Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

REDUCING COMPUTATION TIME OF STOCHASTIC SIMULATION-BASED OPTIMIZATION USING PARALLEL COMPUTING ON A SINGLE MUTLI-CORE SYSTEM

Mohammed Mawlana

Mississippi State University
823 Collegeview St,
Starkville, MS 39762, USA

Amin Hammad

Concordia University
1455 De Maisonneuve Blvd. W.
Montreal, QC H3G 1M8, CANADA

ABSTRACT

This paper presents a framework for implementing a simulation-based optimization model in a parallel computing environment on a single multi-core processor. The behavior of the model with multicore architecture is studied. In addition, the impact of multithreading on the performance of simulation-based optimization is examined. The framework is implemented using the master/slave paradigm. A case study is used to demonstrate the benefits of the proposed framework.

1 INTRODUCTION

Multi-objective optimization using metaheuristic algorithms is one of the active research areas in the field of construction engineering and management. Among the different metheuristic algorithms, Genetic Algorithms (GAs) are the most commonly used (Liao et al. 2011). However, it has been reported repeatedly that the use of metahueristic optimization methods, such as GAs, for optimizing large-scale construction projects requires long computational time (Feng et al. 2000; Li and Love 1997; Li et al. 1999; Hegazy and Petzold 2003; Kandil and El-Rayes 2005).

Combinations of simulation with metaheuristic optimization methods, known as simulation-based optimization, have been used to optimize construction operations (Glover et al., 1996; Alberto et al. 2002; Hegazy and Kassab 2003; Cao et al. 2004; Marzouk and Moselhi 2004; Zhang et al. 2006; Marzouk et al. 2009; Yang et al., 2012; Mawlana and Hammad 2013; Salimi et al., 2014; Alanjari et al. 2015; Salimi et al., 2015). The use of discrete event simulation to evaluate the objective functions adds another dimension to the complexity of the optimization problem and as a result, it increases the required computational time. If stochastic simulation is used, then a number of replications must be performed for each solution in order to obtain a sound estimation of the objective functions. The computation time, for a simple simulation mode, will increase almost in a linearly manner as the number of replications performed is increased as shown in Figure 1. Although the computational time is very small per solution, this time will increase rapidly as more candidate solutions are evaluated.

Several researchers, in the field of construction engineering and management, have proposed the use of parallel computing in order to reduce the required computation time to solve the optimization problem (Kandil and El-Rayes 2006; Kandil et al. 2010; Yang et al. 2012; Salimi et al. 2014; Salimi et al. 2015). Couple of works can be found related to the use of parallel computing for simulation-based optimization model. Yang et al. (2012) proposed integrating Particle Swarm optimization algorithm with Monte-Carlo simulation to plan bridge maintenance. The model was implemented in a parallel computing framework on a cluster of computers. Salimi (2014) proposed using Non-dominated Sorting Genetic Algorithm with discrete event simulation to optimize bridge construction operations. The model was implemented in a parallel computing framework on a server and a cluster of computers.

While the previous studies have provided efficient solutions for reducing the computation time of simulation-based optimization models, they require the use of a cluster of computers or multiprocessor server machine. However, these hardware systems come with a high price tag and they may not available

for construction planners. There has been little or no reported research exploring the promising potential of parallel computing on a single multi-core processor in optimizing construction planning problems.

