Objective Function Effect Based Pattern Search—Theoretical Framework Inspired by 3D Component Layout

Chandankumar Aladahalli
e-mail: chandan@alumni.cmu.edu

Jonathan Cagan[1]
e-mail: cagan@cmu.edu

Kenji Shimada
e-mail: shimada@cmu.edu

Department of Mechanical Engineering,
Carnegie Mellon University,
5000 Forbes Avenue,
Pittsburgh, PA 15213

Though pattern search algorithms have been successfully applied to three-dimensional (3D) component layout problems, a number of unanswered questions remain regarding their parameter tuning. One such question is the scheduling of patterns in the search. Current pattern search methods treat all patterns similarly and all of them are active from the beginning to the end of the search. Observations from 3D component layout motivate the question whether patterns should be introduced in some different order during the search. This paper presents a novel method for scheduling patterns that is inspired by observations from 3D component layout problems. The new method introduces patterns into the search in the decreasing order of a priori expectation of the objective function change due to the patterns. Pattern search algorithms based on the new pattern schedule run 30% faster on average than conventional pattern search based algorithms on 3D component layout problems and general 2D multimodal surface minimization problems. However since determining the expected change in objective function value due to the patterns is expensive, we explore approximations using domain information. [DOI: 10.1115/1.2406095]

Keywords: pattern search, 3D component layout, expected change in objective function

1 Introduction

The 3D component layout problem is a well studied [1] combinatorial optimization problem. Szykman and Cagan [2] defined 3D component layout as follows:

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 interfere (i.e., occupy the same space), while satisfying optional spatial and performance constraints on the objects.

Applications of the 3D component layout problem include layout of automobile engine compartments [3], automobile trunk packing for maximizing luggage space [4], transmission layout [5], layout and routing of a heat pump [3], and almost any mechanical or electro-mechanical machine whose components need to be placed to satisfy constraints and optimize one or more objectives.

3D component layout problems have been successfully solved by pattern search algorithms [3,6]. Pattern search algorithms are effective iterative methods for minimizing functions that are nonlinear, multimodal, and have no derivative information available [7]. They use patterns, which are search directions with varying step sizes, to explore a search space in a greedy manner. In 3D component layout, the patterns are the translations and rotations of components and step sizes are the amounts of translations and rotations.

Traditional pattern search methods for 3D component layout apply all available patterns throughout the search with their step sizes decreasing as the search progresses. But observations from 3D component layout show that patterns corresponding to different components affect the objective function by different amounts due to differences in component size and geometry. Small improvements due to moves of smaller components are wasted in the earlier stages of search since they are overridden by larger improvements due to bigger components. In the light of these observations the question arises whether it is useful to introduce patterns in some particular order rather than introducing all of them in the beginning of the search. To answer this question, we propose that patterns be introduced into the search in decreasing order of their effect on the objective function. The results presented in this paper show that doing so results in faster solutions to 3D component layout problems. Though inspired by 3D component layout observations, the new scheduling method is applicable to other domains where pattern search algorithms are used. This is demonstrated by the decrease in run time when pattern search with the new schedule is used for minimization of general 2D multimodal surfaces.

In this paper we measure the effect on the objective function value by computing the expected change in objective function value due to a move. Estimating the expected change in objective function is expensive, but it provides a means to validate our hypothesis that introducing patterns in decreasing order of their effect on the objective function is better than introducing them at the same time at the beginning of the search. Further, the effect on the objective function can be inexpensively estimated with similar algorithm performance by domain specific metrics, as briefly summarized in this paper and presented in detail in a companion paper [8].

The remainder of the paper is organized as follows. Section 2 presents observations from 3D component layout that motivate the question of scheduling patterns and our proposed solution. Section 3 discusses the 3D component layout problem and current pattern search methods in detail. Section 4 presents our new objective function-based pattern search (OPS) algorithm developed to answer the question of scheduling patterns. Section 5 presents an analytical comparison between the new and old algorithms. Sec-

[1]Corresponding author.

Contributed by the Design for Manufacturing Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received November 11, 2005; final manuscript received March 16, 2006. Review conducted by David Kazmer.

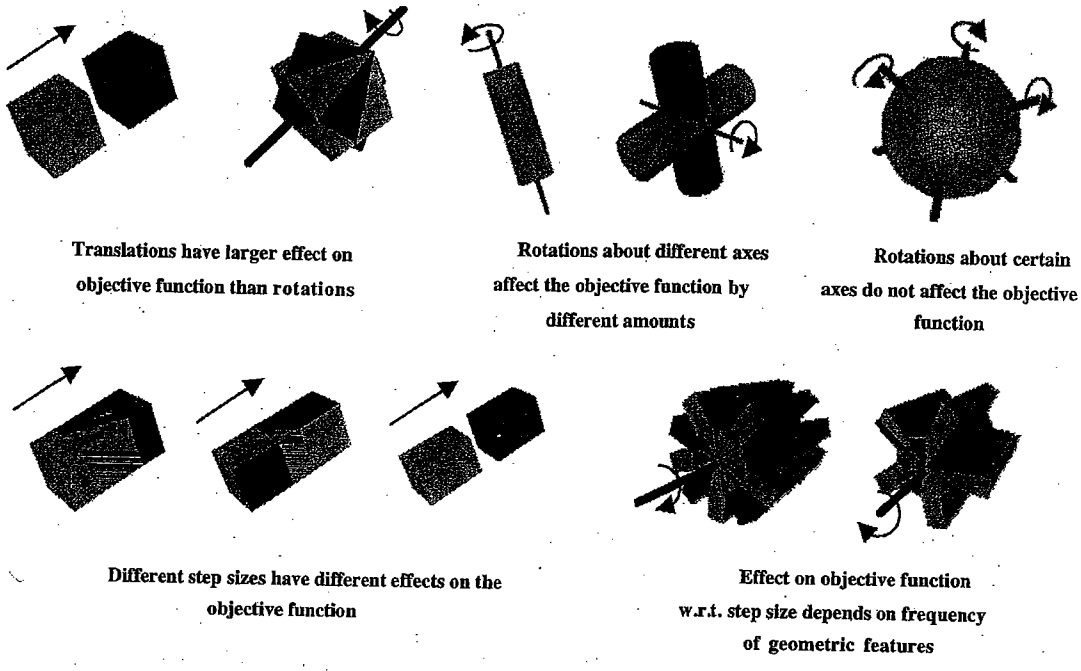


Fig. 1 Effect on objective function that is the sum of pairwise intersection volumes in 3D layout due to different patterns and step sizes

tion 6 presents results comparing traditional and the new pattern search algorithm. Section 7 presents concluding remarks and directions for future work.

2 Motivation and Hypothesis

Observations from using pattern search algorithms for 3D layout show patterns affect the objective function by different amounts. For example in 3D component layout, rotation of spheres does not affect the intersection volume at all. Rotating a cube may result in a smaller effect on typical placement objectives than translating it. Figure 1 illustrates how size and geometry of components determines the effect of patterns on an objective function such as the sum of pair wise intersection volumes in a layout. From the observations in the figure, the following two questions arise. First, in what order should patterns be introduced in the search? For example, while packing cubes into a larger cube is it better to start with translations and introduce rotations later instead of translating and rotating them at the same time? The current pattern search methods apply all patterns right from the beginning of the search.

