Pattern search methods for user-provided points*

Pedro Alberto[1], Fernando Nogueira[1], Humberto Rocha[2], and Luís N. Vicente[3]

[1] Departamento de Física, Universidade de Coimbra, 3004-516 Coimbra, Portugal
[2] Universidade Católica, Pólo de Viseu, 3504-505 Viseu, Portugal
[3] Departamento de Matemática, Universidade de Coimbra, 3001-454 Coimbra, Portugal

Abstract. We show how pattern search methods can be adapted to the optimization problem contexts where there are ways to provide points that can lead to an objective function decrease. The paradigm here is that it is the user and the optimization algorithm together, and not the optimization algorithm alone, that lead the calculation of new points. We are especially concerned with problems where objective function evaluations are expensive and for which parallel computing is available.

In this short paper we describe how pattern search methods for unconstrained optimization problems of the form

$$\min \ f(x), \quad x \in \mathbb{R}^n$$

can be applied when the user can and wishes to provide a routine to compute new points. An example of this situation arises in molecular geometry optimization, where new points can be provided by the user by applying physically relevant geometrical transformations to the current configuration. These geometrical transformations can potentially lead to a decrease in the objective function, i.e., in the total energy of the cluster of atoms that is being considered [1].

Pattern search methods exhibit enough flexibility to accommodate the user-provided point calculation. The main idea is to use patterns that fill the space surrounding the current iterate with a reasonable distribution of pattern points and pattern directions. In this way, a new point calculated by the user can be projected onto the pattern in such a way that the projected pattern point is reasonably close to the point provided by the user. The objective function is only evaluated at the projected pattern point and not at the user-provided point – an important requirement to regularize the overall algorithm and guarantee convergence properties [1]. The pattern must also be defined so that the linear algebra involved in the projection can be cheaply computed.

We will provide a very brief introduction to pattern search methods, introducing only the notation necessary to describe the accommodation of user-provided

* This work was supported by FCT (under the grant Praxis/P/FIS/14195/1998), by Centro de Física Computacional, and by Centro de Matemática da Universidade de Coimbra.

points. The reader is referred to the papers [2, 3, 5, 7] for motivation, theory, and other related material on pattern search methods.

If the iteration k of a pattern search method is *successful*, the next iterate must provide a decrease in the objective function: $f(x_{k+1}) < f(x_k)$. Pattern search methods iteratively generate points in an integer lattice (pattern), visiting at each iteration k a subset of the pattern called the *mesh* M_k. The mesh M_k can be defined through a set $\mathcal{V}$ of m positive bases[1] and a mesh size parameter $\Delta_k > 0$ in the following way:

$$M_k \ = \ \{x_k + \Delta_k \mathcal{V} z : \ z \in W \subseteq \mathbb{Z}^{|\mathcal{V}|}\},$$

where $|\mathcal{V}|$ is the sum of the number of vectors in all positive bases. The choice we actually made in our implementation of the pattern search methods for user-provided points is

$$W = \{ne_i : n \in \mathbb{N}, \ i = 1, \ldots, |\mathcal{V}|\},$$

where e_i is the i-th column of the identity matrix of order $|\mathcal{V}|$. Choices for $\mathcal{V}$ are described and discussed in [1].

The mechanism of pattern search methods consists of two steps at every iteration. In the first step, called the *search step*, a finite search is performed on the mesh, with the goal of finding a new iterate that decreases the value of the objective function at the current iterate. This step, called the *search step*, searches only a finite number of points in the mesh. The search step provides the flexibility for a global search, and influences the quality of the local minimizer or stationary point found by the method [1, 4, 6]. If the search step is unsuccessful, a second step, called the *poll step*, is performed around the current iterate with the goal of decreasing the objective function.

The poll step follows stricter rules and appeals to the concept of positive bases. In this step the candidate for a new iterate x_{k+1} is chosen in the *mesh neighborhood* around x_k

$$\mathcal{N}(x_k) \ = \ \{x_k + \Delta_k v : \ \text{for all } v \in V_k(x_k)\},$$

where $V_k(x_k)$ is a positive basis chosen from the finite set $\mathcal{V}$ of positive bases. The poll step attempts to perform a local search in a mesh neighborhood that, for a sufficient small mesh parameter Δ_k, is guaranteed to provide an objective function reduction, unless the current iterate is at a stationary point. If the poll step also fails, then the mesh parameter Δ_k must be decreased.

Pattern search methods for user-provided points can now be described for use in a parallel environment where, say, N_p processors are available.

Algorithm 1 (Pattern search methods for user-provided points)

0. Initialization Choose a rational number $\tau > 1$ and an integer number $m_{max} \geq 1$. Choose $x_0 \in \mathbb{R}^n$ and $\Delta_0 \in \mathbb{R}_+$. Set $k = 0$.

[1] A positive basis for $\mathbb{R}^n$ can be defined as a set of nonzero vectors of $\mathbb{R}^n$ whose nonnegative combinations span $\mathbb{R}^n$, but no proper set does. It can be shown that every positive basis has between $n + 1$ and $2n$ elements.

