PARALLEL DIRECT SEARCH METHODS FOR SIMULATION-BASED OPTIMIZATION

Frank Thilo, Carsten Müller, and Manfred Grauer

Information Systems Institute, University of Siegen, Hölderlinstr. 3, D-57068 Siegen, Germany
thilo@fb5.uni-siegen.de

Keywords: Optimization, distributed computing, computational engineering

In many engineering disciplines, the use of realistic computing models has become an invaluable tool in the design process. Complex simulation codes are able to approximate the behavior of intricate systems or the properties of components without the need for costly physical experimentation. Optimization algorithms can be used to automatically find the set of parameters within the design space for which the simulation promises the most desirable characteristics. However, there are several challenges that must be met to successfully apply this technique to real-world problems.

First, the objective function to be minimized (or maximized) is given only implicitly and a time-consuming simulation is necessary to calculate its value for a given set of parameters. Thus, almost no assumptions can be made about this function, which will often be highly non-linear and multimodal. Furthermore, there are usually a number of constraints that divide the design space into feasible and infeasible regions of unknown geometry. Derivate information is typically not provided by the simulation codes, and due to numerical noise, the objective function might also be non-smooth. These characteristics make it very hard to apply some classical methods such as gradient-based approaches. A class of optimization algorithms that can be used are so-called direct search methods (1).

Each solution candidate generated by the optimization algorithm must be evaluated, hence necessitating the execution of a time-consuming simulation. Despite the increase of computing power, typical runtimes of a single simulation still span from a few minutes to many hours. This is caused by the demand for larger models, greater accuracy, and the adoption of coupled multiscale and multiphysics simulation codes (2). During the course of the optimization, hundreds or thousands of evaluations are necessary, resulting in very long runtimes. Two common approaches to decrease the time needed are the use of surrogate functions (3) and parallelization. Since the computation time spent within the optimization algorithm itself is several orders of magnitude lower than the time needed for a single simulation, it is useless to introduce parallelism to the internal operations of the algorithm. Instead, the goal is to design the algorithm in a way that allows it to utilize the results of many simulations that can be run simultaneously and independently of each other.

In this paper, eight such parallel direct search methods for simulation-based optimization problems are examined. Most of them are based on well-known sequential search methods and were modified to exploit parallel computing resources:

- Distributed Polytope Search (4) applies geometric operations to a set of points in the search space to generate new solutions. Infeasible solutions are repaired by moving them towards the center of gravity.
- Parallel Scatter Search (5) is a parallel implementation of the well-known scatter search meta-heuristic.
- Asynchronous Parallel Pattern Search (6) is a pattern search method with the unique property of asynchronous parallel operation.
- Simulated Annealing (7) is a parallel variant of the classical SA method which uses a stochastic, temperature-dependant acceptance function to avoid getting stuck in local minima.

- Great Deluge Algorithm (8) is similar to SA but uses a different acceptance function based on a flood level.
- Particle Swarm Optimization (9) simulates a swarm of particles moving through the search space and attracting each other.
- Genetic Algorithm (10), an incarnation of the bioinspired search method for real-valued decision variables.
- Evolution Strategies (11) are closely related to GA but add the concept of so-called strategy parameters, which enable self-adaptation of the search strategy.

Some of the algorithms generate a sufficient number of new solution candidates per iteration in their original, sequential form and are thus easily extended to make use of parallel computation. Others, like Distributed Polytope Search, differ significantly from the algorithm they have been derived from. Furthermore, some of the algorithms can also operate in asynchronous mode, meaning that further operation is not suspended until all pending simulations have finished. This is especially important in a heterogenous computing environment where the runtimes of the simulations vary significantly.

Advances in the area of service-oriented architectures (12) and grid computing (13) make it easier to use resources beyond geographical and organizational boundaries, theoretically enabling even small companies to utilize many thousands of CPUs on demand. However, the problem of licensing still limits the use of commercial simulation software in these environments. Thus, while most of the observations will also apply to large scale computing, the paper focuses on degrees of parallelism of up to a few hundred CPUs – typical of compute clusters or enterprise grids.

The algorithms were used to solve several real-world problems in different engineering disciplines. This includes sheet metal forming and optimization of metal alloy casting processes in the automotive industry, and facility optimization in groundwater management. Results are presented for both a test function as well as two problems from industrial practice. The computational experiments were performed on a 300 CPU Linux Opteron cluster. While the test function allows for an extensive examination of the algorithms' performance over a wide range of utilized CPUs and different problem dimensions, the simulation-based optimization problems indicate the relevance of the contribution to non-academic tasks.

REFERENCES

[1] Kolda, T.G., Lewis, M.R., Torczon, V. (2003): *Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods*, SIAM Review, Volume 45, Number 3, pp. 385-482.

[2] Cramer, E., Dennis, J., Frank, P., Lewis, R., Shubin, G. (1993): *Problem formulation for multidisciplinary optimization.*, SIAM Journal on Optimization 4, 754-776.

[3] Booker, A.J., Dennis, J.E., Frank, P.D., Serafini, T.B., Torczon, V., Trosset, M.W. (1999): *A rigorous framework for optimization of expensive functions by surrogates*, Structural Optimization, 17, pp. 1–13.

[4] Thilo, F., Junghans, U., Grauer, M., Kaden, S., and Hillebrandt, J. (2005): *Reducing Groundwater Management Costs by Parallel Simulation-based optimization* in: Proceedings Computing and Control in the Water Industry, CCWI 2005, Exeter, pp. 135-140.

[5] Marti, R., Laguna, M., and Glover, F. (2006): *Principles of Scatter Search*, European Journal of Operational Research 169 (2), 359-372.

[6] Hough, P., Kolda, T. G., and Torczon, V. (2001): *Asynchronous Parallel Pattern Search for Nonlinear Optimization*, SIAM Journal of Scientific Computing 23(1), 134-156.

[7] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. (1983): *Optimization by Simulated Annealing*, Science 220(4598), 671-680.

[8] Dueck, G. (1993): *New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel*, Journal of Computational Physics 104(1), 86-92.

[9] Kennedy, J., Eberhart, R., and Shi, Y. (2001): *Swarm Intelligence*, Morgan Kaufmann Publishers.

[10] Ahn, C.W. (2006): *Advances in Evolutionary Algorithms: Theory, Design and Practice*, Springer.

[11] Schwefel, H.-P., and Beyer, H.-G. (2002): *Evolution strategies – a comprehensive introduction*, Natural Computing 1(1), 3-52.

[12] Singh, M. P., and Huhns, M. N. (2005): *Service-Oriented Computing – Semantics, Processes, Agents*, John Wiley & Sons.

[13] Foster, I., and Kesselman, C. (Eds.) (2004): *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann.