Exploring the Effectiveness of Various Patterns in an Extended Pattern Search Layout Algorithm

Su Yin
General Electric Corporate R&D,
K1-2A68, P.O. Box 8,
Schenectady, NY 12301
e-mail: yinsu@crd.ge.com

Jonathan Cagan
Department of Mechanical Engineering,
Carnegie Mellon University,
Pittsburgh, PA 15213
e-mail: cagan@cmu.edu

Automated synthesis of product layout has the potential of substantially reducing design cycle time while allowing for quick check of interference, clearance, scale and fit prior to the building of physical prototypes. The search for optimal positions and orientations of parts in the layout typically requires a huge number of iterations. An extended pattern search layout algorithm based on coordinate search was introduced in an earlier paper and shown quite effective over the previous state-of-the-art. Coordinate search is a simple and straightforward way of implementing the extended pattern search method in the layout problem. However, it is not taking advantage of the wide variety of heuristics admissible in pattern search methods for identifying promising search directions. By introducing various search patterns and exploring their effectiveness in the layout problem, the question of whether complex tactics can do better than the basic coordinate pattern search is addressed. This paper presents four different heuristics for generating pattern directions in the extended pattern search layout algorithm: the conjugate direction method, the modified gradient method, the rank ordering method, and the simplex method. These heuristics are utilized to identify promising search directions and update the set of pattern directions used in the algorithm over iterations. The performance of the different heuristics is compared to that of the basic coordinate extended pattern search layout approach. [DOI: 10.1115/1.1641185]

1 Introduction

Product layout and packaging plays an important role in consumer product design and development. The placement of automobile parts in a new line of vehicles and the layout of electromechanical systems are examples of typical applications. With rapid advances of CAD modeling capability, digital CAD models are increasingly replacing traditional paper drawings and physical models in part design. To fully take advantage of digital product development and bring digital prototyping to the system level, an automated intelligent layout and packaging tool is highly desired.

The automated placement of 3-D components is usually modeled as an optimization problem, with locations and orientations of components as design variables and various packaging goals as objective functions. Design goals may include the higher packing density, the lower center of gravity, the lower routing and configuration costs, as well as performance measures from evaluation or simulation. Constraints such as component overlap, container protrusion, or spatial relationship violations are often added to the objective function as penalty terms.

The exploration of the layout space to find a good solution is not a trivial task. It has been shown that the placement of blocks of different sizes into a rectangular container belongs to the class of NP-hard problems [1]. Consequently it is highly unlikely that the global optimal solution can be obtained in an amount of time bounded by a polynomial in the size of the problem, resulting in prohibitive computation time for large problems. Research on 2-D circuit layout showed that the layout space is fractal-like [2], which means multiple local optima exist and simple gradient methods are not suitable for the global exploration of such a space to identify good solutions. Heuristic algorithms are often relied on to generate good and acceptable solutions in reasonable time instead of exhaustively searching for the marginally better global optimal solution.

A variety of algorithms have been used for the layout problem. Some of them place restrictions on types of layout goals that can be optimized or shapes of objects that can be dealt with. Others may require a huge number of iterations and lengthy parameter tuning process to achieve convergence. An extended pattern search layout algorithm based on the coordinate search pattern was introduced in Yin and Cagan [3,4] to solve the general layout problem efficiently. The performance of the extended pattern search layout algorithm was compared to that of a robust simulated annealing algorithm using an adaptive annealing schedule [5] and probabilistically modified move selection [6]. The extended pattern search algorithm consistently showed one-to-two orders of magnitude improvement in run time for results with the same quality on all the test problems. As expected, the method was also shown superior to both random and downhill search.

Pattern search methods are characterized by the nature of generating exploratory moves through different choices of pattern directions. The coordinate-based extended pattern search works well on the layout problem. However, it does not take advantage of the wide variety of heuristics admissible in pattern search methods for identifying promising search directions. Our hypothesis is that the number of iterations required to achieve convergence in the layout problem may be substantially reduced if suitably good search directions can be identified. However, additional computing cost is involved for the calculation and update of search directions. It would be interesting to find out whether complex tactics are able to do better in terms of efficiency than the simple coordinate extended pattern search method.

In this paper we introduce four types of heuristics for updating pattern directions in the extended pattern search layout algorithm and compare them with the basic coordinate-based extended pattern search approach. These patterns demonstrate a variety of different properties, based on traditional optimization algorithms, to more thoroughly explore the search potential. Traditionally, pattern search algorithms have used very simple methods to determine search directions. In this paper we look toward more sophisticated approaches that have succeeded in optimization algorithms to determine if that success can transfer to the pattern search algorithms. Although each of these is well understood as an optimi-

Contributed by the Design Theory and Methodology Committee for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received October 2000; revised June 2003. Associate Editor: D.C. Thurston.

zation approach, none have been used in the extended pattern search algorithm or for application in the layout problem.

The paper is organized as follows: Section 2 surveys different algorithms used for the layout problem. Section 3 presents the basic pattern search algorithm and its layout extensions. Section 4 describes four types of patterns based on different heuristics for choosing exploratory moves in the extended pattern search layout algorithm, their theoretical foundations and implementation details in the layout context. Section 5 summarizes the performance comparison of the extended pattern search method using different patterns. Section 6 draws conclusions.

2 Related Work

A number of algorithms are used for the exploration of the design space of three-dimensional layout tasks. Heuristic-based algorithms are often used in operations research for the packing of boxes into shipping containers [7]. The basis for heuristic-based algorithms is a set of rules derived from common sense or experiences to provide insights into mechanisms behind efficient packing. The rules are likely to be domain dependent and the possible combinations of rules may be large, making it difficult to know which combinations are likely to lead to effective packing for a specific problem. Gradient-based algorithms calculate the gradient of the objective function and then search in the negative direction of the gradient. Landon and Balling [8] and Kim and Gossard [9] used the method to solve packaging problems. Gradient-based algorithms can significantly reduce the computation time required to converge to an optimal solution by limiting the search to reasonable directions. However, multiple runs from different starting points may be necessary since the layout space is multi-modal and each run only finds the nearest local optimal solution with respect to the initial design. Further, the methods are problematic when the objective function includes evaluation or simulation terms with no explicit derivatives available.

Stochastic algorithms have been very successful in addressing the layout problem. Genetic algorithms (GAs) use coding schemes to map design variables (positions and orientations of components) into strings of symbols. Mutation and crossover operators are used to modify current designs and a fitness function evaluates and selects the designs to the next generation [10]. The strength of GAs lies in their robustness since they deal with a population of candidate solutions. The drawbacks include the long coding and decoding, the large number of function evaluations required to achieve an optimum, and the difficulty in determining the ratio of mutation and crossover operators for convergence. Simulated annealing (SA) algorithms [11] are stochastic optimization methods based on probabilistic hill-climbing. They have been successfully applied to 2-D circuit layout [12,13] as well as 3-D mechanical layout problems [14-20]. SA algorithms are flexible as well as powerful. The methods converge to good solutions only if a suitable cooling schedule is found and a huge amount of computing time has to be invested.

An extended pattern search layout algorithm was presented in Yin and Cagan [4] and its one-to-two orders of magnitude gain in speed over an SA-based layout algorithm showed promises for an efficient layout method. The method is based on traditional pattern search approaches in that moves are allowed along pattern directions only and a large step size is used early in the search and scaled down gradually over iterations. Several extensions were introduced to make the algorithm stochastic and prevent the solution from being trapped in inferior local optima. The method is summarized in the next section.

3 Basic Pattern Search Method and Its Layout Extension

Pattern search methods were introduced by Hooke and Jeeves [21] to solve curve-fitting problems. Torczon and Trosset [22] explicated the common structure and key features of the methods. Yin and Cagan [3,4] introduced several extensions to the basic method and applied it to the layout problem.

1. Find a step $s_k = \Delta_k P_k$ from exploratory moves (Δ_k, P_k);
2. If $f(x_k + s_k) < f(x_k)$, $x_{k+1} = x_k + s_k$;
3. Update (Δ_k, P_k);
4. Iterate until $\Delta_k < \Delta_{min}$.

Fig. 1 The basic pattern search method

3.1 Basic Pattern Search Method. Pattern search methods are a subset of direct methods that rely exclusively on the direct comparison of objective function values. The methods have strong ties to response surface methodology used in design of experiments where local models of objective functions are constructed and search patterns are proposed.

The basic pattern search method defines a set of move directions P_k (called the pattern directions) and a step length parameter Δ_k. From an initial design state, a trial move is attempted along each pattern direction. Any step that leads to a better state is accepted and the step length is reduced only if trial moves along all pattern directions fail to identify a better state. The search stops when the step length is smaller than a pre-specified minimum step length. The selection of pattern directions and the step length can vary for specific methods. The basic pattern search method outlined in Fig. 1 is adapted from Torczon and Trosset [22].

3.2 Coordinate-Based Extended Pattern Search (CEPS) Layout Algorithm. Pattern search methods possess some nice features that make them useful for the exploration of the nonlinear and multi-modal layout space. They are zero-order methods; thus no derivative information is required. The methods have good global behavior and the choice of the starting point does not have a big influence on the methods' ability to converge to a stationary point. However, extensions are necessary to make the general pattern search algorithm converge to good solutions instead of any stationary point. Also, the basic pattern search algorithm is made stochastic so that a variety of good solutions from multiple runs can be generated.

The basic objective formulation for the general layout problem can be written as:

Minimize

$$f = w_1 \sum_{i=1}^{n} V_i R_i + w_2 \sum_{i=1}^{n} \hat{V}_i \hat{R}_i + w_3 \sum_{i=2}^{n} \sum_{j=1}^{i-1} \left(\hat{V}_{ij} \hat{R}_{ij} + \frac{V_i V_j}{R_{ij}^2} \right). \quad (1)$$

The first term in the objective function attracts the components to stay inside the container and the second and the third terms penalize the amount of container protrusion and component overlap, respectively. V_i represents the volume of the ith component and R_i is the distance from the center of the component to the center of the container. $\hat{V}_i$ is the volume of the ith component that is outside of the container and $\hat{R}_i$ is the distance from the center of $\hat{V}_i$ to the center of the container. $\hat{V}_{ij}$ is the volume of intersection between the ith and the jth components, R_{ij} is the distance between the centers of the two components, and $\hat{R}_{ij}$ is the distance between $\hat{V}_{ij}$ and the jth component. w_i's are the weights for normalizing the terms. An octree representation [18] is used for fast intersection evaluation between components of complex shapes.

The basic coordinate-based extended pattern search method used in Yin and Cagan [4] alternates search directions along X-, Y- and Z- axes. Translational and rotational moves are used until no improvement can be made. Then the step size is adjusted and the swapping moves are attempted, which means the locations of pairs of components are exchanged if it leads to a better state. Five extensions are introduced to the basic pattern search method to help converge to good solutions and make the algorithm efficient. The five extensions are listed here and the justifications of using them can be found in Yin and Cagan [4]:

- The order of selecting components and pattern directions is randomized.
- The step size is generally reduced over iterations. However, step size may be increased for more thorough explorations.
- Swaps are used only if translations and rotations fail to identify a better solution, allowing substantial changes in the design state before another cycle of local explorations with translational and rotational moves.
- Search directions are constructed to take into account the constraints.
- The resolution levels of octree models used for interference evaluation are related to the step size for efficiency.

The coordinate-based extended pattern search layout algorithm is tested on a set of benchmark problems originally solved by a robust simulated annealing algorithm [20]. Compared to the probabilistic hill-climbing methods of simulated annealing, the extended pattern search method explores the space in a more restrictive manner and thus requires far fewer evaluations to converge. One-to-two orders of magnitude time-savings are achieved on the test problems for layouts of the same quality [4].

4 Extended Pattern Search (EPS) Layout Algorithm With Different Heuristic Bases

The coordinate-based extended pattern search layout algorithm alternates the search along each coordinate direction using the predetermined step size. The method is simple and robust: the set of pattern directions spans the 3-D space and does not degenerate. However, the coordinate-based pattern search does not explore alternative search directions which could be more effective. Four new types of heuristics are proposed here to update pattern search directions in the algorithm to reflect promising directions. The four pattern search heuristics are: the conjugate direction method, the modified gradient method, the rank ordering method, and the simplex method. These four types of heuristics are applicable to the general layout problem regardless of the objective function formulation and the types of component geometry. Further, the heuristics only require information about objective values to identify trial steps that are suitably good directions of descent. The conjugate direction method and the modified gradient method utilize the history of previous moves to build new search directions. The conjugate direction method is a zero-order method that directly uses the objective values, while the modified gradient method uses the finite-difference method to approximate the first-order information. The rank ordering method and the simplex method are based on the sampling of the design space at multiple points to estimate promising search directions. The two methods differ in the way the search directions are chosen and the steps are taken at the conclusion of the sampling point evaluations. Together these methods explore the effectiveness of the main aspects of many optimization search methods within the layout problem.

4.1 Conjugate Direction Extended Pattern Search Method. This variation of the extended pattern search layout method is based on Powell's conjugate direction method. A conjugate direction is generated to replace one of the original directions in the set after each iteration of coordinate search. Conjugate directions are used since they tend to be promising search directions which could lead to a reduced number of iterations required for convergence.

4.1.1 Powell's Method. Powell's method [23] is an efficient zero-order method based on the concept of conjugate directions. The method minimizes a quadratic function in n or fewer conjugate search directions, where n is the number of design variables. The basic concept of Powell's method is to search in n orthogonal directions and then generate a conjugate direction by connecting the first and last design points in the iteration, which becomes the $n+1$ search direction. The $n+1$ one-dimensional search needed to obtain each conjugate direction is defined as one iteration. Intuitively this newly generated direction is likely to be a promising direction, thus one of the orthogonal directions is replaced by the conjugate direction in the next iteration of search.

Each one-dimensional search in Powell's method requires several function evaluations to find the optimum along the line. If some search direction gains no improvement in the objective function evaluation, the subsequent search direction will not be conjugate. Also, after a few iterations the method has a tendency to generate nearly dependent directions, causing the search to halt before an optimum is reached. One modification to the method is to reset the directions to the original orthogonal set periodically. Another more complicated technique determines whether a new direction should be added after each iteration and which direction from the original set should be replaced based on the effectiveness of each move.