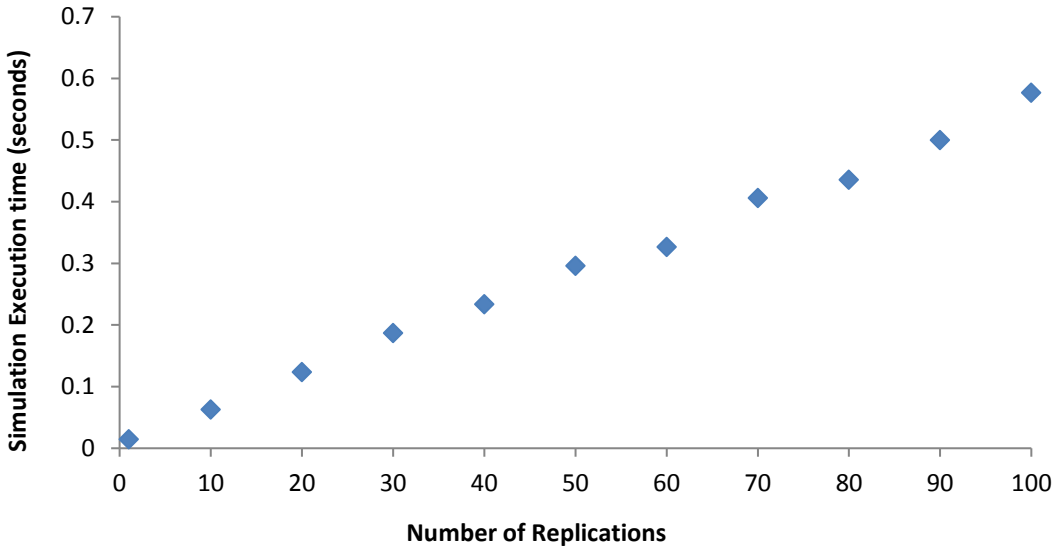

Figure 1: Simulation execution time versus the number of replications.

The main objective of this paper is to reduce the computation time required for performing a stochastic simulation-based multi-objective optimization. This objective is realized by developing a parallel computing framework on a single multi-core processor using existing optimization and simulation tools. The proposed framework uses the master/slave paradigm (Munawar et al. 2008) and it attempts to answer the following research questions: (1) How parallel computing can be implemented using existing optimization and simulation tools? (2) How the simultaneous multithreading technology impacts the computation time? (3) What is the impact of the size of the Random Access Memory (RAM) on the computation time? (4) What is the time saving that can be achieved using this framework? and (5) What is the optimum number of cores to be used?

Reducing the computation time of optimizing construction operations is highly desirable especially when re-planning and re-allocation of resources might be necessary when a deviation from the initial plan is detected. Furthermore, in the planning phase, the time saving achieved by the proposed framework can be used to increase the confidence in the optimality of the solutions by increasing the number of evaluated candidate solutions, which results in covering a larger portion of the search space of the optimization problem.

2 PROPOSED FRAMEWORK

This paper presents a framework for implementing the simulation-based optimization model in a parallel computing environment on a single multi-core processor. Previous research have used parallel computing in simulation and optimization. However, the behavior of running discrete event simulation-based optimization on a single system with multicore processor (i.e. laptops and desktops) has not been studied. Therefore, it is of interest to examine the impact of multi-core and multithreading on the performance of simulation-based optimization.

The proposed parallel simulation-based optimization framework (Figure 2) can be used by decision makers to improve the efficiency of the current practice of decision-making in construction projects. The process starts by generating an initial population of size E by the master core (core 1) as shown in line 2 in Figure 3. Then, the master core subdivides the population among the number of slave cores assigned to the optimization process as shown in lines 4 to 18 in Figure 3. Core 1 acts as a master during the generation

and the fitness evaluation of the population. During the evaluation process, the master core could become a slave core if the optimization algorithm is set to perform the execution on: (1) one core, or (2) a number of cores that is larger than the total number of available cores. For example, in a computer with four cores, the master core will become a slave core if the optimization is set to perform the execution on 5 cores or higher. Each slave core, thereafter, evaluates the performance measure indices of the assigned candidate solutions using discrete event simulation. Once all the slaves evaluate all the generated candidate solutions, the master core evaluates the fitness of the population. If the termination criterion is met, the optimization process ends and the Pareto solutions are presented. Otherwise, the master core will sort the current population and generate a new population by performing cut-splice and mutation. The new population will go through the same steps again.

3 PERFORMANCE METRICS

Three performance metrics are used to measure the performance of the proposed parallel computing framework. The elapsed time metric is used the measure the time required to solve the optimization problem. The required time is measured from the beginning of the simulation-based optimization to its termination. Using this metric, the time saving that is achieved using the proposed framework can be measured. In addition, the impact of the simultaneous multithreading technology can be examined.

The parallel speedup metric measures the amount of time saved by executing the optimization problem in parallel (Cantú-Paz 2000). In addition, it is used to study the impact of the number of cores on the framework performance. It is calculated by dividing the required time to solve the optimization problem using sequential computing (traditional method) by the required time to solve the optimization problem using parallel computing as shown in (1).

$$SU = \frac{T_{seq}}{T_{par}} \tag{1}$$

where SU is the achieved speedup; T_{seq} is the required time using sequential computing; and T_{par} is the required time using parallel computing.

