Parallel Global Optimization of High-Dimensional Problems

Siegfried Höfinger, Torsten Schindler, and András Aszódi

Novartis Forschungsinstitut,
Brunnerstraße 59, A-1235 Vienna, Austria
{siegfried.hoefinger,torsten.schindler,andras.aszodi}@pharma.novartis.com
http://www.at.novartis.com/healthcare/nfi/default.asp

Abstract. A parallel version of an optimization algorithm for arbitrary functions of arbitrary dimension N has been developed and tested on an IBM-Regatta HPC system equipped with 16 CPUs of Power4 type, each with 1.3 GHz clock frequency. The optimization algorithm follows a simplex-like stochastic search technique aimed at quasi-complete sampling of all the local minima. Parallel functionality is incorporated with the Message Passing Interface - MPI - version 1.2. The program is applied to typical problems of dimension N=60 and N=512 and the results are analyzed with respect to operability and parallel scalability.

1 Introduction

Many practical problems in science and technology are theoretically solvable but practically intractable because they require too long a computation. Computing time becomes a limiting factor particularly in mathematics and physics, where for certain problems the underlying basic principles defining the problem are well-known. However, they often lead to so many equations, or boost the dimensionality of a mathematical space so drastically, that a resulting computational procedure is hindered from successfully deriving a solution within an acceptable time frame. On the other hand supercomputers steadily increase in power and scope. Thus the aforementioned limitations may be overcome by high-performance computing systems and parallel architectures.

In this present study we investigate a general search algorithm that is usually applied to finding an optimum point - a minimum - on a highly complex functional surface. This minimum represents the behaviour of a particular N-dimensional scalar function $f_N(x_1, x_2, x_3, \ldots x_N)$ with respect to variations of the function's arguments $x_1, x_2, x_3, \ldots x_N$. Thus for the 2-dimensional case one could imagine a typical picture of a geological relief. A path in the (x_1, x_2) plane allows us to follow the rise and fall of the function's values on $f_2(x_1, x_2)$ the surface of the landscape. The goal of the search algorithm in such a case is to discover the positions of valleys and related valley base altitudes, finally picking out the optimal point with minimum geographical altitude. The problem with this approach immediately becomes clear even with this simple-minded 2-dimensional

D. Kranzlmüller et al. (Eds.): Euro PVM/MPI 2002, LNCS 2474, pp. 148–155, 2002.

example. It is difficult to determine the global optimal point because the algorithmic procedure should on the one hand try to detect valleys, but on the other hand also try to escape from these local valleys again once they are detected. This is because a local solution need not necessarily be a globally optimal point but the search for the latter is the ultimate goal of the algorithm. Furthermore, the situation becomes worse with increasing dimension N because the number of theoretical local minima grows exponentially also with N. For a real-world problem where N is of the order of 1000 or more the problem therefore becomes CPU bound.

In this paper we present a parallel application that performs the search for various local minima in parallel. In section 2 we outline the basic principles of direct search methods like the simplex optimization and their suitability to parallelization. The procedure is applied to a model test in section 3 and the results are analyzed in section 4.

2 Direct Search Methods – Simplex Optimization

One of the easiest methods to detect minima on a given functional surface is the simplex-method of Spendley, Hext and Himsworth [1] which was subsequently extended by Nelder and Mead [2]. The advantage of this particular method is that there is no need for gradient information, ∇f_N, of the function under consideration. As in a trial-and-error method one starts by setting up a set of N+1 non-identical points and computes corresponding functional values for these and subsequently rearranges all these N+1 positions in a well-defined way. The ultimate objective is to move the set of points closer and closer to the minimum. The idea of the Nelder and Mead algorithm is to successively improve the set of points by always searching for the least optimal point, i.e., the one with the highest functional value. This point is then moved using a geometrical transformation, specifically by a reflection through the centroid of all the remaining points in the simplex.

As pointed out by Torczon [3], the Nelder and Mead extension of the simplex-method exhibits the phenomenon of *restricted convergence*. This basically means that if the requested accuracy of the solution is too great the method fails to find the minimum. As a possible solution to this problem Torczon herself suggested the implementation of Parallel Multi-Directional Search (PMDS) [4]. Within the PMDS framework the way of improving the set of points forming the simplex is different to that of Nelder and Mead. The first major difference is the selection of the best adapted point for use as the centre of reflection, i.e., the point having the smallest functional value and therefore closest to the desired minimum. Secondly, all the remaining N points are then reflected through this selected centre, which is a process that may, in principle, be performed in parallel, hence the parallel character of this new simplex variant. However, according to Bassiri and Hutchinson [5], the phenomenon of *restricted convergence* was not eliminated by employing PMDS. This formed the motivation for the latter authors to devise an alternative method for performing the different moves during stepwise alter-

ation of the set of points which comprise the simplex. Note that another way of discriminating the various methods is to look at the number of points that are involved in the update of the simplex. Whereas in the early methods only one point was selected and became subject to geometric operations, in the second set of methods starting with PMDS, all points but one (the one with smallest functional value) were chosen and shifted around according to the employed set of geometric rules.

