PARALLEL DECOMPOSITION METHODS
FOR BIOMECHANICAL OPTIMIZATION

Byung Il Koh[1], Jeffrey A. Reinbolt[2,3], Benjamin J. Fregly[2,3], and Alan D. George[1]
[1]Department of Electrical & Computer Engineering, University of Florida, Gainesville, FL
[2]Department of Mechanical & Aerospace Engineering, University of Florida, Gainesville, FL
[3]Department of Biomedical Engineering, University of Florida, Gainesville, FL
E-mail: fregly@ufl.edu, Web: www.mae.ufl.edu/~fregly

INTRODUCTION

Optimization algorithms are frequently used to solve system identification or movement prediction problems utilizing complex three-dimensional (3D) musculoskeletal models (Pandy, 2001). To date, gradient-based, simplex, simulated annealing, genetic, and particle swarm algorithms have been used for such applications (Van Soest and Casius, 2003; Neptune, 1999; Pandy, 2001; Van den Bogert *et al*., 1994). For large-scale problems, these algorithms have a high computational cost since they iteratively evaluate a 3D simulation to determine the cost function and constraints. Moreover, as the complexity of 3D movement models increases, the computational expense of repeated simulations increases dramatically. Although the performance of single-processor computers has increased in recent years, computation time can still be a limiting factor.

To circumvent this limitation, the computational load of 3D biomechanical optimization problems can be decomposed and distributed to multiple processors in a parallel computer system. Two general approaches are possible to develop such parallel algorithms. The first parallelizes the optimizer, either finite difference calculations for gradient-based optimizers (Pandy, 2001) or samples within the search space for global algorithms (Van Soest and Casius, 2003). The second approach parallelizes the simulation (or analysis function) itself so that the workload is distributed to the processors with finer granularity. Together, the optimization algorithm and simulation can be viewed as the upper and lower levels, respectively, of a two-level process, where either level can be parallelized. To date, no biomechanical studies have sought to parallelize the lower level or both levels simultaneously.

This paper describes three parallel algorithms for 3D biomechanical optimization and analyzes performance based on the level of parallelization. The concepts are demonstrated using a biomechanical system identification problem for a 3D kinematic ankle joint model. Joint positions and orientations in the body segments that result in the best match to experimental movement data are determined using an unconstrained gradient-based optimization algorithm.

METHODS

BFGS Optimization Algorithm: An unconstrained gradient-based optimization algorithm (Broydon-Fletcher-Goldfarb-Shanno, or BFGS) available in commercial software (VisualDOC, VanderPlaats R&D, Colorado Springs, CO) was used for the upper-level optimization in this study. The BFGS algorithm calculates a gradient-based search direction followed by a line

search along that direction through function evaluations. The function evaluations for gradient calculations are inherently parallelizable, since no data dependencies are involved.

Description of Analysis Function: A 3D biomechanical system identification problem was used to evaluate three parallel decomposition methods. The problem involves determination of patient-specific parameter values that permit a 3D kinematic ankle joint model to reproduce experimental movement data as closely as possible (Van den Bogert *et al*., 1994). Twelve parameters (treated as design variables) specify the fixed positions and orientations of joint axes in adjacent body segments (i.e., shank, talus, and foot) within the 8 degree-of-freedom (DOF) kinematic ankle model. The system identification problem is solved via a two-level optimization approach. Given the current guess for the 12 parameters, the lower-level simulation (or sub-optimization) adjusts DOFs in the model so as to minimize the 3D coordinate errors between modeled and experimental surface markers. The upper-level optimization adjusts the 12 parameters defining the joint structure so as to minimize the cost function calculated by the lower-level optimization over all time frames.

In mathematical form, the above system identification optimization problem can be stated as follows:

$$f(x) = \min_{p} \sum_{t=1}^{nf} e(p,t) \tag{1}$$

with

$$e(p,t) = \min_{q} \sum_{i=1}^{nm} \sum_{j=1}^{3} \left(\hat{a}_{ij}(t) - a_{ij}(p,q) \right)^2 \tag{2}$$

where Eq.(1), the cost function for the upper-level optimization, is a function of patient-specific model parameters p and is evaluated for each of nf recorded time frames t. Eq.(2) represents the lower-level cost function and uses nonlinear least squares to adjust the model's DOFs q to minimize errors between experimentally measured marker coordinates $\hat{a}$ and model-predicted marker coordinates a, where nm is the number of markers whose three coordinates are used to calculate errors. The sum of the squares of 3D marker coordinate errors obtained from lower-level optimization of every time frame produces the upper-level cost function.

Decomposition method 1 (Gradient calculation decomposition): A general approach for decomposing a gradient-based optimization is to evenly distribute the gradient calculation function evaluations to different processors. Since there are no data dependencies involved in these function evaluations, this method is commonly used for parallel gradient-based optimization algorithms (Pandy, 2001). The same decomposition method was implemented in the present study using VisualDOC API functions with the Message Passing Interface (MPI).

Decomposition method 2 (Analysis function decomposition): Analysis function decomposition is based on knowledge of the independent nature of the sub-optimizations. The analysis function performs a separate sub-optimization for each time frame of data. To create a parallel version this functionality, the seed for each sub-optimization was re-initialized to zero after every five time frames. This permitted parallelization in blocks of five time frames without affecting computational speed or numerical accuracy.

Decomposition method 3 (Combined decomposition): Combined decomposition is a hybrid approach that combines the benefits of gradient calculation and analysis function decomposition. During gradient calculations, available slave processors are divided into groups for gradient calculation decomposition, where processors in each group execute assigned lower-level optimizations exactly as in analysis function decomposition. However, during line searches, combined decomposition is identical to analysis function decomposition, since line searches could not be parallelized.