The second question is whether pattern step sizes should be scaled by different factors instead of a single factor? For example, will a pattern search algorithm for 3D layout run faster if the translation step sizes are scaled slower than the rotation step sizes? Current pattern search methods scale step sizes of all patterns by the same factor. To answer the above two questions we hypothesize the following:

While minimizing a nonlinear discontinuous multimodal function using a pattern search algorithm it is efficient to introduce patterns into the search and adjust their respective step sizes such that the expected change in objective function due to the corresponding moves is similar and decreases as the search proceeds.

Patterns should be introduced into the search such that the algorithm seeks larger improvements in the objective function first and then seeks successively smaller improvements. At the same time in every iteration, it should be ensured that moves corresponding to the different patterns are chosen such that their effect on the objective function is similar. Our proposed algorithm, introduced in this paper, thus drives pattern search by decreasing the effect on objective function due to patterns rather than by simply decreasing step size of patterns. We call this algorithm OPS.

The key motivation behind the OPS algorithm is that it does not waste moves that result in smaller changes in the objective function earlier in the search. In the earlier stages of traditional pattern search algorithms, the smaller changes to the objective function are overridden by the larger changes. Hence moves that result in smaller changes are wasted. Also, in the traditional algorithms, smaller improvements earlier in the search force the algorithm to reapply moves of all the patterns again resulting in an increased number of iterations to converge. This is because these algorithms only decrease step size when none of the patterns result in an improvement in the objective function at the current step size. The proposed OPS algorithm avoids this by ensuring that at every iteration, all moves have the same effect on the objective function.

The following section discusses the relevant background of 3D component layout and current pattern search algorithms for 3D component layout. Section 4 then presents the proposed OPS algorithm.

3 Background

Yin and Cagan [3] first applied pattern search algorithms to the 3D component layout problem. The following subsections describe the 3D layout problem and pattern search algorithms in detail.

3.1 3D Component Layout. Many mechanical and electromechanical products are essentially a combination of functionally and geometrically inter-related components. The spatial location and orientation of these components affects a number of physical quantities of interest to the designer, engineer, manufacturer, and the end user of the product. Some examples of these quantities are compactness, natural frequency, ease of assembly, routing costs, and accessibility. 3D component layout concerns itself with determining the optimal spatial location and orientation of a set of components given some objective function and constraints. This objective function can include a quantification of a variety of measures such as the amount of cable used in the engine compartment of a car, or the packed volume in the trunk of a car, or the center

of gravity of a space vehicle. Constraints could include spatial relationships between components and between a component and the container.

3.1.1 Mathematical Model. The 3D layout problem is modeled as follows:

Minimize $O(x_1,x_2,..,x_n)$ subject to

$$l_i \leq c_i(x_1,x_2,..,x_n) \leq u_i, i = 1..m$$

where $x_1,x_2,..,x_n$ represent the coordinates of particular points on the different components along three independent axes and the orientations of the components about three independent axes. $c_1,c_2,..,c_m$ are functions representing spatial and performance constraints between the different components. The l_i's and u_i's represent the lower and upper bounds on the constraints, respectively.

The objective function $O(x_1,x_2,..,x_n)$, is decomposed into $I(x_1,x_2,..,x_n)$, $C(x_1,x_2,..,x_n)$, and $O'(x_1,x_2,..,x_n)$ as follows

$$O(x_1,x_2,..,x_n) = I(x_1,x_2,..,x_n) + C(x_1,x_2,..,x_n) + O'(x_1,x_2,..,x_n)$$

$I(x_1,x_2,..,x_n)$ is the sum of the pairwise intersection volume between components and the protrusion of the components outside the container. In stochastic layout algorithms, components are typically allowed to penetrate each other initially. The penalty term $I(x_1,x_2,..,x_n)$ pushes the components apart as the algorithm proceeds and thus tries to avoid intersections and protrusions.

$C(x_1,x_2,..,x_n)$ represents the constraint violation penalty. It is a measure of how much the constraints are violated. It is equal to zero if all the constraints are satisfied. The constraint violation penalty for distance constraints is the absolute difference between the distance mandated by the constraint and the actual distance in the violation [2]. Performance constraint violations are penalized in proportion to the amount of violation in the performance criteria. For example, if the natural frequency of a layout is constrained, then the penalty for violation of this constraint is the absolute difference between the actual frequency and the desired frequency [6].

$O'(x_1,x_2,..,x_n)$ represents other objectives of interest in packing such as packing density, length of routing, or any other quantity that can be numerically quantified.

3.1.2 Intersections and Protrusions. The most important component of the objective function for 3D component layout is the sum of the amount of pairwise intersections and the amount of protrusion outside the container, which is defined by $I(x_1,x_2,..,x_n)$. Evaluating these intersections and protrusions is very expensive if accurate polyhedral models of 3D objects are used. Instead octree models of objects are used to approximate the intersections between components and protrusions out of the container quickly [9].

3.1.3 Solution Technique. Many different stochastic search algorithms have been applied to the 3D layout problem [1]. These include genetic algorithms [10–13], simulated annealing [2,14–17], and extended pattern search (EPS) [3]. Extended pattern search is basically pattern search with extensions to make it stochastic.

The current algorithm of choice for the 3D layout problem formulated above is the EPS algorithm. The problem formulated in the previous section is discontinuous and has many local minima. Also the number of solutions grows exponentially with the number of components [6]. In light of the above facts, a near global minimum is sought instead of the global minimum. The extended pattern search algorithm is able to navigate discontinuities, escape from local minima, and explore the available solution space reasonably well before arriving at the near global minimum.

3.2 Pattern Search Algorithms. Pattern search methods are a subclass of direct search methods that utilize only direct comparisons of objective function values rather than derivative information. Direct search methods are especially well suited for problems where no gradient information is available. A variety of direct search methods have been developed and used over the past 50 years [7,18–20]. Torczon and Trosset [7] explicated the common structure and key features of these search methods and defined a general framework called the generalized pattern search (GPS) method. Torczon [21] also established a rigorous framework to mathematically deal with these direct search methods and proved their convergence. Pattern search developed mainly as a technique for numerical function minimization. Usually the function to be minimized consisted only of a few variables and was nonlinear. Yin and Cagan [3] first applied the pattern search algorithm to the 3D component layout problem. They introduced several modifications of the algorithm for 3D component layout, resulting in the EPS algorithm [3,6] discussed in Sec. 3.2.3.

3.2.1 Definitions. Torczon and Trosset [7] synthesized the various pattern search methods developed over the past 50 years into the common framework of the GPS algorithm. Following are some of the definitions used by Torczon to describe pattern search. The complete set of definitions can be found in Torczon [21]. Here only those definitions that are useful for understanding the pattern search framework are presented to highlight the essential differences between our new algorithm and traditional pattern search algorithms.

The objective of the pattern search algorithm is to minimize a function $f: R^n \rightarrow R$. R and Z denote the sets of real and integer numbers, respectively.