The third metric is the parallel efficiency of executing the optimization problem in parallel, which measures the portion of the processor power used to solve the optimization problem. The other portion of the processor power is typically consumed by synchronization and communication overhead (Fox et al. 1988). This metric is calculated by dividing the achieved speedup by the number of cores used as shown in (2).

$$EF = \frac{SU}{NC} \tag{2}$$

where EF is the efficiency of executing the program in parallel; and NC is the number of cores used.

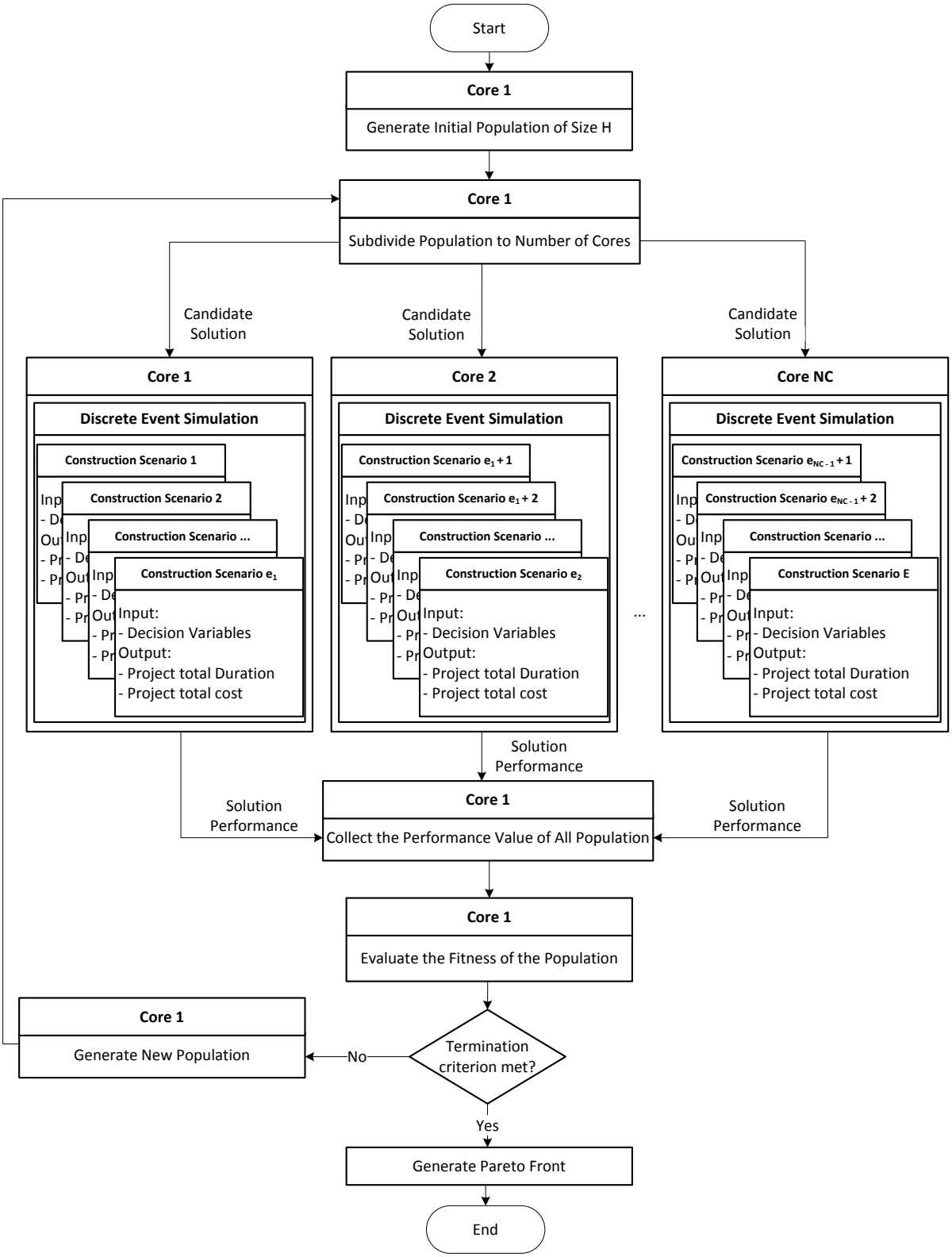

Figure 2: Parallel simulation-based optimization flowchart.

```
1  // Repeat for all generations
2  FOR l = 1 TO L
3  // Repeat for all solutions in the population
4    FOR e = 1 TO E
5      IF rank = 1
6        Evaluate e using Core 1
7        RETURN D_e and C_e
8      ELSE IF rank = 2
9        Evaluate e using Core 2
10       RETURN D_e and C_e
11     ELSE IF rank = 3
12       Evaluate e using Core 3
13       RETURN D_e and C_e
14     ELSE IF rank = 4
15       Evaluate e using Core 4
16       RETURN D_e and C_e
17     END IF
18   END FOR
19 END FOR
```

Figure 3: Algorithm for distributing the population among the cores.

4 FRAMEWORK IMPLEMENTATION

The simulation model of the construction operation is implemented in STROBOSCOPE (Martínez 1996). On the other hand, Darwin optimization framework (Wu et al. 2012), which utilizes an fmGA and equipped with parallel computing capabilities, is used to solve the optimization problem. The implementation of the integration between these two tools was done in Microsoft Visual C# (Microsoft Corporation 2015). STROBOSCOPE was embedded in Darwin optimization framework to evaluate each candidate solution generated by the optimization through simulation.

In order to enable the integration between the optimization framework and STROBOSCOPE, the latter is defined as an object as shown in Figure 4. In other words, STROBOSCOPE was embedded in Darwin optimization framework to evaluate each candidate solution generated by the optimization through simulation. The first two lines define STROBOSCOPE as an object called StropApp. The third line represents the name of the function which will be called to start STROBOSCOPE. Lines 4 and 6 to 8 will try to create an instance of STROBOSCOPE if it is not created yet. Since STROBOSCOPE was not designed to be executed in parallel on a multi-core single processor, several