For the present purposes we will use the set of simplex-update-rules specific to the PMDS approach, which all are exhaustively defined from

$$p_i^*(x_1, x_2, x_3, \ldots x_N) = (1+t)p_l(x_1, x_2, x_3, \ldots x_N) - tp_i(x_1, x_2, x_3, \ldots x_N) \quad (1)$$

update-rule	t-setting	application pre-condition
reflection	$t = 1$	not converged, standard operation
expansion	$t > 1$	one point of the reflected set gives rise to an even smaller f_N than current optimum
contraction	$0 < t < 1$	reflection fails to produce any point with f_N smaller than the current optimum

where p_i refers to one particular point of the N+1 points forming the simplex; p_l represents the only stationary point having the lowest functional value, and t is a parameter following the tabular description given above.

2.1 Enhancements towards Global Search

The procedure outlined in section 2 does not find a globally optimal point. It merely enables the local minimum lying next to the initial simplex to be discovered. Therefore additional features need to be established to enable the search for an overall optimal point. We suggest the following implementation:

- Use a regular simplex and standard simplex moves. Use the simplex moves only for translocation on the functional surface.
- At all the N+1 points defining a simplex employ independent local minimizations. The minimizations of these N+1 points are independent tasks and may be done in parallel.
- The resulting N+1 local minima are stored.
- Screen the list of detected local minima for unique entries as well as for new-coming entries to monitor the progress of the global search.
- Standard simplex-termination would occur when all the N+1 points define roughly the same position (a minimum - local or global). When this happens in the global search, all the N+1 points are re-initialized and the simplex starts afresh.
- The global search terminates if no new local minima are discovered even after several simplex re-initialization steps.
- Pick the point with smallest f_N from all the recorded unique local minima as the likely global minimum.

2.2 Parallel Aspects

From the profiling of a typical example with N=60 we could estimate the relative CPU-load for the isolated local minimization step to be of the order of 99.98 %. Thus within this context it was just a natural consequence to develop a parallelized version of the algorithm, described in section 2.1, that could perform the local minimization step at all the N+1 positions of the simplex in parallel. A straightforward solution was to split the N+1 local minimization steps into fractions, that could then be distributed over parallel operating CPUs.

The parallel algorithm is characterized as having very short communication intervals on the one hand and long lasting, independently operating parallel intervals on the other hand. The communication overhead is mainly due to sending the N+1 simplex coordinates to all participating parallel nodes and receiving the N coordinates of corresponding minima back from the parallel nodes. Compared to the time the individual nodes have to spend on computing their own sets of local minima this communication time is to a great extent negligible. The application inherent characteristics form the basis for an efficient parallel implementation using MPI. A master-node scheme was therefore adopted, where the master-process was responsible for all simplex-related organizational work, and the node-processes were due to local minimization jobs. Master "send" operations included sending of the N+1 points forming the current simplex to each of the nodes, while the nodes would just work on a predefined subset of these N+1 items and send back the computed corresponding local minima. Therefore there was no need to invoke higher level collective communication operations. We simply could get by with broadcasting buffered messages of length N+1 and receiving buffered messages of identical length via specific node-master sends.

For incorporation of parallel features the Message Passing Interface [6] seemed to be the most appropriate. This is because of the small additional programming efforts we had to undertake in obtaining a parallel version from modifying the already existing (and functional working) serial code. Furthermore, the target multiprocessor machine we had in mind for evaluation purposes was already equipped with a proprietary version of MPI-2 including FORTRAN 90 bindings (IBM parallel environment for AIX, version 3, release 2). Another important aspect when choosing MPI as the communication protocol was portability, since the described application is still being further developed and the number of available parallel architectures within our research institute is steadily growing. Regardless what parallel environment will finally become our target, the 16-CPU SMP IBM-machine appeared to better perform with MPI than PVM for comparable parallel tasks.

