pattern search algorithm.

Figure 4 shows a snapshot of the final layout of the Saturn car engine compartment problem by the extended pattern search algorithm. The problem is used to illustrate a more realistic problem where the components are of complex geometry and several spatial constraints need to be satisfied. Table 4 shows the performance comparison of the algorithms on the problem; again the extended pattern search algorithm is about 15 times faster than the simulated annealing-based algorithm. The solution is not meant to be compared with the actual layout of a Saturn car since only a portion of the components are modeled and included in the example.

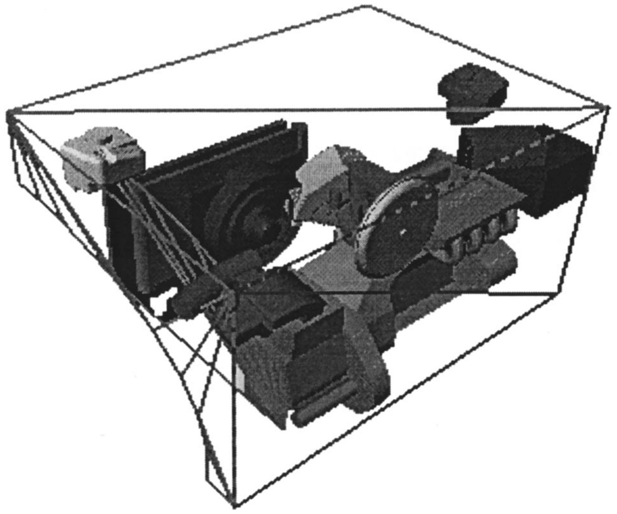

Fig. 4 Saturn car engine compartment layout by extended pattern search layout algorithm

6 Concurrent Layout and Routing Using Extended Pattern Search Methods

To illustrate the applicability of the extended pattern search algorithm to a wider range of problems, we extend the algorithm to solve a heat pump concurrent layout and routing problem. The problem description can be found in Szykman et al. [29]. Additional pattern search strategies are introduced and the layout objective function is augmented to incorporate routing measures.

The exploratory moves for the layout remain the same, but a set of exploratory moves for routing is added. The routing moves include adding a bend, removing a bend, and relocating a bend. The algorithm differs from the simulated annealing-based approach in Szykman and Cagan [19] in that we use a pattern method to relocate the bends. Again arbitrary initial routes are

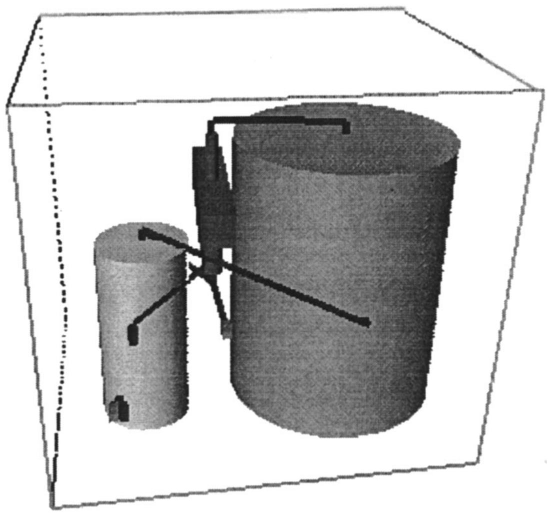

Fig. 5 Heat pump layout and routing by extended pattern search layout algorithm

Table 5 Performance comparison for heat pump layout and routing problem

	Extended Pattern Search Layout & Routing			Simulated Annealing-Based Layout & Routing		
Resolution	Evals (x1000)	Time (sec)	obj	Evals (x1000)	Time (sec)	obj
Level 3	2.0	1.2	1584.3	267.9	123.0	1938.3
Level 4	2.1	3.0	1855.2	276.1	260.1	1855.2

generated and the routes are randomly ordered. One bend in each route is picked each time and an exploratory move is applied to the bend to take the route to a new state if it is better than the original one according to an objective function. The routing moves and the layout moves are interlaced and the layout and routing measures are taken into account concurrently.

The objective function consists of routing cost terms such as route length and bend number, in addition to the usual terms for a layout problem. A snapshot of one of the final layouts by the extended pattern search algorithm is shown in Fig. 5, and the performance comparison for the two algorithms is shown in Table 5, indicating a two-order of magnitude improvement in run time with the extended pattern search algorithm for equivalent quality solutions.

7 Conclusions

In this paper, we introduced an extended pattern search algorithm for the layout and routing problem and demonstrated its power of generating good quality layouts quickly. We believe the approach to have a significant advantage over the current state-of-the-art by giving equivalent quality solutions one-to-two orders of magnitude faster or higher quality solutions with the same run time. In addition to applying pattern search to the layout problem we extended the basic algorithm by randomizing search orders, adding constraint related search directions, allowing occasional step jumps, strategically using swapping moves, and judiciously choosing hierarchical models.