The *pattern matrix* defines the set of patterns. The *pattern matrix* is itself defined by two components, a *basis matrix* and a *generating matrix*. The basis matrix is any nonsingular matrix $B \in R^{n \times n}$, and therefore consists of linearly independent search directions. The generating matrix is of the form $C^k \in Z^{n \times m}$, where $m > 2n$. Here m and n represent the number of patterns and the number of search dimensions, respectively. Here the superscript k denotes the fact that this is the generating matrix for the kth iteration. The generating matrix linearly combines the independent vectors (search directions) defined by the columns of the basis matrix to define patterns. The generating matrix is further decomposed as follows

$$C^k = [M^k - M^k \; L^k] = [\Gamma^k \; L^k] \tag{1}$$

Here M^k represents a linearly independent set of vectors (patterns) that cover the search space. $-M^k$ makes sure that moves can be made in the negative direction of the patterns derived from M^k. L^k allows the definition of other interesting patterns in addition to the linearly independent patterns that can be derived from M^k and M^k [22]. It is also required that $M^k \in M^k \subset Z^{n \times n}$, where M is a finite set of nonsingular matrices, and that $L^k \in Z^{n \times (p-2n)}$ and contains at least one column, the column of zeros.

Any pattern p^k is then defined by the columns of the matrix $P^k = BC^k$. Also the partition of C^k in Eq. (1) is used to partition P^k as follows

$$P^k = BC^k = [BM^k - BM^k \; BL^k] = [B\Gamma^k \; BL^k] \tag{2}$$

By spanning R^n, the pattern matrix component $B\Gamma^k$ makes sure that all possible search dimensions are included among the set of patterns. BL^k is the set of other patterns that may be useful in exploring the solution space.

The scalar parameter, Δ^k, denotes the step size at any particular iteration k. Using the pattern matrix defined above and Δ^k, a *step* or *move*, s_i^k, is defined as follows

$$s_i^k = \Delta^k p_i^k \tag{3}$$

where p_i^k denotes the ith column of the pattern matrix P^k. Hence p_i^k determines the direction of the move and Δ^k defines the magnitude of the move.

1. Find a step s_i^k from $\Delta^k P^k$;
2. If $f(x^k + s_i^k) < f(x^k), x^{k+1} = x^k + s_i^k$, else $x^{k+1} = x^k$;
3. Update (Δ^k, P^k);
4. Iterate until $\Delta^k < \Delta^{\min}$.

Fig. 2 The generalized pattern search algorithm

3.2.2 Generalized Pattern Search Algorithm. The GPS algorithm uses the set of patterns P^k to explore the search space. For example moving two units along the x direction and one unit along the y direction is a possible pattern in 2D component layout. The magnitude of the steps is controlled by the step size control parameter Δ^k. The algorithm is described in Fig. 2 [21].

In the initial stages of the search the step sizes are large so that the algorithm can get to any region in the search space. As the algorithm proceeds the step size is decreased until a threshold step size is reached after which the algorithm terminates. At a given step size, a trial move is attempted along a pattern direction. Any step that leads to a better state is accepted. A trial move is then attempted again and so on. When all attempts to make a successful move at a step size have failed, the step size is reduced. The algorithm is discussed in detail in Ref. [21]. The extended pattern search algorithm is described next.

3.2.3 Extended Pattern Search Algorithm. Yin and Cagan [3] introduced extensions to the generalized pattern search algorithm to create the EPS algorithm and applied it to 3D layout. The extensions include randomized search orders, step jumps, swap moves, and a hierarchical objective function model. The extensions were introduced to add stochastic characteristics to the generalized pattern search algorithm when applied to the 3D component layout.

Previously pattern search techniques were recommended for problems with relatively few variables and with highly discontinuous and nonlinear objective functions [7]. Yin and Cagan [3] applied the pattern search algorithm to 3D layout, a problem with a large number of variables compared to the typical number of variables in nonlinear function optimization solved using the generalized pattern search algorithm. The typical 3D component layout problems we explored, have 10–100 components with six degrees of freedom each and hence have a dimension of about 60–600.

Extended pattern search for 3D layout has another deviation from the GPS algorithm. In EPS the translation and rotation patterns are independent groups. Therefore, there are two pattern matrices: one for translation and the other for rotation. This results in two independent GPS algorithms running simultaneously. Thus in this paper, in addition to comparing the GPS algorithm with our new algorithm, we modify the new algorithm to match this characteristic of the EPS algorithm to enable comparison.

4 The New Algorithm—Objective Function-Based Pattern Search (OPS)

The proposed algorithm is similar in spirit to the GPS algorithm but the driving force is now the effect on the objective function value and not the step size as in the GPS algorithm. While the GPS algorithm starts with moves of patterns with the largest step size, the new algorithm proposes to start with moves that have the largest effect on the objective function. The GPS algorithm decreases the step size of the patterns to create new moves as it searches the problem space. The new algorithm proposes to select patterns and corresponding step sizes such that their effect on the objective function decreases as it searches the problem space.

In this paper we measure the effect on the objective function as the expected change in objective function value. In essence, the expected change in objective function is the average change in the objective function that can be anticipated when a move is applied. Statistically, the expected change in objective function value due to a move is the average of the changes to the objective function when the move is applied from all possible locations in the search space. We denote this expected change in objective function value by E. Different patterns result in different values of E at different step sizes. The algorithm depends on a mapping of the patterns and corresponding step sizes to E. This mapping is denoted by $E(s_i, p_i)$, where s_i is the step size and p_i is the ith pattern and is defined using a statistical definition of expectation [23] as follows

$$E(s_i,p_i) = \int_{x_1=x_{1_{\min}}}^{x_1=x_{1_{\max}}} \int_{x_2=x_{2_{\min}}}^{x_2=x_{2_{\max}}} \cdots \times \int_{x_n=x_{n_{\min}}}^{x_n=x_{n_{\max}}} |f(x) - f(x+s_i p_i)| \rho(x) dx_1 dx_2 \ldots dx_n$$

where x is the solution vector comprising of the design variables $x_1, x_2, \ldots, x_n$ and $\rho(x)$ is the probability of solution x defined as the following uniform distribution

$$\rho(x) = \frac{1}{\prod_{i=1}^{i=n} (x_{i_{\max}} - x_{i_{\min}})}$$

The above definition defines the expected change in objective function value due to a move with step size x_i, of a pattern p_i as the average of the changes in the objective function value when is applied from all possible points in the search space. The above uniform probability distribution ensures that the changes in the objective function due to application of the move from all points in the search space are weighted equally.

The key idea behind the OPS algorithm is that it proceeds by decreasing E. A pattern is introduced into the search only when E falls below the maximum expected change in objective function value at its largest step size, $E(s_{i_{\max}}, p_i)$. Thus, patterns are introduced in the decreasing order of their effect on the objective function. Also, a decrease in E results in a decrease in step sizes of the moves, but at different rates for different patterns. The flowchart of the algorithm is presented in Fig. 3 and is contrasted with the GPS algorithm.

4.1 The OPS Algorithm—Overview. The algorithm begins with a user defined set of patterns (pattern matrix P_{span}) that span the search space. Next, the mapping between the step size of a pattern and the expected change in objective function value, $E(s_i, p_i)$, is determined for all patterns. This step is not required in the GPS algorithm since it is driven by decreasing step size and not decreasing expected objective function change.

Next, the algorithm starts with the largest possible expected change in objective function value, $E_{\max}$. It creates moves of patterns with step sizes that give $E_{\max}$ using the mapping $E(s_i, p_i)$. Next, these moves are applied repeatedly until the objective function ceases to improve. In contrast, the GPS algorithm starts with the largest step size and applies moves corresponding to all patterns at that step size.

Next, the desired expected change in the value of the objective function, E, is scaled down and moves that give this new E are applied. Once these moves cease to improve the objective function value E is scaled down further. This process continues until a threshold $E_{\min}$ is reached. In contrast the GPS algorithm successively scales down the step size of the patterns until a threshold step size, $s_{\min}$, is reached.