All three decomposition methods were implemented using a master-slave paradigm on a Linux-based PC cluster (1.33GHz Athlons each with 256MB memory on a 100Mbps switched Fast Ethernet network) in the University of Florida HCS Research Laboratory.

RESULTS AND DISCUSSION

Parallel decomposition at one or both levels resulted in significant performance improvements compared to sequential optimization of the ankle joint system identification problem. Total execution time decreased as the number of processors increased for each of the parallel decomposition methods (Fig. 1a). For comparable numbers of processors, analysis function decomposition was the fastest and gradient calculation decomposition the slowest. Consistent with these observations, parallel efficiency was better than 90% for analysis function decomposition with 5 and 10 processors while for gradient calculation decomposition, it dropped from 59% to 28% as the number of processors increased from 4 to 12 (Fig. 1b).
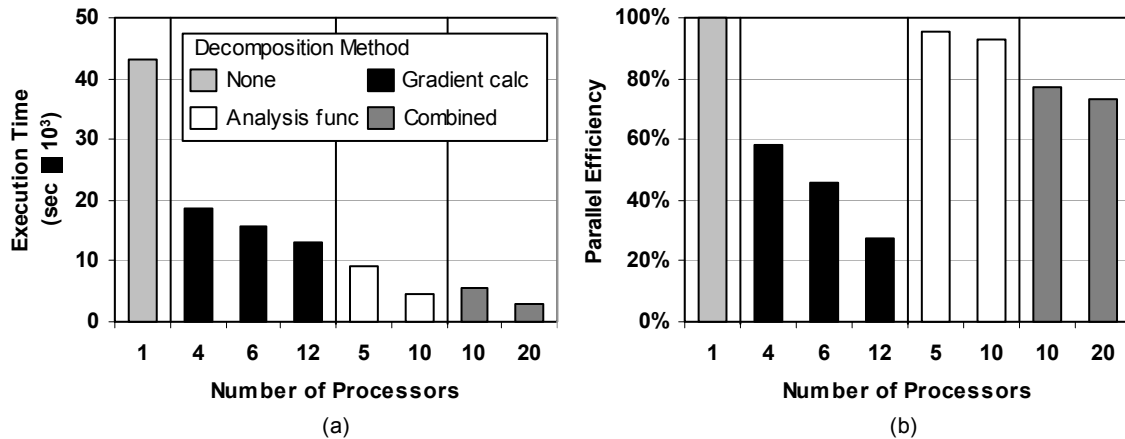

(a)   (b)

Figure 1: Performance metrics for the three parallel decomposition methods as a function of the number of processors. (a) Total execution time, (b) Parallel efficiency – sequential execution time divided by the product of parallel execution time and the number of processors.

Though gradient calculation decomposition is the most common, analysis function decomposition produced the most computationally efficient parallel algorithm. Somewhat surprisingly, combined decomposition performed slightly worse than analysis function decomposition for a comparable number of processors, indicating that the additional work to parallelize the optimizer was not helpful for this problem. Since parallelization effort is typically spent on the optimizer rather than the analysis function, these results suggest that future parallel biomechanical optimization efforts should consider focusing on the analysis function instead.

3

The main reason gradient calculation decomposition performed the worst was our inability to parallelize line searches, resulting in load imbalance. However, for analysis functions with a larger number of design variables, the effects of line search inefficiencies may be diminished relative to gradient calculation computation time. The other limitation of this method is that the maximum number of processors is bounded by the number of design variables. This limitation would also be eliminated for large-scale problems with hundreds of design variables. Analysis function decomposition performed the best primarily because of its finer workload granularity as the number of processors increased. This decomposition method is not limited by the characteristics of the optimization algorithm or the number of design variables and is able to parallelize the largest percentage of the total computations. Combined decomposition exhibited slightly worse performance than analysis function decomposition due to the presence of idle processors during line searches.

SUMMARY

This study has presented three parallel algorithms for 3D biomechanical optimization. The algorithms were applied to a biomechanical optimization problem involving system identification of a 3D kinematic ankle joint model. Parallelization of optimizer gradient calculations resulted in the worst performance for a fixed number of processors while parallelization of the analysis function resulted in the best performance. Thus, parallelization of the optimizer may not always be the most computationally efficient choice. Significant performance gains for gradient-based optimizers would be obtained by parallelizing line searches as well. An interesting direction for future research would be to apply analysis function decomposition to other computationally intensive 3D biomechanical optimization problems using gradient and non-gradient parallel optimization algorithms. To reduce idle processor time, dynamic load balancing or asynchronous parallel optimization algorithms could be explored to improve parallel performance in heterogeneous environments with different processor speeds.

REFERENCES

Neptune, R.R. (1999). Optimization algorithm performance in determining optimal controls in human movement analyses. *J. Biomech. Eng.*, **121**(2), 249-252.

Pandy, M.G. (2001). Computer modeling and simulation of human movement. *Annu. Rev. Biomed. Eng.,* **3**, 245-273.

Van den Bogert, A.J. et al. (1994). In vivo determination of the anatomical axes of the ankle joint complex: an optimization approach. *J. Biomechanics*, **12**, 1477-1488.

Van Soest, A.J.K. & Casius, L.J.R. (2003). The merits of a parallel genetic algorithm in solving hard optimization problems. *J. Biomech. Eng.*, **125**(1), 141-146.

ACKNOWLEDGMENTS