problems were encountered during the implementation. When the optimization framework is run in parallel, it creates multiple instances of the user defined objective function. The number of instances is equal to the number of cores. Since the objective functions defined in this research utilizes STROBOSCOPE, multiple instances of STROBSCOPE are created at the same time. Doing so will force STROBSCOPE to crash. In order to overcome this problem, two steps are necessary. The first step is defining an object for each STROBOSCOPE's instance. For example, if the number of the cores is four, four objects of STROBOSCOPE must be defined as shown in Figure 4. Each STROBOSCOPE instance has a unique object name. The second step is inserting a delay function between the creations of the instances. This is shown in line 5 of Figure 4. The gap between the

creations is set to 3,500 milliseconds. This delay period was chosen by trial and error and it is the shortest possible delay period to avoid crashing.

```
 1  public static object StroboApp;
 2  public static System.Type objDocType;
 3  static object GetStrobo() {
 4    Try  {if (StroboApp == null) {
 5      Thread.Sleep(3500);
 6      objDocType = System.Type.GetTypeFromProgID("Stroboscope.Document");
 7      StroboApp = System.Activator.CreateInstance(objDocType);} }
 8    catch (Exception ex) { MessageBox.Show("Program error: " + ex.Message, "Error1");
 9      StroboApp = null;}
10  return StroboAp };
```

Figure 4: Defining STROBOSCOPE as an object.

To this point, the optimization framework will be able to create multiple instances of STROBOSCOPE without any problem. However, the candidate solutions will only be sent to the first instance since the optimization framework is unaware of other objects. Therefore, the second problem is the synchronization between the optimization framework and STROBSCOPE. This synchronization is necessary to subdivide and distribute the candidate solutions of the population to the correct core based on its rank as early described in Figure 3. This implementation is shown in Figure 5. Similar IF statements are defined for every STROBOSCOPE instance. Line 2 checks the rank of the candidate solution. For example, if the rank is equal to 1, then the optimization framework will call the STROBOSCOPE instance assigned to core 1. Line 3 will pass the decision variables (i.e., x1 to x13) to STROBOSCOPE and return the values of the objective functions.