Due to the focus on decreasing E, the OPS algorithm differs from the GPS algorithm in two important ways. First, in a given iteration not all patterns are active, i.e., patterns that cannot affect the objective function by the amount E are not included in the pattern matrix. This is because there is an upper bound on the step size of the moves. This upper bound is usually imposed by the size of container or by the space in which the layout is carried out. This means that the set of moves in a particular iteration may not

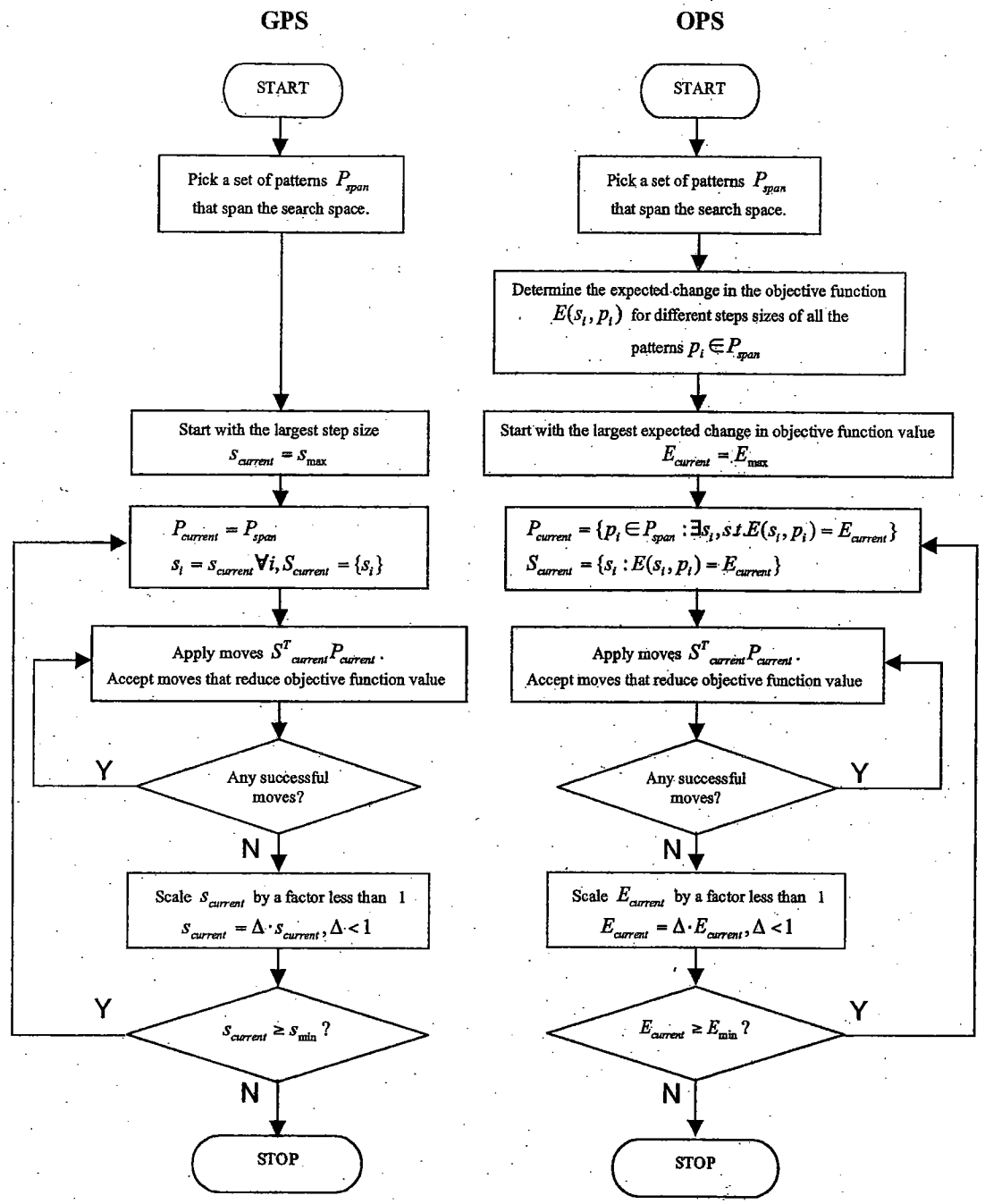


Fig. 3 GPS algorithm (left) and proposed class of OPS algorithms (right)

span the entire search space. In contrast, the GPS algorithm requires that at every step size, there be patterns corresponding to all the dimensions of the search space. This is required to satisfy the conditions placed on the pattern generating matrix C^k in Eq. (1). Therefore the step size can be decreased only when after perturbing all the dimensions of the search space an improved objective function has not been found. This is not guaranteed or required in the new algorithm.

For example, in a problem where the objective is intersection volume and where there are very big objects and very small objects, the early set of iterations in the OPS algorithm will generally not contain moves corresponding to the very small objects because their E is comparatively very small. Hence during initial stages of search the small objects are not perturbed. Even with the same object, such as a cube, big translation moves have higher E than rotational moves. Therefore, in this example, the early iterations will not have moves corresponding to the rotation of the cube.

The second difference between GPS and OPS is that GPS employs a single step size scaling parameter for all the patterns, whereas OPS allows the use of different step size scaling parameters for each pattern. Different step size scaling parameters may result from the rate of change of E for different patterns being different. For example, smaller objects may have a different rate of change of E compared to larger objects or the rate of change of E for rotations may be smaller compared to that of translations.

4.2 The OPS Algorithm—Determining the Mappings $E(s_i,p_i)$. The OPS algorithm uses the expected change in the objective function $E(s_i,p_i)$ due to the patterns as a measure of their effect on the objective function. This expected change $E(s_i,p_i)$ is computed empirically. A move is applied multiple times from different locations in the problem space (different locations and orientations of the components) and the absolute difference in objective function values before and after the move is averaged to arrive at the expected change in objective function value due to the move. The details of the computation are presented in the remainder of this subsection. We are aware that computing the expected change in objective function empirically is very expensive and offsets subsequent reduction in runtime by the OPS algorithm. It serves, however, to build a theoretical framework for scheduling patterns. It is used to validate our hypothesis presented in the previous section and demonstrate that it is indeed useful to schedule patterns based on their effect on the objective function. In a companion paper [8] we propose to substitute E by suitable domain specific metrics for a class of problems based on the performance found in this work. These metrics are quick to calculate but equally effective as the actual expected change in objective function. We developed one such metric in Ref. [24] that uses the amount of intersection between an object after a move is applied and the same object before the move is applied to drive pattern search. In this paper the general foundation is presented.

Given a function $f(x_1,x_2,..,x_n)$, the expected change in objective function value, $E(s_i,p_i)$ corresponding to pattern p_i is obtained as follows. First, $f(x_1,x_2,..,x_n)$ is reduced to a one-dimensional (1D) function $g_1(s)$ of the step size, s, of the pattern p_i as follows

$$g_1(s) = f(x^* + sp_i)$$

where x^* is the vector denoting fixed values of the variables $\langle x_1x_2..x_n\rangle^T$; p_i is the pattern of interest; and s is the step size of pattern p_i.

Next, this 1D function is sampled by applying the pattern at small step sizes successively and computing the objective function value.