4.1.2 Conjugate Direction Extended Pattern Search (CDEPS) Layout Algorithm. This variation of the extended pattern search layout algorithm is to start with coordinate search directions with the fixed step size and update the direction set using the idea similar to that of Powell's conjugate direction method. The difference between the conjugate direction extended pattern search (CDEPS) method and Powell's method is that Powell's method uses the line search to find exact minima along n directions, while CDEPS only samples the positive and/or negative design points that are of the current step size away from the original design point. Although the quadratic interpolation or other line search methods can be used during the one-dimensional exploratory move in CDEPS, it is a tradeoff between the level of improvement achieved by the move and the number of function evaluations required. We find that pattern exploratory moves with the fixed step size are sufficient in providing information about the local contour. To work around the possible degeneration of search directions and ensure reasonable convergence, a modification to the basic direction-updating procedure is used following the recommendations from Powell [23]. The pattern direction that leads to the most improvement in an iteration becomes the candidate to be discarded and a newly generated direction is added to the direction set.

The procedure of the conjugate direction extended pattern search method is shown in Fig. 2. Only steps related to the core exploratory moves associated with CDEPS are described; other general pattern search-based layout approaches such as the randomized search order, the use of swap moves and step jumps still apply but are not discussed here to avoid repetition. A separate set of pattern directions is maintained and updated for each component.

4.2 Conjugate Gradient Extended Pattern Search Method. Gradient methods are usually more efficient than zero-order methods since they use more information on which to base optimization decisions. For the general layout problem, there is no explicit gradient information available, thus the finite difference method has to be used to supply the information. A variant of the extended pattern search layout method based on the first order conjugate gradient method is described in this section.

1. For $k = 1..n$, $s_k = \Delta_k P_k$, if $f(x_k \pm s_k) < f(x_k)$, $x_{k+1} = x_k +/- s_k$;
2. Find $m \in (1, n)$, so that $f(x_{m-1}) - f(x_m)$ is a maximum;
3. Generate a new direction $s_{n+1} = x_n - x_0$;
4. Let $f_1 = f(x_0), f_2 = f(x_n), f_3 = f(2x_n - x_0)$;
5. If $f_3 > f_1$, $x_0 = x_n$, use the original P_k in the next iteration, otherwise, $x_0 = 2x_n - x_0$, $P_k = [P_1, \ldots, P_{m-1}, P_{m+1}, \ldots, P_n, P_{n+1}]$.

Fig. 2 The conjugate direction extended pattern search method

1. For $k = 1..n$, $s_k = \Delta_k P_k$, evaluate $f_k = f(x_k + s_k)$ or $f(x_k - s_k)$;
2. Find $f_{min} = min(f_k)$ and $f_{max} = max(f_k)$, and estimate $\nabla f(X^q)$ and $|\nabla f(X^q)|$;
3. Calculate $S^q = -\nabla f(X^q) + \beta_q S^{q-1}$;
4. Move along S^q if it leads to improvement.

Fig. 3 The conjugate gradient extended pattern search method

4.2.1 Conjugate Gradient Method. The conjugate gradient method [24] is a modification to the steepest descent method so that it has an improved convergence rate. The steepest-descent method searches in the negative direction of the gradient of the objective function. The conjugate gradient method starts with the steepest descent direction. On subsequent iterations, the search direction is defined as

$$S^q = -\nabla f(X^q) + \beta_q S^{q-1}, \tag{2}$$

where S^q and S^{q-1} are the current and the previous search directions. $\nabla f(X^q)$ is the gradient of the function at the design point. β_q is a scalar parameter defined as

$$\beta_q = \frac{|\nabla f(X^q)|^2}{|\nabla f(X^{q-1})|^2}. \tag{3}$$

From the expression we can see that the information from the previous iteration is carried on to the current iteration by the scalar parameter β_q. The conjugate gradient method is conceptually similar to Powell's method, except that the gradient information is used instead of n (the number of design variables) unidirectional searches for a conjugate direction.

4.2.2 Conjugate Gradient Extended Pattern Search (CGEPS) Layout Algorithm. This variation of the extended pattern search layout algorithm uses the finite-difference method to estimate the gradient direction at each design point and calculates search directions based on the conjugate gradient method. A trial step along each coordinate direction is used to sample the design space in the neighborhood of the design point. The steepest descent direction and the relative rate of change due to the steps are then calculated. The initial search is along the steepest descent direction using the pattern step size, and the following search directions are adjusted using information from previous iterations as indicated by Eq. (2) and Eq. (3). A small step size is used for the sampling of the space to obtain the gradient information since the finite-difference method generates a suitably good direction of descent only if the step size is small. A larger step size is used for the moves along search directions to accelerate the convergence. Figure 3 outlines the core exploratory moves in the CGEPS layout algorithm. Here again the steps listed are for moving a single component in the layout space only. The extensions discussed in Section 3.2 still apply to this particular variation of the EPS layout approach.

4.3 Rank Ordering Extended Pattern Search (ROEPS) Method. Rank ordering extended pattern search was introduced by Lewis and Torczon [25] as a variant of the basic pattern search method. The method is based on a heuristic for approximating the direction of steepest descent and has the benefit of reducing the worst case of $2n$ objective evaluations to $n+1$ evaluations in each iteration. We introduce a version of rank ordering pattern search to the extended pattern search layout algorithm.

4.3.1 Rank Ordering Pattern Search. The rank ordering pattern search is motivated by the following heuristics. Suppose in addition to the objective value at the current design point, the objective values at n (the number of design variables) other neighborhood design points are also available. Then the direction from the point with the highest objective value to the point with the lowest objective value should be a crude estimate of the direction of steepest descent. Unlike the finite-difference approximation to the direction of steepest descent, the rank ordering method ignores the distances between the points.

The basic pattern search methods can require as many as $2n$ objective evaluations in the worst case. That means searching along n coordinate directions for both positive and negative moves before a step that produces an improvement in the objective value can be taken or the step size has to be decreased. In the rank ordering pattern search, only n independent points need to be evaluated before a move direction can be identified, which means $n+1$ objective evaluations if the evaluation along the generated direction is also counted.

The rank ordering pattern search method starts with the sampling of the space at $n+1$ points. The points are ranked according to their objective values. Three possible steps can be taken to move the sampling points in the space: a reflection step, an expansion step, or a shrink step. These steps are applied to the sampling points using the best sampling point as the pivot point. Figure 4 illustrates the three possible steps for a two-variable problem. Here b, s, and w represent the best, the second best, and the worst ranked sampling points, and b', s' and w' represent the new sampling points after the move. The new sampling points are

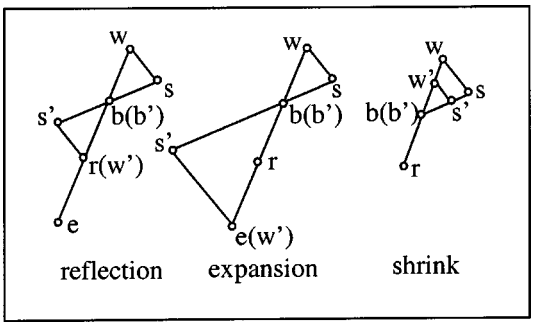

Fig. 4 The rank ordering pattern search moves

1. For $k = 1..n$, $s_k = \Delta_k P_k$, evaluate $f_k = f(x_k + s_k)$ or $f(x_k - s_k)$;
2. Rank ordering the sampling points to obtain the best point b and the worst point w;
3. Let $r = 2b\text{-}w$, evaluate $f(r)$;

If $f(r)<f(b)$, **the smaller objective, the better**
let $e = 3b\text{-}2w$, and evaluate $f(e)$;
else shrink step;

5. If $f(e)< f(r)$, expansion step; else reflection step;
6. Reorder the points and iterate.

Fig. 5 The rank ordering extended pattern search method

to be rank ordered before the next iteration continues. The search stops when the distances between the sampling points are sufficiently small.

4.3.2 Rank Ordering Extended Pattern Search Layout Algorithm. This variation of the pattern search layout algorithm associates the rank ordering patterns with different components. Four sampling points are defined for each component and the rank ordering method is used to move the sampling points in the design space until convergence. An example of the choice of sampling points is the initial design point plus one sampling point along each coordinate direction. The four points can be viewed as the vertices of a tetrahedral shape. The objective function is evaluated at each sample point and the points are ranked according to their objective values. Two additional points may need to be evaluated along the direction connecting the worst and the best sampling points in order to determine how the tetrahedral shape evolves. The two points are marked as r and e respectively in Fig. 4 to represent the reflection point and the expansion point. A set of branching rules are used to select the reflection, expansion, or shrink steps based on the relative objective values at point r, point e and the best sampling point as illustrated in Fig. 5.

4.4 Simplex Extended Pattern Search Method. This variant of the extended pattern search method is based on the Nelder-Mead simplex method. The simplex EPS method is similar to the rank ordering EPS method in that a sampling of space at multiple points is used to estimate promising search directions. The two methods differ in the heuristics used and the steps taken at the conclusion of each iteration.

4.4.1 Nelder-Mead Method. The Nelder-Mead method [26] is a direct search method that allows its simplex to adapt its shape to the local contours of the objective function. The initial simplex is constructed by sampling the space at the starting design point and n neighborhood design points. Five possible steps can be used to obtain the new simplex shape: an expansion step, a reflection step, an outside contraction step, an inside contraction step, and a shrink step. A two-variable simplex method and its five possible steps are depicted in Fig. 6.