3 Results

A MPI-parallel version of the global simplex-like search algorithm described in section 2.1 was derived from the serial variant written in FORTRAN 90. A rather

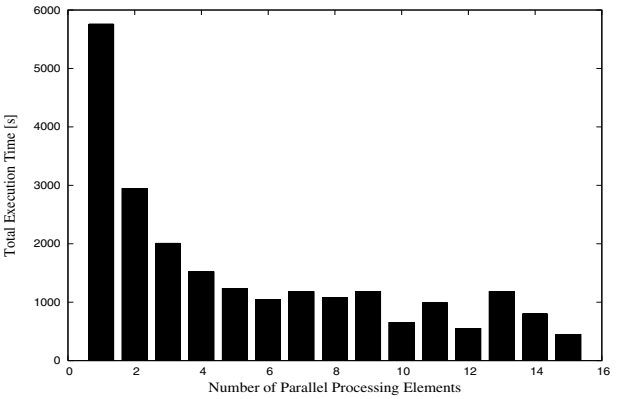

Fig. 1. Parallel performance of a simplex-like global search algorithm for a N=60 dimensional test function. This example is explored over 100 successive simplex cycles. The parallel architecture was an IBM-Regatta SMP system with 16 CPUs of type Power4 (1.3 GHz)

complicated test function [7],

$$f_N(\boldsymbol{x}) = \sum_{i=1}^{i=N/2} a\, x_{2i-1}^2 + b\, x_{2i}^2 + c\, cos(\alpha\, x_{2i-1}) + d\, cos(\gamma\, x_{2i}) - c - d \qquad (2)$$

with $a = 1.0, b = 50.0, c = 3.0, d = 4.0, \alpha = 3\pi, \gamma = 4\pi$ has been used for all the test runs analyzed below.

At first the dimensionality was chosen to be N=60 and then a 100 cycle simplex-like global search procedure was invoked and the execution time measured on a single Power4 CPU (1.3 GHz) of a SMP 16-node IBM-Regatta server. The identical test calculation was repeated with successively increasing numbers of parallel processing elements (finally reaching 15 nodes). The master process itself accounted for a separate single CPU. A graphical representation of the results is given in Fig. 1. The resulting speed-up using 15 processors was 12.7-fold.

In addition a second run was set up where the dimensionality of the problem was increased to N=512 and the code executed on all available 16 CPUs of the parallel machine. Here the aim was to measure the number of newly-identified local minima in the course of the global search. Fig. 2 shows the percentage of the N+1 simplex points that led to newly identified local minima throughout the first 10 iteration steps.

4 Discussion

A typical simplex reflection move is demonstrated on a 2 dimensional example case in Fig. 3. As explained in section 2 there are 2+1 points involved and if

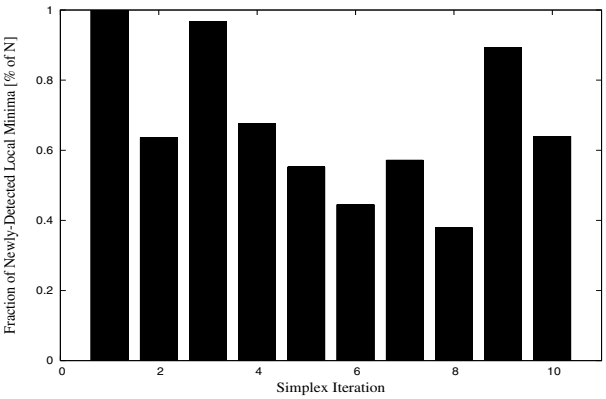

Fig. 2. Rate of de-novo identification of local minima in the course of iterative simplex moves with following local minimizations for a N=512 dimensional test case

we follow the PMDS-strategy according to Torczon we move all points but one. The point that remains unchanged is the one having smallest functional value f_2 from the first, which is the boundary point of the two triangles in Fig. 3. Next we apply local minimization steps at all the various simplex positions. This leads us to the closest local minimum, which is illustrated by the zigzag-pathways in the contour plot of Fig. 3. This must be considered as the major difference to other existing global search methods. First of all we do not feed back the information of positions of local minima to the subsequent simplex-moves. The detected local minima are just stored and screened for new values. This gives us an idea of the progress of the global optimization process. Secondly, we perform the local minimization jobs, which all are independent tasks, in parallel. In particular we apply a minimization procedure similar to the one outlined in [8]. The important thing to realize with this method is that it is a gradient-free technique and that it employs the well-known Powell method of local minimization.

In the beginning of the procedure and when all the local minima coincide we initialize the simplex. For this purpose we define one point randomly. Based on this random point a new regular simplex is formed with randomly chosen edge-length.