This process is repeated for a large number (N) of random valid values for the variables $x_1,x_2,..,x_n$ to derive $g_1(s),g_2(s),..,g_N(s)$. Once $g_1(s),g_2(s),..,g_N(s)$ are available, $E(s_i,p_i)$ is computed as follows

$$E(s_i,p_i) = \frac{\sum_n \sum_s |g_n(s) - g_n(s+s_i)|}{N \cdot \|s\|}$$

where $\|s\|$ denotes the number of elements in the inner summation and N denotes the number of functions $g_1(s),g_2(s),..,g_N(s)$ generated from random starting points.

Since the patterns are linear combinations of independent search directions, a large step size is equivalent to the sum of successive smaller step sizes. Hence the change in objective function due to a step size, s_i, can be computed by finding the absolute difference between the function $g_i(s)$ at any point s and the same function evaluated at $s+s_i$. The change in objective function is thus computed for multiple functions $g_1(s),g_2(s),..,g_N(s)$ and an average taken to give the expected change in objective function $E(s_i,p_i)$ due to that step size.

The $E(s_i,p_i)$ thus mapped has one potential drawback. It might become a constant after a certain step size if the maximum step size is large compared to the size of the component. This is because 3D component layout objective functions are heavily weighed with the amount the intersections and the change in amount of intersection becomes constant once the component is displaced beyond a step size greater than its dimension. If the actual $E(s_i,p_i)$ becomes constant then two problems arise. First, there is no unique step size that gives the maximum value of expected change in objective function value. This could be resolved by picking the largest step size that gives the maximum expected change in objective function value. But doing so leads to another problem. When the desired expected change in objective function value is reduced from the maximum value, then there is a drastic reduction in step size. This leads to the loss of exploration over a big range of step sizes.

The above problem is resolved by approximating the $E(s_i,p_i)$ by a *power law relation* that ensures a strictly decreasing mapping between the step size and expected change in objective function value. A *power law relation* is defined as follows

$$E(s_i,p_i) = A_i \cdot s_i^{e_i}$$

where A_i and exponent e_i are constants specific to pattern p_i. A power law also translates into an elegant mapping between the rate of decrease of E and the step sizes corresponding to a pattern. In other words to scale E by a factor Δ, the step size s_i needs to be scaled by a factor Δ^{1/e_i}.

Figure 4 shows the objective function variation for two different patterns and their respective $E(s_i,p_i)$ along with the fitted *power law relation*. As seen in the figure, the power law is fitted by a linear interpolation on the log–log plot of the expected change in objective function versus the step size. For rotations where the expected change in objective function may be periodic, we fit the power law to the first instance of the period.

Since a power law is fit to the mapping $E(s_i,p_i)$, the computation of the mapping can be reduced to finding the expected change in objective function for the smallest and the largest step size of a pattern. This is done by finding the difference between objective function values before and after applying moves from a sufficient number (100) of random starting points. This reduces the amount of computation to determine the mapping.

Since the pattern step sizes have an upper bound, the expected change in objective function $E(s_i,p_i)$ of the patterns also has an upper bound. Different patterns have different upper bounds on their $E(s_i,p_i)$. The OPS algorithm makes use of these differences in upper bounds to start applying a pattern only when E has fallen below its upper bound. This is the first difference from the GPS algorithm mentioned Sec. 4.1.

Also, since the $E(s_i,p_i)$ approximations for the different patterns have different exponents e_i in their power law relations, their corresponding step sizes will be scaled down at different rates. This is the second difference from the GPS algorithm mentioned in Sec. 4.1.

5 Analytical Comparison of GPS and OPS

A theoretical comparison between the GPS and OPS algorithms can be made by deriving an analytical relation between pattern search parameters and the runtime. With some assumptions, the number of iterations taken by a pattern search algorithm can be derived as a function of the number of patterns, p, the number of step sizes, l, and the probability of a move being accepted, α.

THEOREM 5.1. *Given a set of p patterns, their respective step sizes, and the probability of acceptance (objective function improvement) of a move α, the expected number of iterations N_k, after which none of the patterns is able to improve the objective function is given by*

$$N_k = \frac{p}{(1-\alpha)^p}$$

Proof. Proof is presented in the Appendix.

Theorem 5.1. establishes that the number of iterations required on an average for a set of patterns to exhaust their search for improvements increases exponentially with the number of patterns. This is a classic combinatorial coupling effect since the success of one pattern results in the reapplication of all patterns. It is obvious that a reduction in the number of patterns can reduce the number of iterations required for them to terminate their

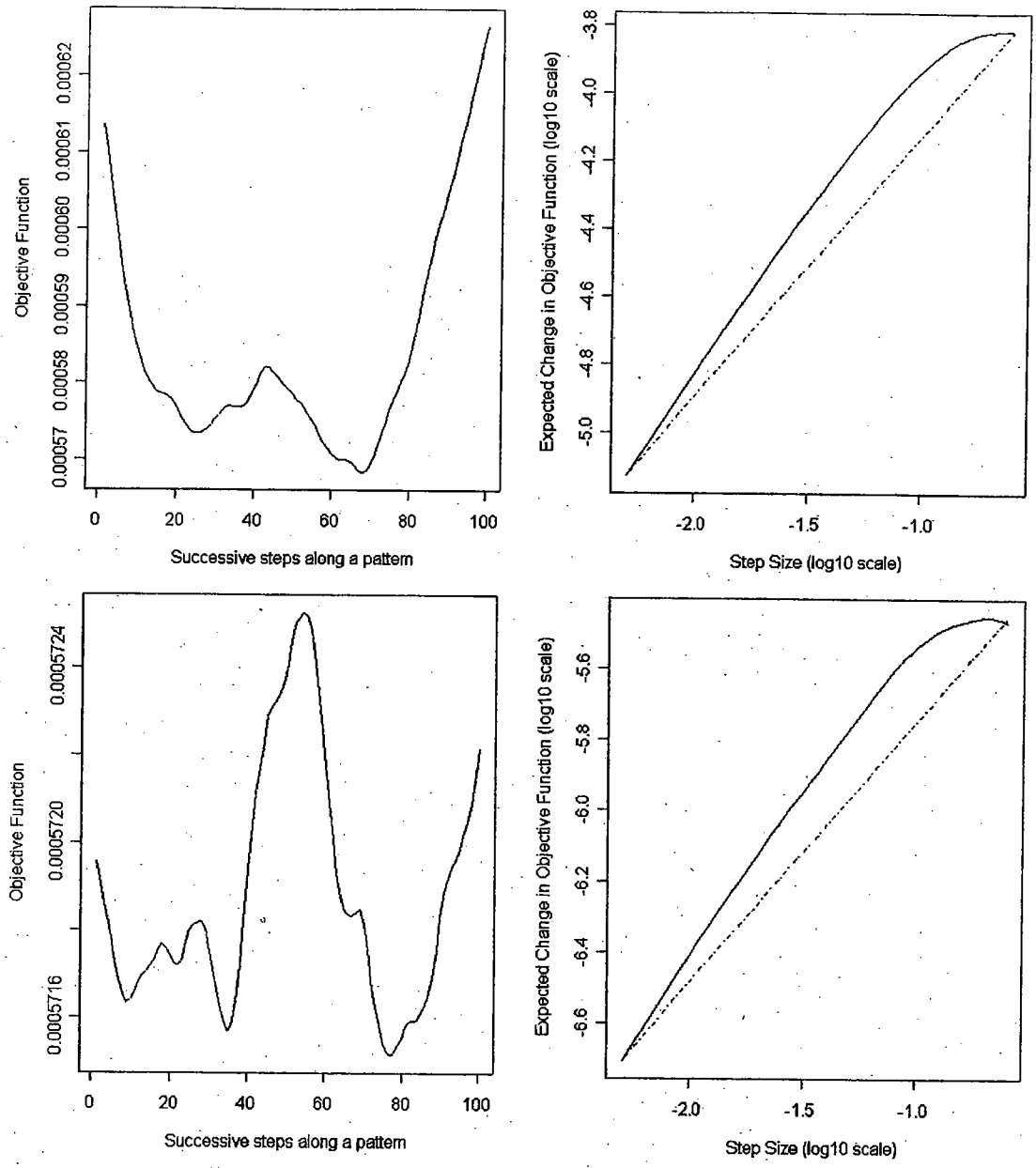


Fig. 4 Objective function variation with successive steps along a pattern (left) and the resultant expected change in objective function value as a function of the step size (solid lines) and fitted power law relation (dashed line) (right)

search. The OPS algorithm can be shown as using this reduction in number of patterns to reduce the number of iterations.

From Theorem 5.1, assuming that the probability of accepting a move, α, is the same at different step sizes of the patterns, the expected number of total iterations N_{GPS} for a GPS algorithm that applies the set of p patterns at l different step sizes is l times the expected number iterations at a single step size and is given by

$$N_{GPS} = \frac{pl}{(1-\alpha)^p}$$

The analysis can be extended to the OPS algorithm. Note that the OPS algorithm operates differently. Not all of the p patterns are active in the earlier stages of the search. Assuming that the probability of acceptance of a move is the same as that for the GPS algorithm we get the following relation for the expected number of total iterations

$$N_{OPS} = \sum_{t=1}^{t=1} \frac{p_t}{(1-\alpha)^{p_t}}$$

where p_t is the number of active patterns in the tth value of the desired expected change in objective function value in the OPS algorithm. It can be seen that $p_t \leq p \forall t$ since the number of patterns at each desired expected change in objective function value E is at most equal to the total number of patterns. Therefore $N_{OPS} \leq N_{GPS}$.

This analysis makes many assumptions such as a constant probability of acceptance across the different patterns, their step sizes and across the two algorithms, and the similarity of the objective function value obtained by the two algorithms. However, it provides an insight into how the OPS algorithm is able to reduce the total number of iterations by decoupling the patterns. The decoupling is particularly effective in the earlier stages of the OPS

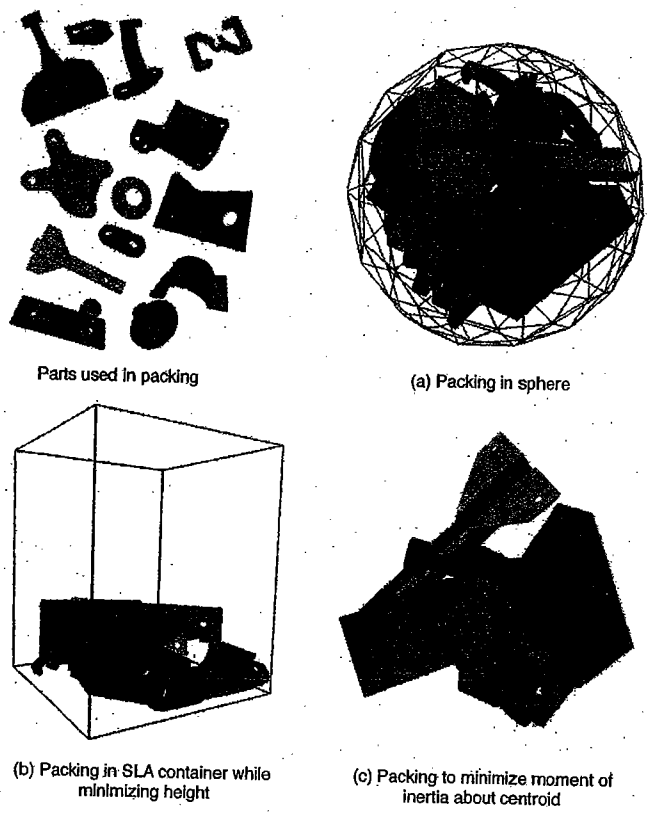


Fig. 5 The three examples used to compare the GPS and OPS algorithms

algorithm when only a few patterns that affect the objective function by a large amount are active and thus $p_i \ll p$.

6 Results and Discussion

The OPS and GPS algorithms were compared on three different 3D component layout problems. Also, to demonstrate the effectiveness of OPS for minimizing general multimodal functions, it was compared to GPS on 2D fractal surfaces. The results of the comparison are presented in the following subsections.

6.1 Results on 3D Component Layout Problems. Three different 3D layout spaces were used to compare the performance of the new OPS algorithm with the old GPS and EPS algorithms. The objective functions were: (1) sum of pairwise intersections and protrusions out of container; (2) sum of intersections and protrusions and the maximum height of the packing [25]; and (3) sum of intersections and protrusions and the second moment of the packing about the centroid of the packing. The three examples are shown in Figure 5. For each of the objective functions, two different problem instances were used for the comparison: one with 13 components and the other with 18 components in the packing. All components are arbitrary shaped and include real parts from different engineering sources.

The cuboidal container in packing problem (*b*) in Figure 5 represents a stereolithography apparatus (SLA) working volume. The ultimate goal of this problem is to develop a solution for the minimum height packing problem in SLA containers. The packing problem (*c*) in Figure 5 has no container. The objective in this problem was to minimize the second moment of inertia about the centroid of the packing. The second moment of inertia, I, is calculated as follows

$$I = \sum_{k=1..n} I_k$$

where n is the total number of components in the packing. I_k is the moment of inertia about the centroid of the packing for the kth component in the packing and is defined as follows

$$I_k = \int\int_V \int |l - l_c|^2 dV$$

where $|l - l_c|$ is the Euclidean distance between a point l and the centroid of the packing l_c.

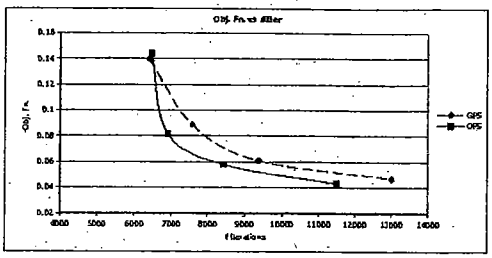
In the examples for packing in the SLA container and packing to reduce the second moment of inertia about the centroid, the intersection and protrusion component of the objective function was weighted such that the total amount of intersection and protrusion was less than 1% of the sum of volumes of the objects being packed.

The patterns used were translations along three axes and rotations about three axes of each component in the layout and were the same for all the algorithms. The stopping criterion for the GPS algorithm is a threshold step size, whereas the stopping criterion for the OPS algorithm was a threshold E. The threshold E was chosen such that the average step size of the patterns at this threshold is close to the threshold step size of the GPS algorithm.

Results of the new OPS algorithm and the old GPS algorithm are compared in Fig. 6. The three columns show the objective function values and number of iterations for two examples each of the three different packing problems. For each of the examples, the objective function and number of iterations is tabulated and plotted. In the table, the four rows each show the average objective function values and run times (number of iterations) for four different lengths of run. The length of a run affects the final quality of solution. The objective function values for the two algorithms were matched as close by as possible to make a comparison of the run times. This was done by controlling the number of steps in which the GPS and OPS algorithms decrease the step size and expected change in objective function value, E, respectively.