Future work will explore alternative patterns and perturbations that affect performance of the algorithm. Pattern search methods employ heuristics to identify search directions; alternative heuristics may be introduced with domain-specific knowledge to guide the search and make the algorithm more efficient. Another area of future work is to incorporate more complex analyses required for industrial applications by taking advantage of the run time savings over the simulated annealing-based algorithm, and explore types of problems most suitable for solving using the extended pattern search algorithm.

8 Acknowledgment

The authors would like to thank the National Science Foundation under grant MIP-9420372 for supporting this work.

References

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

[2] Fujita, K., Akagi, S., and Hase, H., 1991, ''Hybrid Approach to Plant Layout Design Using Constraint Directed Search and an Optimization Technique,'' *Advances in Design Automation 1991: Proceedings of the 17th ASME Design Automation Conference*, **1**, pp. 131–138.

[3] Jajodia, S., Minis, I., Harhalakis, G., and Proth, J. M., 1992, ''CLASS: Computerized Layout Solutions Using Simulated Annealing,'' Int. J. Product. Res., **30**, pp. 95–108.

[4] Cagan, J., Degentesh, D., and Yin, S., 1998, ''A Simulated Annealing-Based Algorithm Using Hierarchical Models for General Three-Dimensional Component Layout,'' Comput.-Aided Design, **30**, No. 10, pp. 781–790.

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

[6] Torczon, V., and Trosset, M., 1997, ''From Evolutionary Operation to Parallel Direct Search: Pattern Search Algorithms for Numerical Optimization,'' Comput. Sci. Stat., **29**, pp. 396–401.

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

[8] Dyckhoff, H., 1990, ''A Typology of Cutting and Packing Problems,'' Eur. J. Oper. Res., **44**, pp. 145–159.

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

[10] Landon, M. D., and Balling, R. J., 1994, ''Optimal Packaging of Complex Parametric Solids According to Mass Property Criteria,'' ASME J. Mech. Des., **116**, pp. 375–381.

[11] Kim, J. J., and Gossard, D. C., 1991, ''Reasoning on the Location of Components for Assembly Packaging,'' ASME J. Mech. Des., **113**, pp. 402–407.

[12] Wodziack, J., and Fadel, G. M., 1996, ''Bi-Objective Optimization of Components Packing Using a Genetic Algorithm,'' AIAA-96-4022-CP, pp. 352-362, Paper presented at the *NASA/AIAA/ISSMO Multidisciplinary Design and Optimization Conference*, Seattle, WA

[13] Corcoran III, A. L. and Wainwright, R. L., 1992, ''A Genetic Algorithm for Packing in Three Dimensions,'' *Applied Computing: Technological Challenges of the 1990's—Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, Kansas City, KS, pp. 1021–1030.

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

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

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

[17] Smith, N., Hills, W., and Cleland, G., 1996, ''A Layout Design System for Complex Made-to-Order Products,'' ASME J. Mech. Des., **7**, No. 4, pp. 363–375.

[18] Szykman, S., and Cagan, J., 1995, ''A Simulated Annealing Approach to Three-Dimensional Component Packing,'' ASME J. Mech. Des., **117**, 2A, pp. 308–314.

[19] Szykman, S., and Cagan, J., 1996, ''Synthesis of Optimal Non-Orthogonal Routes,'' ASME J. Mech. Des., **118**, No. 3, pp. 419–424.

[20] Szykman, S., and Cagan, J., 1997, ''Constrained Three Dimensional Component Layout Using Simulated Annealing,'' ASME J. Mech. Des., **119**, No. 1, pp. 28–35.

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

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

[23] Lam, J., and Delosme, J., 1988, ''Performance of a New Annealing Schedule,'' *Proceedings of the 25th ACM/IEEE Design Automation Conference*, pp. 306–311.

[24] Hooke, R., and Jeeves, T. A., 1961, ''Direct Search Solution of Numerical and Statistical Problems,'' J. Assoc. Comput. Mach., **8**, No. 2, pp. 212–229.

[25] Lewis, R. M., and Torczon, V., 1996, ''Pattern Search Algorithms for Bound Constrained Minimization,'' TR 96-20, ICASE, NASA Langley Research Center, Hampton, VA 23681-0001, SIAM J. Optimization, **9**, No. 4, pp. 1082–1099.

[26] Lewis, R. M., and Torczon, V., 1996, ''Rank Ordering and Positive Bases in Pattern Search Algorithms,'' TR 96-71, ICASE, NASA Langley Research Center, Hampton, VA 23681-0001.

[27] Torczon, V., 1997, ''On the Convergence of Pattern Search Methods,'' SIAM J. Optimization, **7**, No. 1, pp. 1–25.

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

[29] Szykman, S., Cagan, J., and Weisser, P., 1998, ''An Integrated Approach to Optimal Three Dimensional Layout and Routing,'' ASME J. Mech. Des., **120**, No. 3, pp. 510–512.