So far, the parallel computing implementation works without any errors. For each candidate solution, a new STROBOSCOPE instance will be created and the simulation model will run for that candidate solution. At the end of the simulation, that STROBOSCOPE instance will be terminated. The process of creating and terminating each STROBOSCOPE instance takes on average 4 seconds. In the case of evaluating 100,000 candidate solutions, the simulation-based optimization would take 111 hours just to create and terminate a STROBOSCOPE instance. However, by taking advantage of the ability of STROBOSCOPE to run multiple simulation models in a single instance, there is no need to create an instance for each candidate solution. A number of STROBOSCOPE instances equal to the number of cores used for the parallel computing are created at the beginning of the optimization process. Each STROBOSCOPE instance is only created once and all the corresponding candidate solutions are evaluated in those instances. This is done by creating a new simulation model for each candidate solution within a STROBOSCOPE instance. Running too many models in one instance will slow down the speed of the simulation engine and will lead it to crash approximately after running 1,600 simulation models. In addition, the simulation will start to slow down dramatically after running 200 simulation models. To overcome this problem, each STROBOSCOPE instance will close all the simulation models when they reach 200 models. This implementation is shown in lines 4 to 8in Figure 5. In addition, a delay function of 900 milliseconds is inserted before closing the models to prevent STROBOSCOPE from crashing.

5 CASE STUDY

The performance metrics of the simulation-based optimization using stochastic simulation is compared across three different desktop computers to demonstrate the benefits of the proposed framework. The three computers are equipped with Intel Core i7, 3.4 GHz Quad-core processor. Each of these computers is equipped with different RAM size. Computers 1, 2, and 3 have a RAM of 8 GB, 12GB, and 16GB, respectively. The architecture of a processor is shown in Figure 6. The processor has four physical cores

where each physical core has two hardware threads. These hardware threads are also called logical cores. Each physical core can execute two threads at the same time, which is known as simultaneous multithreading (Hillar, 2010).

```
1  i = 0
2  if (rank == 1) {
3     total = temp.testStroboRun(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13);
4     i = i + 1;
5     if (i == 200) {
6        Thread.Sleep(900);
7        strob.objDocType.InvokeMember("CloseAllOutputs");
8        i = 0;}}
```

Figure 5: Directing the candidate solutions towards their corresponding core.

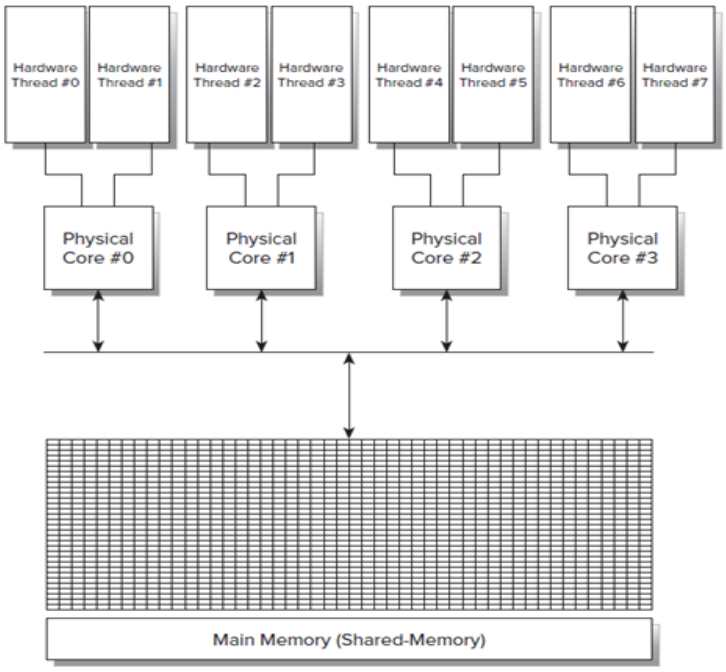

Figure 6: Architecture of Intel i7 Quad-core processor (Hillar, 2010).

The case study is about constructing a precast full span box girder bridge using launching gantry method. The bridge consists of 35 identical spans of length 25 m. The simulation model and the durations of the tasks used in this model can be found in (Mawlana, 2015).