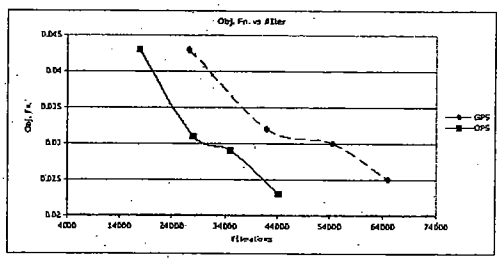
It should be noted that the gains in the runtime by the OPS algorithm as presented here are offset by the extra preprocessing to determine the effect on the objective function due to the moves.

0.140	6401	0.144	6466	-2.86	-1.02
0.089	7562	0.082	6901	7.87	8.74
0.061	9364	0.058	8421	4.92	10.07
0.047	13006	0.043	11487	8.51	11.68



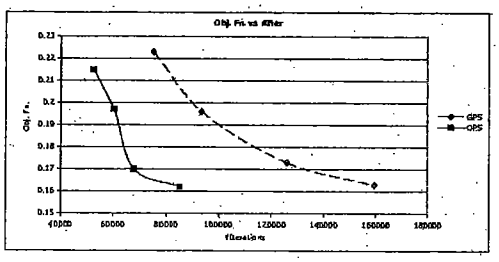
Packing in sphere - Example 1

0.043	27185	0.043	17776	0.00	34.61
0.032	41952	0.031	28041	3.13	33.16
0.03	54558	0.029	34972	3.33	35.90
0.025	64895	0.023	44225	8.00	31.85



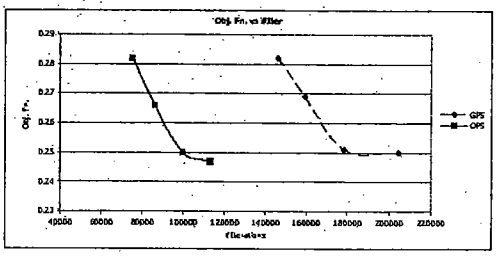
Packing in sphere - Example 2

0.223	75183	0.215	52433	3.88	30.26
0.196	93518	0.197	60189	-0.47	35.64
0.173	125836	0.17	67759	1.70	46.15
0.163	159415	0.162	85293	0.70	46.50



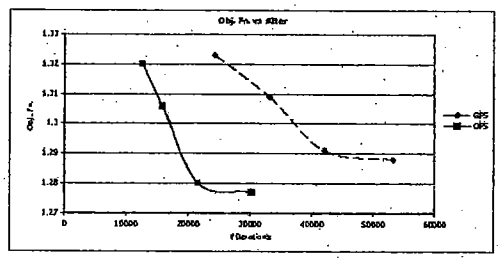
Packing in SLA container - Example 1

0.282	146109	0.282	75161	3.88	30.26
0.269	159326	0.266	86124	-0.47	35.64
0.251	178406	0.25	99635	1.70	46.15
0.25	204617	0.247	112932	0.70	46.50



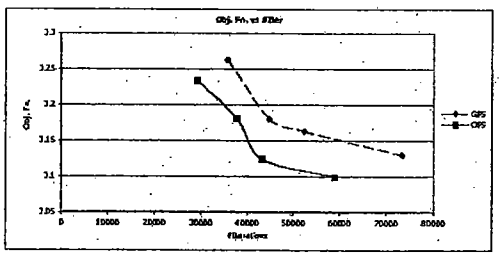
Packing in SLA container - Example 2

1.323	24319	1.32	12553	3.88	30.26
1.309	33078	1.306	15768	-0.47	35.64
1.291	42012	1.28	21543	1.70	46.15
1.288	53231	1.277	30188	0.70	46.50



Minimizing moment of inertia - Example 1

3.263	35812	3.234	29213	3.88	30.26
3.18	44876	3.181	37728	-0.47	35.64
3.163	52459	3.124	43267	1.70	46.15
3.13	73342	3.099	59042	0.70	46.50



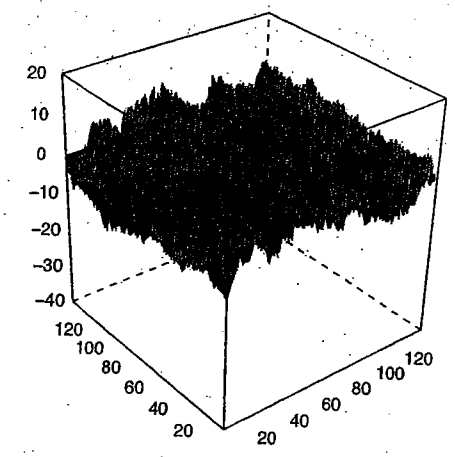
Minimizing moment of inertia - Example 2

Fig. 6 Results comparing the GPS and OPS algorithms for the three packing problems. Each of the three columns represents two examples each for the three packing problems.

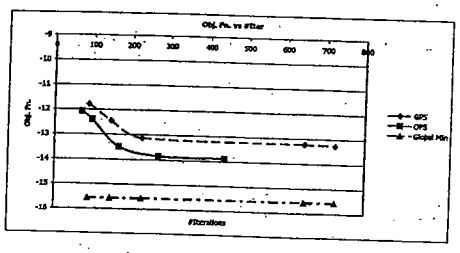
The results, however, demonstrate that scheduling patterns based on their effect on the objective function improves runtime. In a companion paper [8] we propose a means to estimate the effect on the objective function inexpensively by suitable domain specific metrics for a class of problems, enabling efficient use of this theory.

The results in Fig. 6 are based on 120 runs of each algorithm to generate solutions from different random initial configurations. The 120 runs are divided into 40 sets of three runs each. From each set of three runs, the best solution is picked. From the 40 best solutions thus picked, the average is taken to give the objective function values shown in Fig. 6.

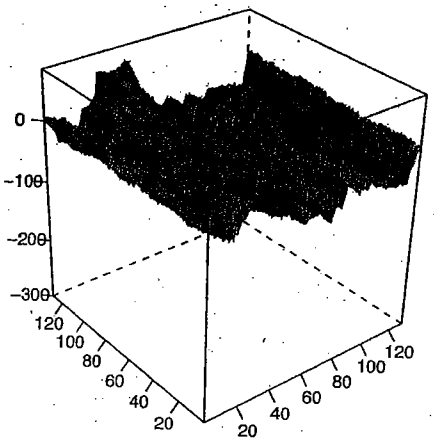
6.2 Results on 2D Fractal Surfaces. To test the applicability of the OPS algorithm on general multimodal objective functions, we generated 2D fractal surfaces and applied the GPS and OPS



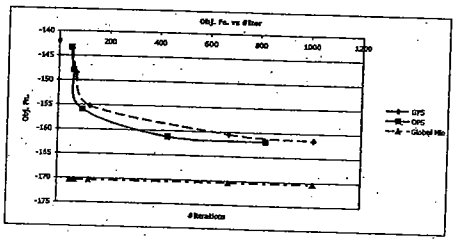
-11.79	86.86	-12.07	68.12	7.57	21.58
-12.44	145.46	-12.4	95.76	-1.15	34.17
-13.12	226.92	-13.48	165.5	14.55	27.07
-13.21	646.33	-13.84	269.75	26.92	58.27
-13.26	723.86	-13.89	439.79	27.28	39.24



