A PARALLEL SIMPLEX OPTIMIZER AND ITS APPLICATION TO HIGH-BRIGHTNESS STORAGE RING DESIGN *

H. Shang†, M. Borland, Argonne National Laboratory, Argonne, IL 60439, USA

Abstract

An efficient parallel Simplex optimizer was developed that can on run Solaris and Linux clusters. It can optimize the result of running essentially any program or script that returns a penalty function value. We have used this optimizer with `elegant` to optimize the dynamic aperture of storage ring designs. This paper discusses the optimization algorithm and performance, design of penalty functions, optimization results, and applications in storage ring design.

INTRODUCTION

Optimization is commonly used in accelerator design to find linear optics solutions. Such optimizations are usually fairly fast as linear optics computations are themselves fast. For high-brightness storage rings, optimization of nonlinear elements (e.g., sextupoles) is also important in obtaining sufficient dynamic aperture. However, such optimization can be very time consuming as the basic computations are time consuming. For this reason, we developed an efficient parallel Simplex optimizer that can run on Solaris and Linux clusters. It can optimize the result of running essentially any program or script that returns a penalty function value.

ALGORITHM

The original simplex method [1, 2] is based on the simple geometric idea that a simplex of (n+1) vertices in input-independent variable-vector space can be transformed into a new one by changing a single vertex, and in particular, by reflecting a selected vertex through the centroid of the remaining n vertices (or equivalently through the centroid of the $(n+1)$ vertices with an appropriate reflection coefficient other than unity). Further transformations involving a change in a single vertex were given geometric names such as "contraction" or "expansion" [3, 4]. Changes affecting n vertices were defined and applied under the titles "shrinkage," "translation," and "rotation" [5]. The above transformations were developed for serial algorithms for which minimizing the number of new vertices is the target since the normal measure of work done is the total number of function evaluations. However, for parallel algorithms many function evaluations (FEs) may be performed at the same time. The term Effective Function Evaluations (EFEs) [5] is used as a measure of complexity or the number of time slots required on a particular parallel machine.

* Work supported by the U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

† shang@aps.anl.gov

In serial algorithms, the transformations of reflection, expansion, and contraction define a new simplex by changing only one selected point, say X_j, so that the operator T_i, $i \neq j$, simply selects and repeats X_i. For parallel transformations, the new simplex is created by applying a parallel operator on every vertex in the original simplex

$$(X0, X1, \ldots, Xn)_{new} = T(X0, X1, \ldots, Xn), \quad (1)$$

where T is a parallel operator that can be expressed as a product of the serial transformations. A new simplex is formed in parallel by employing n parallel processors to compute the n new points (and their corresponding function values). Our method is a revised version of Bassiri's method [6, 7]. Bassiri's method is very efficient in terms of EFEs compared with the Torczon Parallel Multi-Directional Search (PMDS) method [6]. In addition to transforming the vertexes in parallel as done in Bassiri's method, we apply the above three parallel transformations and shrinkage in parallel. Parallel shrinkage transformation shrinks every other n vertexes toward the low point (which has the smallest function value). The simplex will then be replaced by the trial simplex that contains the overall minimum. Therefore, our method is expected to be three times faster than Bassiri's method.

We also added other features:

- Trial simplex from reflection through low point.
- Replacement of the high point.
- Abnormal transformation, which uses larger steps than normal transformation.
- Comparison with previous best to assess convergence. The default convergence test is to compare the difference between high and low points.
- Restart feature, in which the best result is written to a file and can be used for the initial values in next optimization.

Replacing the high point and comparing with the previous best result helped speed convergence in some cases. Abnormal transformation provides faster convergence when the initial guess is far from the minimum.

PERFORMANCE

Bassiri's method reduced the EFEs by about 1000 times compared with the PMDS method. Our parallel simplex method was checked with the testing functions used by Bassiri including Powell's function, TRID-Dixon's function, and Rosenbrock's function. The difference between

high point and low point in the new simplex is used for the convergence test. For Powell's function, the reflecting through centroid method converges slower than reflection through low point method. If the difference between the high point and low point in the new simplex that contains the overall minimum is used for the convergence test, the EFEs are about the same as Bassiri's. However, the accuracy is much greater, for example the $F(x)$ is 1.0×10^{-21} with tolerance of 1.0×10^{-2} and 1.0×10^{-68} with tolerance of 1.0×10^{-11}. If the difference between the current best result and the previous best result is used for the convergence test, the EFEs is reduced by five times with the same accuracy, that is, our method converges about four times faster than Bassiri's method, as shown in Table 1. The x_0 in the tables is the initial or guessed value vector of the independent variables.

Table 1: Powell's Function (n=4); $x_0 = (3, -1, 0, 1)^T$.