The vertices of the initial simplex are labeled as b, s and w in the 2-D example to represent the best, the second-best, and the worst points in the simplex according to their corresponding objective values. A line is connected from w to m, the midpoint of b and s, and up to 4 additional points (point r, point e, point $c1$ and point $c2$) along the line may need to be sampled to determine the new simplex shape. The solid lines in Fig. 6 represent the simplex

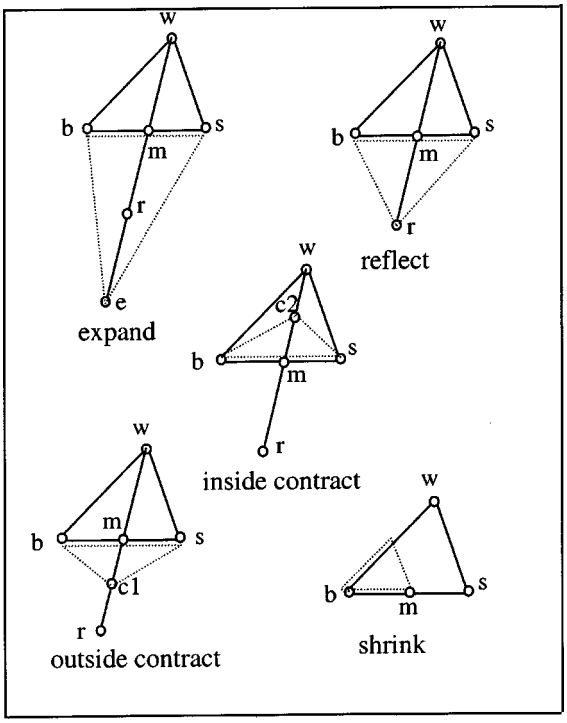

Fig. 6 The simplex extended pattern search moves

shapes before the steps are applied and the dashed lines represent the simplex shapes afterwards. The rules for selecting different steps are to ensure the new vertex in the simplex has a better objective value than the discarded point w. The vertices in the new simplex are evaluated and re-labeled at the conclusion of each step. The search continues until the stopping criteria are met, which means the vertices in the simplex are close enough to each other that the differences between the objective values at the vertices are negligible.

4.4.2 Simplex Extended Pattern Search (SEPS) Layout Algorithm. The simplex extended pattern search layout algorithm associates a simplex shape with each component and chooses the initial vertices of the simplex shapes in the same way as the rank ordering method does. The vertices are labeled as b (the best), s (the second best), t (the third best), and w (the worst) according to their objective values. The center of the triangle constructed by points s, t and w is labeled as point m. The point w is discarded at the end of each iteration in favor of one of the points resulting from the expansion, the reflection, the outside contraction, the inside contraction, or the shrink steps. The new simplex vertices are evaluated and labeled and the search iterates until convergence is achieved. The simplex EPS layout method is outlined in Fig. 7.

5 Performance

The performance of the extended pattern search layout algorithm using the four different heuristics presented in the previous section is tested on a set of benchmark problems originally solved by the basic coordinate-based extended pattern search algorithm. The set of test problems are briefly described as follows and the detailed problem description can be found in Cagan et al. [20]:

- Eight equally sized cubes to be packed into a cubic container of eight times volume;
- 16 cogwheels to be packed into a cubic container. The container is sized such that the cogwheels can all fit into the container only if their teeth intermesh;
- The layout of the major parts in an automobile engine compartment satisfying a set of spatial constraints.

1. Label the simplex vertices as b, s, and w, so that $f_b < f_s < f_w$;
2. Let $m = (b + s) / 2$, $L(\rho) = m + \rho (m - w)$, $r = L(1)$, $e = L(2)$, $c1 = L(0.5)$, $c2 = L(-0.5)$;
3. If $f_e < f_r$, expand; else if $f_r < f_b$, reflect;
4. If $f_b < f_r < f_s$, reflect;
5. If $f_s < f_r < f_w$ and $f_{c1} < f_r$, outside contract;
6. If $f_s < f_r < f_w$ and $f_{c1} > f_r$, shrink;
7. If $f_{c2} < f_w < f_r$, inside contract; else shrink;
8. Reorder the vertices and iterate.

Fig. 7 The simplex extended pattern search method

Figure 8 shows the sample solutions to the test problems. Table 1 summarizes the performances of the conjugate direction EPS method and the modified gradient EPS method on the test problems and compares them with that of the coordinate EPS (CEPS) method.

The model resolution level in Table 1 refers to the level of octree decomposition in the intersection calculation scheme [18]. Octree models start from the 3-D bounding box of a component and the box is recursively subdivided into eight octants. The octants are marked as full, partially full, or empty based on the spatial occupancy of the component and the intersection calculation checks whether two octants from a pair of components occupy the same space. Multi-resolution octree representations tradeoff accuracy of the intersection evaluation with the time taken for the evaluation and thus are used to speed up intersection evaluation.

The algorithm is coded in C++ and tested on a Silicon Graphics Indigo2 with 128 MB RAM and 195 MHz, MIPS R10000 CPU. The number of evaluations and the run time are averaged over ten runs on each test problem. The quality of the layouts by different methods is reflected by the objective values. The objective values are not listed in Table 1 since they are comparable when the same set of weights and the same resolution level of octrees are used.

From Table 1 we can see that both the conjugate direction EPS method and the modified gradient EPS method gain some improvement in run time over the coordinate based EPS because of the reduced number of iterations required to achieve convergence. A 17% to 38% saving on run time is achieved by the CDEPS method over the coordinate EPS on different test problems, and a 3% to 30% saving is achieved by the CGEPS method. The coordinate EPS method relies on chance or prior knowledge about the behavior of the objective function to choose a single trial step along each coordinate direction from which an improvement in the objective value is likely to be achieved. The CDEPS method starts with coordinate-based steps and creates a promising new search direction at the conclusion of each iteration of one-dimensional search. The CGEPS method uses the finite-difference method to estimate the gradients and utilizes the gradient information to make the search efficient.

The performance of the rank ordering EPS method and the simplex EPS method is listed in Table 2. The two methods failed to achieve good convergence and comparable solutions to the test problems with more than twice as many evaluations as that used by the coordinate EPS method. This result came as a surprise since theoretically the rank ordering method and the simplex method can reduce the worst case number of evaluations required in each iteration by almost half. Obviously it is the average cost of evaluations that is playing a role in the layout problem. The coordinate EPS method is not bad in terms of efficiency since no sampling of the space is necessary and the trial moves are mostly successful, especially in the beginning of the search when the layout is in infeasible states with a lot of room for improvement. The rank ordering EPS method and the simplex EPS method guarantee that a direction of steepest descent can be found after $n + 1$ evaluations in each iteration, but the price paid is the need to sample the space at multiple points before the direction and extent of search steps can be determined.

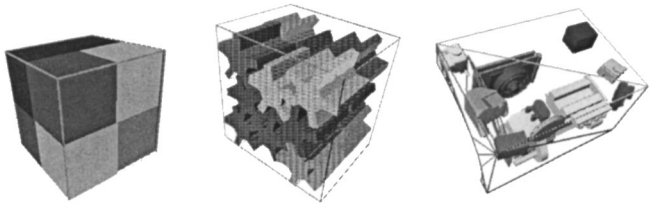

Fig. 8 Sample solutions to the test problems

6 Conclusions

Pattern search methods employ heuristics to identify promising search directions. In this paper, four different heuristics are introduced for the layout problem to guide the search in an extended pattern search method and make the algorithm more efficient. The four types of heuristics are based on the conjugate direction method, the modified gradient method, the rank ordering method, and the simplex method. The heuristics are tested on a set of layout problems and their performance is compared to that of the basic coordinate extended pattern search method. The results suggest that exploratory moves may supply information for the improvement of pattern directions. However, strategies that require a small number of function evaluations should be used to estimate

Table 1 Performance comparison between Coordinate Extended Pattern Search (CEPS), Conjugate Direction Extended Pattern Search (CDEPS) and Conjugate Gradient Extended Pattern Search (CGEPS)

	Coordinate EPS (CEPS)		Conjugate Direction EPS (CDEPS)			Conjugate Gradient EPS (CGEPS)		
Test Problem & Model Resolution	Evals (× 1000)	Time (sec)	Evals (× 1000)	Time (sec)	CDEPS/ CEPS in Time	Evals (× 1000)	Time (sec)	CGEPS/ CEPS in Time
Cube, Level 3	4.1	1.7	3.4	1.2	70%	4.1	1.6	94%
Cube, Level 4	4.4	9.0	3.3	5.6	62%	4.3	8.7	97%
Cogwheel,	36.7	18.9	32.4	14.6	77%	30.1	13.9	73%
Cogwheel,	39.5	67.7	33.9	53.8	79%	31.4	56.1	83%
Engine, Level 3	28.8	5.1	22.3	3.8	75%	21.2	3.6	70%
Engine, Level 4	28.4	16.9	23.3	14.1	83%	22.9	14.2	84%

Table 2 Performance comparison between Coordinate Extended Pattern Search (CEPS), Rank Ordering Extended Pattern Search (ROEPS) and Simplex Extended Pattern Search (SEPS)