The stochastic simulation-based multi-objective model was run for two eras, where each era has 500 generations and each generation has a population size of 100. The model was set to be executed with the number of cores from 1 to 9 cores (including the master). In the case of 9 cores, one of the cores will be used both as a master and a slave as explained in Section 2. This is done to study the impact of the simultaneous multithreading on the performance metrics of the proposed framework. Figure 7 shows the required time to solve the optimization problem for this case study. Table 1 shows the achieved speedup, efficiency, and the time saving between the three computers, which can be attributed to the difference in

the size of the RAM. All the computers achieved the highest speedup when 4 cores are used. This can be because using more cores results in more time consumed by the algorithm for synchronization and communication overhead. The three computers reduced the computation time by around 37% when 4 cores are used compared to using 1 core. It can be noticed that the efficiency of the three computers are decreasing as the number of cores is increased. Among the three computers, Computer 3 has the highest speedup and the shortest time required. Using Computer 3 resulted in an average time saving of 24% and 22% over Computers 1 and 2, respectively. On the other hand, Computer 2 achieved an average saving of 3% over Computer 1.

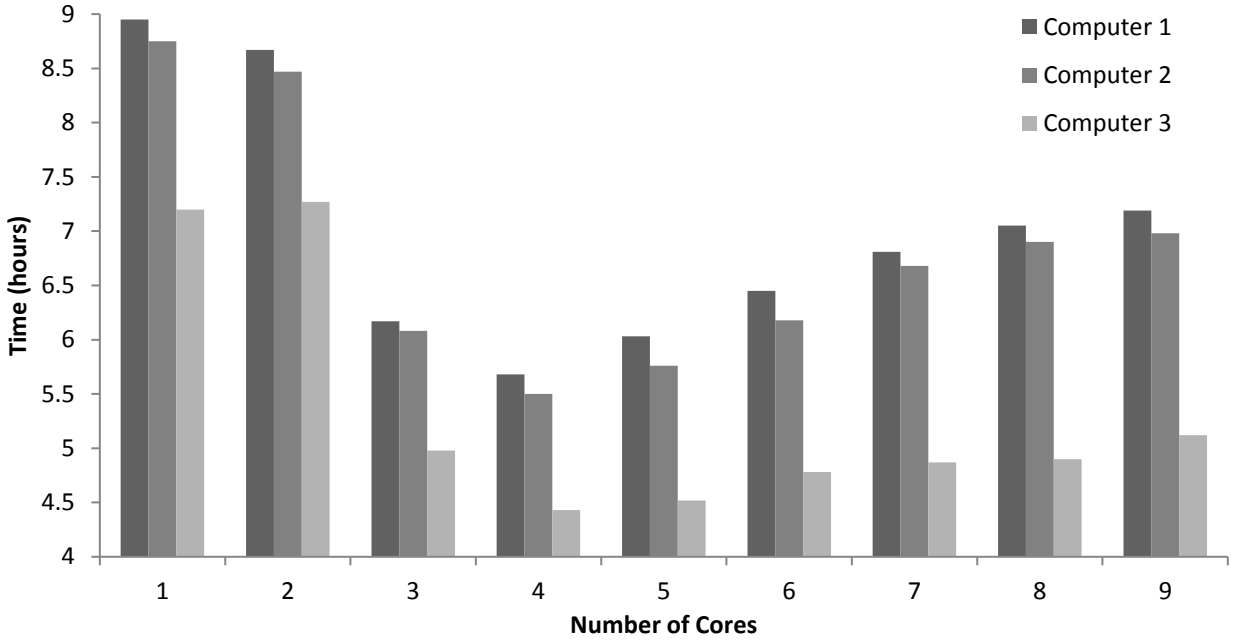

Figure 7: Required time to solve the optimization problem for the Case Study.

Table 1: Achieved speedup, efficiency, and time saving for the Case Study.