Parallel scaling exhibits a non-optimal behaviour with large numbers of CPUs. For example as seen from Fig. 1 increasing the number of CPUs from 10 to 11 actually results in slow-down instead of speed-up. However this is due to the fact that we have chosen a 60-dimensional case. There is no optimal way of splitting a group of 60 elements into equal-sized portions of 11 fractions. So for example in such a case we have to employ 8 CPUs with 5 minimizations, and always 1 CPU with 6,7 and 8 minimizations, which is certainly shifting the bottleneck to the process dealing with 8 minimizations. Therefore the best parallel performance is observed in situations where 60 may be divided through the num-

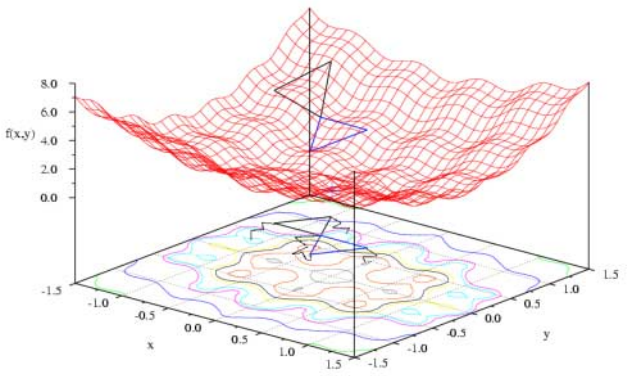

Fig. 3. Simplex reflection move illustrated with a 2 dimensional example. From the 2+1 simplex points the one with smallest f_2 is held constant, while the remainder are reflected through this very point. In the contour plot we also indicate the following local minimization steps

ber of CPUs (see Fig. 1). Furthermore according to Amdahl's Law, with a serial fraction of 0.0002, we should see a much higher degree in parallel speed-up. For example using 12 CPUs we should theoretically obtain a factor of 11.97x but see 10.32x instead. The reason for this is that different local minimization steps may last differently long depending on how far away the point we start with is actually located from the next local minimum.

There is a characteristic drop in the number of newly detected local minima every second step of the simplex moves (see Fig. 2). This is the effect of closer related simplex-point-positions for expansion and contraction moves which both follow the standard reflection move. So what we see in Fig. 2 is the effect of alternating reflection and contraction/expansion moves from which only the first type always leads to a completely different set of positions (also compare to Fig. 3). In addition we report that the N=512 example needed more than 3 full 16xCPU-days to complete the initial 10 iterations which is far from exhaustive exploration of the entire functional surface.

5 Conclusion

A new global search algorithm has been presented here in which "simplex-like" moves are followed by local minimization steps. The advantage with this new method is that a robust and well-established technique (simplex) is combined with additional local minimization-refinements that may be efficiently performed in parallel. While the current implementation works on a mathematically well defined analytical test-function, the concept may be easily generalized to ex-

plore any arbitrarily complex potential surface. Such minimization problems are often encountered in rational drug design, in particular in the analysis of conformational preferences of small molecules. Another possible application area is the computational study of ligand-receptor interactions (docking) that has the potential of speeding up the lead optimization stage of the drug discovery process. Work is currently underway in our laboratory to explore these application opportunities where the speed-up provided by parallelization offers a significant methodological advantage.

Acknowledgment

The authors would like to thank Dr. Adrienne James for helpful discussions and Dr. Pascal Afflard and his colleagues at Novartis Basel for granting access to their computing facilities.

References

[1] Spendley, W., Hext, G. B., Himsworth, F. R.: Sequential application of simplex design in optimisation and evolutionising operation. Technometrics **4** (1962) 441 149

[2] Nelder, J. A., Mead, R.: A simplex method for function minimization. Comp. J. (1963) 308 149

[3] Torczon, J. V.: On the convergence of the multi-directional search algorithm. SIAM J. Optimization **1** (1991) 123–145 149

[4] Torczon, J. V.: Multi-Directional Search: A Search Algorithm for Parallel Machines. PhD thesis, Rice University, Houston, Texas (May 1989) 149

[5] Bassiri, K., Hutchinson, D.: New Parallel Variants on Parallel Multi-Dimensional Search for Unconstraint Optimisation. research report 94.28, Div. of Computer Science, University of Leeds, UK, (1994) 149

[6] Gropp, W., Lusk, E., Skjellum, A.: Using MPI: Portable Parallel Programming with the Message Passing Interface. 2nd edition. MIT Press, Cambridge, MA, (1999) 151

[7] Dennis, S., Vajda, S.: Semiglobal Simplex Optimization and its Application to Determining the Preferred Solvation Sites of Proteins. J. Comp. Chem. **23**, 2 (2002) 319–334 152

[8] Brent, R. P.: Algorithms for Minimization Without Derivatives, Chapter 7. Dover Publications, Mineola, New York, 0-486-41998-3, (2002) 153