Step Tol	Bassiri		This Paper	
	EFEs	F(x)	EFEs	F(x)
1.0e-02	39	4.49e-03	8	1.37e-03
1.0e-03	43	3.28e-04	10	8.72e-05
1.0e-04	57	2.68e-05	11	2.19e-05
1.0e-05	60	1.66e-05	13	1.37e-06
1.0e-06	64	1.37e-06	15	8.56e-08
1.0e-07	69	1.09e-07	16	2.14e-08
1.0e-08	73	1.12e-08	18	1.33e-09
1.0e-09	77	3.27e-09	20	8.36e-11
1.0e-10	84	3.51e-11	21	2.09e-11
1.0e-11	88	2.91e-12	23	1.30e-12

For TRID-Dixon's function, the reflecting-through-centroid method converges as fast as the reflecting-through-low-point method; however, the latter gives better accuracy. The results with reflecting-through-low-point method are shown in Table 2.

Table 2: TRID-Dixon's Function (n=10); $x_0 = (1, \ldots, 1)^T$.

Step Tol	Bassiri		This Paper	
	EFEs	F(x)	EFEs	F(x)
1.0e-02	123	3.208e-02	14	6.706e-07
1.0e-03	135	5.069e-03	17	1.048e-08
1.0e-04	150	6.447e-04	20	1.637e-10
1.0e-05	168	9.558e-05	22	1.023e-11
1.0e-06	192	3.600e-06	25	1.598e-12
1.0e-07	209	4.665e-07	28	2.498e-15
1.0e-08	218	1.905e-07	30	1.561e-16
1.0e-09	253	6.101e-09	33	2.439e-18
1.0e-10	268	1.282e-09	36	3.812e-20
1.0e-11	294	3.672e-11	38	2.382e-21

Table 2 shows that this algorithm works very efficiently with the TRID-Dixon function. In addition to much greater accuracy, it is about ten times faster than Bassiri's method.

For Rosenbrock's function, the two methods (reflecting through centroid or low point) work similarly. The results of the reflecting-through-centroid method are listed in Table 3. It can be seen that this parallel optimizer for Rosenbrock's function is about 30 times faster than Bassiri's method and delivers much greater accuracy. Bassiri did not test the Dixon function, which is more complicated and turns out to converge slower than other functions. Our test shows that the reflecting-through-centroid method gives a better result than the reflecting-through-low-point method. Also, we obtained better accuracy if the high point of shrinkage simplex is replaced by the local minimum if it is smaller than the high point.

Table 3: Rosenrbrock's Function (n=2); $x_0 = (-1.2, 1)^T$.

Step Tol	Bassiri		This Paper	
	EFEs	F(x)	EFEs	F(x)
1.0e-02	17	4.83	10	7.47e-06
1.0e-03	196	3.96e-02	11	3.81e-08
1.0e-04	250	2.31e-02	13	3.81e-08
1.0e-05	312	4.04e-04	15	1.49e-10
1.0e-06	332	2.17e-04	16	1.49e-10
1.0e-07	409	1.86e-05	18	1.14e-10
1.0e-08	500	9.87e-07	20	5.82e-13
1.0e-09	572	1.17e-07	21	5.82e-13
1.0e-10	654	1.99e-11	23	2.27e-15
1.0e-11			25	2.27e-15

It seems that parallel simplex does not work well for the Dixon function, which implies that parallel simplex may not be able to find the minimum for some applications. It should be noted that the EFEs and accuracy depend on the initial values (i.e., guess values), so that fewer EFEs are needed if the initial point is closer to the minimum position. This parallel simplex provides an option of saving the results into a file, allowing them to be used as the starting point for the next optimization, which saves EFEs and helps convergence for difficult functions.

The testing results indicate that this parallel simplex is more efficient than Bassiri's method [7] with greater accuracy. However, in practical applications rather than mathematical functions, this parallel simplex does not always give good results. As a result, another parallel simplex called parallel single simplex was implemented, which employs direct parallelization of the serial algorithm, i.e., only the high point is transformed but four types of transformation are done in parallel; that is, unlike parallel simplex, which creates four new trial simplexs at each iteration, there is only one simplex at each iteration of parallel single simplex. Parallel single simplex converges slower than parallel simplex, so it is better to start with the result from parallel simplex. The actual application will be studied in the following section.

APPLICATIONS

We developed the parallel optimizer to assist with difficult optimization problems, such as those involved in high-brightness storage ring design. While linear optics optimization, including optimization of the emittance and similar quantities, is well handled by `elegant`'s [8] built-in op-

timizer, optimization of nonlinear properties is more challenging and time consuming. Our goal is sufficient dynamic aperture for injection and lifetime. This includes, of course, off-momentum dynamic aperture, which is important in obtaining good Touschek lifetime.