Core	T_1 (h)	T_2 (h)	T_3 (h)	SU_1	SU_2	SU_3	EF_1	EF_2	EF_3	T_{1-2} (%)	T_{1-3} (%)	T_{2-3} (%)
1	8.95	8.75	7.2	1.00	1.00	1.00	1.00	1.00	1.00	2.23	19.55	17.71
2	8.67	8.47	7.27	1.03	1.03	0.99	0.52	0.52	0.50	2.31	16.15	14.17
3	6.17	6.08	4.98	1.45	1.44	1.45	0.48	0.48	0.48	1.46	19.29	18.09
4	5.68	5.5	4.43	1.58	1.59	1.63	0.39	0.40	0.41	3.17	22.01	19.45
5	6.03	5.76	4.52	1.48	1.52	1.59	0.30	0.30	0.32	4.48	25.04	21.53
6	6.45	6.18	4.78	1.39	1.42	1.51	0.23	0.24	0.25	4.19	25.89	22.65
7	6.81	6.68	4.87	1.31	1.31	1.48	0.19	0.19	0.21	1.91	28.49	27.10
8	7.05	6.9	4.9	1.27	1.27	1.47	0.16	0.16	0.18	2.13	30.50	28.99
9	7.19	6.98	5.12	1.24	1.25	1.41	0.14	0.14	0.16	2.92	28.79	26.65

6 CONCLUSIONS AND FUTURE WORK

This paper presented a framework for implementing the simulation-based optimization model in a parallel computing environment on a single multi-core processor using the master/slave paradigm. This paper: (1) described the proposed framework; (2) implemented the framework and demonstrated its effectiveness. As demonstrated by the case study in this paper, the proposed framework was able to achieve substantial time savings. On average, it saved 37% of the time required to solve the optimization problem. It was found that 4 cores give the optimum computation time reduction.

The limitations of this work are: (1) the parallel computing was done from the GA perspective only; and (2) the fmGA algorithm used in the case studies is designed to work on multiple computers, and as a result, it uses Message Passing Interface (MPI) which is not optimal for multi-core processor. Future work of this research includes: (1) implementing the GA algorithm using Open Multi-Processors (OpenMP) which is best suited for multi-core processors; (2) applying the parallel computing for both the GA and the simulation calculations; (3) studying the impact of the operating system and the processor model on the computation time; and (4) evaluating the performance of the proposed framework using sensitivity analysis.

ACKNOWLEDGMENTS

We would like to acknowledge the support of Bentley Inc. for providing the Darwin Optimization Framework. Furthermore, the help and support provided by Dr. Zheng Yi Wu is highly appreciated. In addition, I would like to thank Dr. Julio C. Martinez and Dr. Photios G. Ioannou for providing the simulation software STROBOSCOPE.

REFERENCES

Alanjari, P., S. RazaviAlavi, and S. AbouRizk. 2015. "Hybrid Genetic Algorithm-Simulation Optimization Method for Proactively Planning Layout of Material Yard Laydown." *Journal of Construction Engineering and Management* 141(10):06015001.

Alberto, L., C. Azcarate, F. Mallor, and P. Mateo. 2002. "Optimizaton with Simulation and Multiobjective Analysis in Industrial Decision-making: A Case Study." *European Journal of Operational Research* 140(2):373-383.

Cantú-Paz, E., and D. E. Goldberg. 2000. "Efficient Parallel Genetic Algorithms: Theory and Practice." *Computer Methods in Applied Mechanics and Engineering 128(2-4):*221-238.

Cao, M., M. Lu, and J. Zhang. 2004. "Concrete Plant Operations Optimization Using Combined Simulation and Genetic Algorithms." In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, 4204-4209. Shanghai, China: IEEE.

Feng, C.-W., L. Liu, and S. A., Burns. 2000. "Stochastic Construction Time-Cost Trade-Off Analysis." *Journal of Computing in Civil Engineering* 14(2):117-126.

Glover, F., J. Kelly, and M. Laguna. 1996. "New Advances and Applications of Combining Simulation and Optimization." In *Proceedings of the 1996 Winter Simulation Conference*, edited by J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, 144-152. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Hegazy, T., and M. Kassab. 2003. "Resource Optimization Using Combined Simulation and Genetic Algorithms." *Journal of Construction Engineering and Management* 129(6):698-705.

Hillar, G. C. 2010. *Professional Parallel Programming with C#: Master Parallel Extensions with .NET 4*. Indianapolis: Wrox Press.

Kandil, A., and K. El-Rayes. 2006. "Parallel Genetic Algorithms for Optimizing Resource Utilization in Large-Scale Construction Projects." *Journal of Construction Engineering and Management* 132(5): 491-498.