1. Search step (in current mesh)

1. For each processor p in $\{1, \ldots, N_p\}$:
   (a) Obtain a point u_{k+1}^p from the user.
   (b) Compute $x_{k+1}^p = x_k + \Delta_k \mathcal{V} z^p$, where z^p is the optimal solution of the integer programming problem

$$\min_{z \in W} \|u_{k+1}^p - (x_k + \Delta_k \mathcal{V} z)\| . \tag{1}$$

   (c) Evaluate f on the mesh point x_{k+1}^p.
2. If

$$\min_{p \in \{1,\ldots,N_p\}} f(x_{k+1}^p) \; < \; f(x_k) ,$$

   then set

$$x_{k+1} \; = \; \text{argmin}_{x_{k+1}^p} f(x_{k+1}^p) ,$$

   and go to step **3**, expanding M_k (search step and iteration are declared successful).

2. Poll step (in mesh neighborhood given by the positive basis)

This step is reached only if the search step is unsuccessful.

1. Obtain a point u_{k+1} from the user.
2. Determine $v_k \in \mathcal{V}$ such that

$$\frac{\langle u_{k+1} - x_k, \, v_k \rangle}{\|u_{k+1} - x_k\|} \; = \; \max_{v \in \mathcal{V}} \frac{\langle u_{k+1} - x_k, \, v \rangle}{\|u_{k+1} - x_k\|} . \tag{2}$$

3. Set $V_k(x_k)$ to the positive basis in $\mathcal{V}$ that contains v_k, and then set $\mathcal{N}(x_k) = \{x_k + \Delta_k v : \text{ for all } v \in V_k(x_k)\}$.
4. List the points in $\mathcal{N}(x_k)$ by increasing order of the values of the angles between $u_{k+1} - x_k$ and the corresponding vectors in $V_k(x_k)$.
5. Following the list given above, divided in groups of N_p points, start evaluating in parallel the function f in $\mathcal{N}(x_k)$.
   Stop if a point $x_{k+1} \in \mathcal{N}(x_k)$ is found such that $f(x_{k+1}) < f(x_k)$. In this case go to step **3**, expanding M_k (poll step and iteration are declared successful).
   If $f(x_k) \leq f(x)$ for every x in the mesh neighborhood $\mathcal{N}(x_k)$, go to step **4**, shrinking M_k (poll step and iteration are declared unsuccessful).

3. Mesh expansion (at successful iterations) Let $\Delta_{k+1} = \tau^{m_k^+} \Delta_k$ (with $0 \leq m_k^+ \leq m_{max}$). Increase k by one, and move to step **1** for a new iteration. (The value of $\tau^{m_k^+}$ can be chosen according to user-provided information.)

4. Mesh reduction (at unsuccessful iterations) Let $\Delta_{k+1} = \tau^{m_k^-} \Delta_k$ (with $-m_{max} \leq m_k^- \leq -1$). Increase k by one, and move to step **1** for a new iteration. (The value of $\tau^{m_k^-}$ can be chosen according to user-provided information.)

We point out that due to our choice of W, problems (1) and (2) are easily solved in the order of $|\mathcal{V}|n$ floating point operations.

It is also important to note that by listing the points in $\mathcal{N}(x_k)$ using the order suggested in step 2.4, the poll step starts by evaluating f in the points of $\mathcal{N}(x_k)$ closer to u_{k+1}.

Although we have described a parallel algorithm, a serial version of the algorithm is a straightforward adaptation of the parallel version. Both versions have been implemented in Fortran 95. The parallel version uses the parallelization protocol MPI. The codes and their documentation can be downloaded from the web site:

```
http://www.mat.uc.pt/~lvicente/psm/
```

The user must provide a routine to compute new points, and a routine to evaluate f at points specified by the algorithm. The calling sequences of these two routines are currently coded in the following form:

```
SUBROUTINE func( n, xk, f )

SUBROUTINE trial_point( n, xk, n_i_userpar, i_userpar, &
                        n_r_userpar, r_userpar, xtrial )
```

The output parameters are `f` and `xtrial`. In the routine `trial_point`, the user is given some information, stored in the integer vector `i_userpar` and in the real vector `r_userpar`, to indicate the amount of effort that can be put in the calculation. For instance, it is natural in the search step to ask the user to make a trial point calculation greedier or less conservative than in the the poll step.

References

1. P. Alberto, F. Nogueira, H. Rocha, and L. N. Vicente. Pattern search methods for molecular geometry problems. Technical Report 00-20, Departamento de Matemática, Universidade de Coimbra, 2000.
2. C. Audet and J. E. Dennis. Analysis of generalized pattern searches. Technical Report TR00-07, Departament of Computational and Applied Mathematics, Rice University, 2000.
3. C. Audet and J. E. Dennis. Pattern search algorithms for mixed variable programming. *SIAM J. Optim.*, 11:573–594, 2001.
4. W. E. Hart. Comparing evolutionary programs and evolutionary pattern search algorithms: A drug docking application. In *Proc. Genetic and Evolutionary Computation Conf.*, 1999.
5. R. M. Lewis and V. Torczon. Rank ordering and positive bases in pattern search algorithms. Technical Report TR96-71, ICASE, 1999.
6. J. C. Meza and M. L. Martinez. On the use of direct search methods for the molecular conformation problem. *Journal of Computational Chemistry*, 15:627–632, 1994.
7. V. Torczon. On the convergence of pattern search algorithms. *SIAM J. Optim.*, 7:1–25, 1997.