Test Problem & Model Resolution	Coordinate EPS (CEPS)		Rank Ordering EPS (ROEPS)			Simplex EPS (SEPS)		
	Evals (× 1000)	Time (sec)	Evals (× 1000)	Time (sec)	ROEPS/ CEPS in Time	Evals (× 1000)	Time (sec)	SEPS/ CEPS in Time
Cube, Level 3	4.1	1.7	10.5	4.1	241%	13.4	5.2	305%
Cube, Level 4	4.4	9.0	11.7	20.3	226%	13.9	23.9	266%
Cogwheel,	36.7	18.9	72.0	45.2	239%	75.6	46.2	244%
Cogwheel,	39.5	67.7	65.2	158.6	234%	96.2	230.3	340%
Engine, Level 3	28.8	5.1	67.2	11.7	229%	56.3	12.8	251%
Engine, Level 4	28.4	16.9	66.1	52.9	313%	57.8	54.0	319%

promising search directions since there is a tradeoff between the savings on computing time from the reduced number of iterations to achieve convergence by searching along promising directions and the time spent on additional evaluations to obtain the new search directions. The number of search directions should be reduced to a minimal size while still maintaining a sufficiently rich set of search directions to capture the direction of steepest descent. While the result shows that the basic coordinate extended pattern search is sufficient to solve the layout problem and the heuristics introduced may achieve only moderate improvement in terms of efficiency, the 30% run time saving through the use of some of the heuristics can still have an appreciable effect on problems when the evaluation of objective terms involves complex analyses (e.g., one evaluation of a finite-element model can take minutes to hours). However, one of the most surprising results is the power of the basic coordinate approach, that adding in more sophistication in search can improve results, but not by a significant amount.

This work forms the foundation for research on alternative patterns for use within the extended pattern search method, and possibly the basic pattern search method as well. Future work will explore alternative types of heuristics, beyond those based on optimization strategies, which focus on the layout domains. As well, the effect of combining multiple heuristics at different stages of search to make more intelligent moves will be investigated.

References

[1] De Bont, F. M. J., Aarts, E. H. L., Meehan, P., and O'Brien, C. G., 1988, ''Placement of Shapeable Blocks,'' Philips J. Res., **43**, pp. 1–22.

[2] Sorkin, G., 1992, ''Theory and Practice of Simulated Annealing on Special Energy Landscapes,'' Ph.D Thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley.

[3] Yin, S., and Cagan, J., 1998, ''A Pattern Search-Based Algorithm for Three-Dimensional Component Layout,'' *Proceedings of the 24th ASME Design Automation Conference (DAC-5582)*, Atlanta, GA, September 13–16.

[4] Yin, S., and Cagan, J., 2000, ''An Extended Pattern Search Algorithm for Three-Dimensional Component Layout,'' ASME J. Mech. Des., **122**(1), pp. 102–108.

[5] 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.

[6] Hustin, M. D., and Sangiovanni-Vincentelli, A., 1987, ''TIM, a New Standard Cell Placement Program Based on the Simulated Annealing Algorithm,'' *IEEE Physical Design Workshop on Placement and Floorplanning*.

[7] Dowsland, K. A., and Dowsland, W. B., 1992, ''Packing Problems,'' European Journal of Operational Research, **56**, pp. 2–14.

[8] 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.

[9] 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.

[10] 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 NASA/AIAA/ISSMO Multidisciplinary Design and Optimization Conference, Seattle, WA.

[11] Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P., 1983, ''Optimization by Simulated Annealing,'' Science, **220**(4598), pp. 671–679.

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

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

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

[15] Szykman, S., and Cagan, J., 1996, ''Synthesis of Optimal Non-Orthogonal Routes,'' ASME J. Mech. Des., **118**(3), pp. 419–424.

[16] Szykman, S., and Cagan, J., 1997, ''Constrained Three Dimensional Component Layout Using Simulated Annealing,'' ASME J. Mech. Des., **119**(1), pp. 28–35.

[17] Szykman, S., Cagan, J., and Weisser, P., 1998, ''An Integrated Approach to Optimal Three Dimensional Layout and Routing,'' ASME J. Mech. Des., **120**(3), pp. 510–512.

[18] 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.

[19] Smith, N., Hills, W., and Cleland, G., 1996, ''A Layout Design System for Complex Made-to-Order Products,'' J. Eng. Design, **7**(4), pp. 363–375.

[20] Cagan, J., Degentesh, D., and Yin, S., 1998, ''A Simulated Annealing-Based Algorithm Using Hierarchical Models for General Three-Dimensional Component Layout,'' Comput.-Aided Des., **30**(10), pp. 781–790.

[21] Hooke, R., and Jeeves, T. A., 1961, ''Direct Search Solution of Numerical and Statistical Problems,'' J. Assoc. Comput. Mach., **8**(2), pp. 212–229.

[22] Torczon, V., and Trosset, M., 1997, ''From Evolutionary Operation to Parallel Direct Search: Pattern Search Algorithms for Numerical Optimization,'' Computing Science and Statistics, **29**.

[23] Powell, M. J. D., 1963, ''An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives,'' Comput. J., **7**, pp. 155–162.

[24] Fletcher, R., and Reeves, C. M., 1964, ''Function Minimization by Conjugate Gradients,'' Comput. J., **7**(2), pp. 149–154.

[25] 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.

[26] Nelder, J. A., and Mead, R., 1965, ''A Simplex Method for Function Minimization,'' Comput. J., **7**, pp. 308–313.