Several approaches have been explored in this work:

- Direct optimization of the dynamic aperture. In this case, dynamic apertures are computed for a series of energies, then analyzed to provide a penalty function value. A possible penalty function is simply the area inside the dynamic aperture summed over all energies.
- Optimization of the number of particles surviving N turns. This is similar to the previous idea, except here we track a beam with initially large phase-space volume, encompassing the target transverse and momentum apertures.
- Minimization of the coefficients of the tune variation due to amplitude and momentum. Here, we use analytical or numerical methods in `elegant` to determine the coefficients of the tune variation.
- Minimization of the tune spreads due to amplitude and momentum spread. In this case, we simply track an ensemble of particles to get the tune spread from transverse amplitude. The tune spread from momentum spread is computed by matrix methods.

We have had little success with the first two methods. A common result with these methods is development of stable, disconnected islands for off-momentum particles. The third method proved problematic because determination of the coefficients of the tune variation is difficult when some of the probe particles are unstable or nearly so.

We found the best success with the last method. Determination of tune spread for a grid of particles is simple and relatively robust. Typically we start by using a grid that covers a relatively small region of the desired aperture. When this has converged, we increase the area covered by the grid and repeat the optimization.

The Advanced Photon Source (APS) storage ring is a 7-GeV synchrotron radiation facility. It presently operates with a lattice giving an effective emittance (beam size times divergence) of 3.1 nm. One of the applications of the parallel optimizer is to help develop lattices with even smaller emittance. These lattices are challenging in that the dynamic and momentum apertures tend to be quite small, resulting in poor lifetime and injection efficiency. To develop candidate lattices, we used `elegant`'s conventional optimization module to find solutions for a variety of working points. In finding these solutions, we also constrained the tune spreads using the last method mentioned above. This involved running `elegant` once for each working point, using up to 40 processors. Following this, several working points were clearly better and were selected for improvement using the parallel optimizer. Figure 1 shows the dynamic aperture prior to use of the parallel optimizer, while Figure 2 shows the improved result after using the parallel optimizer. We see a distinct improvement due to the optimization. Particularly interesting is the absence of any obvious resonance effects in the optimized result.

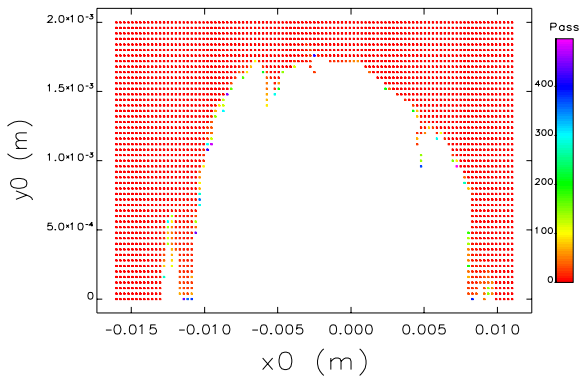

Figure 1: Dynamic aperture before optimization.

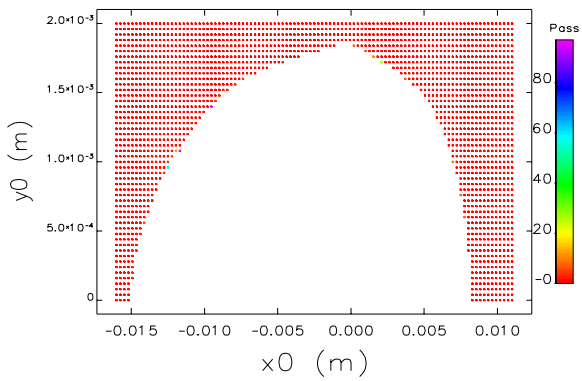

Figure 2: Dynamic aperture after optimization.

REFERENCES

[1] W. Spendley et al., Technometrics, 4:441-6, 1962.

[2] W. H. Press et al., *Numerical Recipes in C*, p 408-416, Cambridge University Press, 1992.

[3] J. A. Nelder and R. Mead, *A Simplex Method for Function Minimization*, Computer Journal, p. 308, 1965.

[4] J. Parkinson and D. Hutchinson, *Numerical Methods for nonlinear Optimization*, Academic Press, 1972.

[5] J. M. Parkinson, Ph.D thesis, Computer Science, University of Leads, May, 1975.

[6] K. Bassiri and D. Hutchinson, *School of Computer Studies Research Report Series*, University of Leads, 1994.

[7] K. Bassiri and D. Hutchinson, *School of Computer Studies Research Report Series*, University of Leads, 1998.

[8] M. Borland, Advanced Photon Source LS-287, September 2000.