$\sigma = 3; s = 0.1; e = 0.1$



-143.54	51.55	-143.32	48.95	-0.79	5.05
-148.44	67.98	-147.9	59.51	-2.47	12.45
-155.21	123.57	-155.93	94.28	4.76	23.71
-160.44	680.31	-161.23	437.18	7.98	35.74
-161.45	1014.75	-161.85	826.09	4.50	18.59



$\sigma = 3; s = 0.25; e = 0.25$

Fig. 7 Results comparing the GPS and OPS algorithms for the two fractal surfaces. The column on the left shows the actual surface.

algorithms to find the global minima of the surfaces. The objective function value at a location $(x\ y)$ is the height of the surface at that point. Fractal surfaces were chosen since a number of engineering optimization search spaces like 2D circuit layout [26], and 3D component layout [3] have been shown to have fractal properties. A 2D fractal surface is generated by the midpoint displacement of a simplex in 2D and then stretching and truncating this surface along one of the dimensions to make the expected change in objective function values along the two axes different.

The 2D fractal surface was generated as follows. The fractal surface is generated over a 2D array by splitting it into two triangles and generating the surface over them. The function values are defined as zero at the three vertices of a triangle. The triangle is then subdivided into four triangles by adding three new vertices at the midpoints of the edges of the original triangle. The function value at the midpoints of the edges is changed by an amount proportional to the distance between the end points by using a midpoint displacement method described in Ref. [27].

The function value at the midpoints is modified by an amount $\Delta = r \cdot s \cdot d^e$, where r is a random number with a Gaussian distribution with mean zero and variance σ, s is a scaling factor in (0, 1); d is the distance between the end points of the edges; and e is the power law exponent and is in (0, 1). The factor s controls the maximum expected change in objective function value and the factor e controls both the maximum expected change in objective function value and the slope of the expected change versus variable change mapping. The resulting triangles are similarly subdivided until the surface is defined over a sufficient number of points. The other triangle is similarly subdivided to complete a 2D array. The 2D array is then truncated by one-half along one direction. To make up for the truncated points along that direction, midpoints are added between the remaining points and the height value at these new points is the interpolated values between its neighbors.

The results comparing the OPS and GPS algorithms on two fractal surfaces are shown in Fig. 7. The five rows in the table on the right hand side show the objective function value reached and the number of iterations for five different lengths of run.

The results indicate that the new OPS algorithm reduces run time by an average of 30% to yield a similar quality solution. As mentioned in Sec. 3.2.3, the EPS algorithm for 3D component layout differs from the GPS algorithm since it groups the translation and rotation patterns separately. Therefore, to compare the new OPS algorithm with the EPS algorithm we separated the translation and rotation patterns in the OPS algorithm and compared it to the EPS algorithm using the same six 3D component layout examples discussed before. The results were similar to those comparing the OPS and GPS algorithms. It was found that the OPS algorithm with separated translations and rotations runs on average 30% faster than the EPS algorithm.

7 Conclusions

This paper proposed a new class of pattern search algorithms—OPS to schedule patterns in pattern search algorithms in decreasing order of their effect on the objective function value. The new algorithm uses a mapping between the step size of different patterns and the expected change in objective function value to drive the search. Moves that result in the biggest change in objective function are applied first, followed by moves with successively decreasing effect on the objective function. This algorithm reduces run time by an average of 30% when compared to current GPS.

The reduction in run time is because, unlike the GPS and EPS algorithms, the OPS algorithm does not waste moves that result in smaller changes in the objective function earlier in the search. In the earlier stages of search of GPS and EPS, the smaller changes to the objective function are overridden by the larger changes. Hence moves that result in smaller changes are wasted. This does not occur in the OPS algorithm because moves are applied in decreasing order of their effect on the objective function.

In this paper we used the expected change in objective function as a mapping between the step sizes of a pattern and their effect on the objective function. While this served to present a theoretical foundation and validate the new pattern schedule, it is computationally very expensive to implement in practice. Related work substitutes this explicit computation of expected change in objective function by a computationally inexpensive, 3D layout specific metric.

Acknowledgment

The authors would like to thank DaimlerChrysler Corporation for partial support of this research. This research effort was also partially sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under Grant No. F045-019-0006. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR or the U.S. Government.

Appendix

THEOREM 5.1. *Given a set of p patterns, their respective step sizes, and the probability of acceptance of a move α, the expected number of iterations N_k, after which none of the patterns is able to improve the objective function, is given by*

$$N_k = \frac{p}{(1-\alpha)^p}$$

Proof. The probability that the patterns are unable to improve the objective function after r applications of all the patterns is the product of the probability that none of the patterns improves the objective function in the last application and the probability that the previous $(r-1)$ applications of the patterns are successful. The former is given by

$$P(\text{none of the patterns succeed}) = (1-\alpha)^p$$

The latter is derived as follows

P(one or more patterns succeed)=[1

$-p$(none of the patterns succeed)]=[$1-(1-\alpha)^p$]and therefore

P(one or more patterns succeed in $r-1$ pattern set applications)

=[$1-(1-\alpha)^p]^{r-1}$Therefore

P(no more improvements after r pattern set applications) = (1

$-(1-\alpha)^p)^{r-1}(1-\alpha)^p$

Therefore the expected number of iterations after which the pattern set is unable to improve the objective function is given by

$$N_k = \sum_{r=1}^{r=\infty} rp(1-(1-\alpha)^p)^{r-1}(1-\alpha)^p$$

where rp is the number of iterations for r applications of p patterns and $[1-(1-\alpha)^p]^{r-1}(1-\alpha)^p$ is the probability that pattern sets are unable to improve the objective function after r applications of p patterns. Further, substituting $\beta=(1-\alpha)^p$

$$N_k = \sum_{r=1}^{r=\infty} rp[1-(1-\alpha)^p]^{r-1}(1-\alpha)^p$$

becomes

$$N_k = \sum_{r=1}^{r=\infty} rp(1-\beta)^{r-1}\beta$$

and moving the constants out of the summation we get

$$N_k = p\beta\sum_{r=1}^{r=\infty} r(1-\beta)^{r-1}$$

Let the summation part be called S. Therefore we have

$$N_k = p\beta S \tag{A1}$$

and

$$S = \sum_{r=1}^{r=\infty} r(1-\beta)^{r-1}$$

The summation component, S, in Eq. (A1) is solved as follows: Substituting

$$\gamma = 1-\beta \text{ in } S = \sum_{r=1}^{r=\infty} r(1-\beta)^{r-1}$$

yields

$$S = \sum_{r=1}^{r=\infty} r(\gamma)^{r-1}$$

This can be written as

$$S = \sum_{r=1}^{r=\infty} \frac{d}{d\gamma}(\gamma)^r$$

By a rearrangement of the derivative and summation we get

$$S = \frac{d}{d\gamma}\sum_{r=1}^{r=\infty} (\gamma)^r$$

which can be written as

$$S = \frac{d}{d\gamma}\left[\left(\sum_{r=0}^{r=\infty}(\gamma)^r\right)-1\right]$$

Since $0<\gamma<1$, we have

$$\sum_{r=0}^{r=\infty}(\gamma)^r = \frac{1}{1-\gamma}$$

Therefore