Kandil, A., K. El-Rayes, and O. El-Anwar. 2010. "Optimization Research: Enhancing the Robustness of Large-scale Multiobjective Optimization in Construction." *Journal of Construction Engineering and Management* 136(1):17-25.

Li, H., J. Cao, and P. Love.1999. "Using Machine Learning and Genetic Algorithm to Solve Time-cost Trade-off Problems." *Journal of Construction Engineering and Management* 125(5):347-353.

Li, H., and P. Love. 1997. "Using Improved Genetic Algorithms to Facilitate Time-Cost Optimization". *Journal of Construction Engineering and Management* 123(3):233-237.

Liao, T. W., P. J. Egbelu, B. R. Sarker, and S. S. Leu. 2011. "Metaheuristics for Project and Construction Management: A State-of-the-art Review." *Automation in Construction* 20(5):491-505.

Martínez, J. 1996. "STROBOSCOPE: State and Resource Based Simulation of Construction Processes." PhD Thesis, University of Michigan.

Marzouk, M., and O. Moselhi. 2004. "Multiobjective optimization of Earthmoving Operations." *Journal of Construction Engineering and Management* 103(1):105-113.

Marzouk, M., H. Said, and M. El-Said. 2009. "Framework for Multiobjective Optimization of Launching Girder Bridges." *Journal of Construction Engineering and Management* 135(8):791-800.

Mawlana, Mohammed. 2015. "Improving Stochastic Simulation-based Optimization for Selecting Construction Method of Precast Box Girder Bridges." PhD thesis, Concordia University.

Mawlana, M., and A. Hammad. 2013. "Simulation-based Optimization of Precast Box Girder Concrete Bridge Construction Using Full Span Launching Gantry." In *Proceedings of the 4th Construction Specialty Conference*, CON-159. Montreal, Quebec:CSCE.

Microsoft Corporation. 2015. "Visual C#." Accessed April 10. http://msdn.microsoft.com/enus/library/vstudio/kx37x362.aspx.

Munawar, A., M. Wahib, M. Munetomo, and K. Akama. 2008. "A Survey: Genetic algorithms and the Fast Evolving World of Parallel Computing." In *Proceedings of the 10th International Conference in High Performance Computing and Communications*, 897-902. Dalian, China:IEEE,

Salimi, S. 2014. "Performance Analysis of Simulation-based Multi-objective Optimization of Bridge Construction Processes Using High Performance Computing." Master's Thesis, Concordia University.

Salimi, S., M. Mawlana, and A. Hammad. 2014. "Simulation-based Multiobjective Optimization of Bridge Construction Processes using Parallel Computing." In *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk, S. D. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, 3272-3283. Savannah, GA: Institute of Electrical and Electronics Engineers, Inc.

Salimi, S., M. Mawlana, and A. Hammad. 2015. "Performance Analysis of Simulation-based Multiobjective Optimization Using High Performance Computing." In *Proceedings of the 2nd International Conference on Civil, Building, Engineering Informatics*, 120. Tokyo, Japan.

Wu, Z. Y., Q. Wang, S. Butala, and T. Mi. 2012. "Darwin Optimization Framework User Manual." Bentley Systems Incorporated, Watertown, CT. http://communities.bentley.com/communities/other_communities/bentley_applied_research/m/bentley_applied_research-files/191193.aspx [Accessed January 15, 2015].

Yang, I., Y. Hsieh, and L. Kung. 2012. "Parallel Computing Platform for Multiobjective Simulation Optimization of Bridge Maintenance Planning." *Journal of Construction Engineering and Management* 138(2):215-226.

AUTHOR BIOGRAPHIES

MOHAMMED MAWLANA received his Ph.D. degree in 2015 and is currently an Visiting Assistant Professor at Mississippi State University. His research interests are automation in construction, parallel computing, risk analysis, and simulation optimization. His e-mail is mohammed_mawlana@hotmail.com

AMIN HAMMAD is a Professor at the Concordia Institute for Information Systems Engineering. His research focuses on automation in construction and sustainable infrastructure lifecycle management systems. His current research investigates several methods and techniques including spatio-temporal information modeling and analysis, simulation, visualization, optimization, wireless communications, sensing, auto-identification, and real-time location tracking. His email is hammad@ciise.concordia.ca.