Distributed Computing for the Optimization of Large-Scale Construction Projects

Amr Kandil, University of Illinois at Urbana-Champaign (akandil@uiuc.edu)

Khaled El-Rayes, University of Illinois at Urbana-Champaign (elrayes@uiuc.edu)

Summary

Available construction time-cost trade-off analysis models can be used to generate trade-offs between these two important objectives, however, their application is limited in large-scale construction projects due to their impractical computational requirements. This paper presents the development of a scalable and multi-objective genetic algorithm that provides the capability of simultaneously optimizing construction time and cost large-scale construction projects. The genetic algorithm was implemented in a distributed computing environment that utilizes a recent standard for parallel and distributed programming called the message passing interface (MPI). The performance of the model is evaluated using a set of measures of performance and the results demonstrate the capability of the present model in significantly reducing the computational time required to optimize large-scale construction projects.

1 Introduction

Construction duration can be reduced by using additional resources, which is often accompanied by additional construction costs. This trade-off has been the focus of many research efforts that attempted to identify an optimal trade-off between these two conflicting objectives (Easa 1989, Chan et al. 1996, Hegazy 1999, Gomar et al. 2002, Mattila and Abraham 1998, Burns et al. 1996, Feng et al. 1997, Li and Love 1997, Hegazy and Ersahin 2001, Hegazy and Wassef 2001, Li et al. 1999, Leu and Hwang 2001, Leu et al. 1999, Feng et al. 2000, Adeli and Karim 1997). Available models that used traditional optimization methods (e.g. linear programming, and dynamic programming) suffered from combinatorial explosion when applied to large construction projects (Adeli and Karim 1997, Mattila and Abraham 1998, Burns et al. 1996). Other models that utilized genetic algorithms (GAs) were not capable of providing an efficient solution for large-scale construction projects (Hegazy and Petzold 2003).

Existing GA models were applied to projects ranging from 7 to 66 activities as shown in Figure 1, while the median number of activities in a construction schedule was found to be slightly greater than 300 activities in a recent survey of construction planners (Liberatore et al. 2001). The example construction project composed of 66 activities required 6 hours of continuous genetic algorithm computations to optimize (Hegazy and Petzold 2003). This computational time is estimated to increase as a function of the square of the project size, which indicates that projects with a larger number of activities are even more computationally expensive (Chan et al. 1996). Distributed computing has been recently applied to solve large-scale optimization problems in a number of civil engineering disciplines including: 1) finite element analysis (Sziveri and Topping 2000); 2) structural design (Adeli 2000); and 3) water network design (Balla and Lingireddy 2000; Alonso et al. 2000). While these research studies have provided efficient solutions for these large-scale engineering problems, there has been little or no reported research exploring the promising potential of distributed computing in optimizing large-scale construction projects.

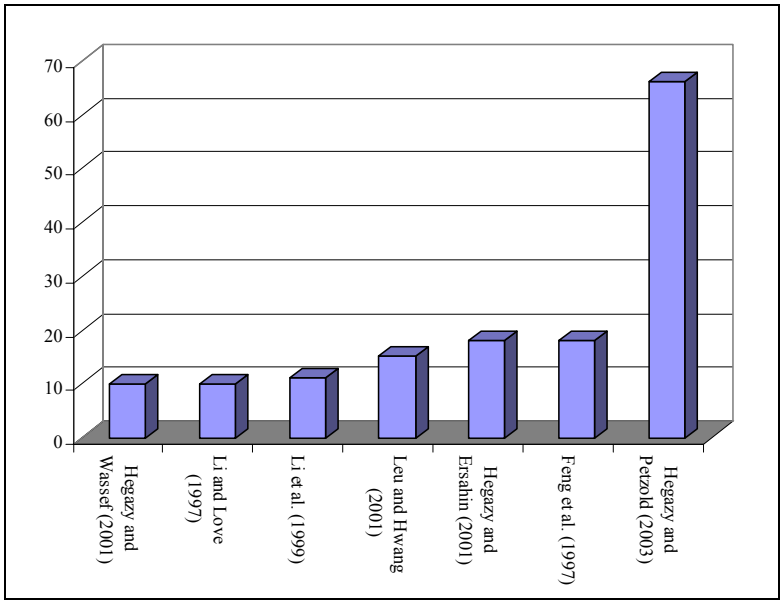

Figure1. Project Sizes in Previous Models

This paper presents the development of a robust optimization model capable of minimizing the cost and duration of real-life large-scale construction projects. The model utilizes: 1) a multi-objective genetic algorithm that supports the simultaneous optimization of project cost and time; 2) a distributed computing environment based on the global parallel GA paradigm; and 3) three performance measures to evaluate the effectiveness of the parallel implementation.

2 Multi-Objective Genetic Algorithm

The present model utilizes an advanced multi-objective genetic algorithm that evaluates the trade-off between project time and cost (Deb 2001; Deb et al. 2001; Zitzler et al. 2001; Watanabe et al. 2002; Hiroyasu et al. 2002). The genetic algorithm is developed in two main phases: 1) genetic algorithm formulation; and 2) genetic algorithm implementation.

2.1 Genetic Algorithm Formulation

The formulated GA aggregates the decision variables in this optimization problem into a group artificial strings that form a population. Each of these strings represents a feasible solution for the trade-off problem, and holds a single value of a decision variable that represents the level of resource utilization (n) for each activity in the project. Resource utilization combines the three decision variables considered in this model namely: (1) construction method (m), which indicates the availability of different types of materials and/or methods that can be utilized; (2) crew formation (f), which represents feasible sizes and configurations for construction crews; and (3) crew overtime policy (p), which represents available overtime hours and nighttime shifts. The GA strings formulated for this model are binary strings with lengths equal to the number of activities (I) multiplied by the number of bits required to represent the available resource utilizations (N) for each activity. For example the binary string shown in Figure 2 represents a project composed of 5 activities that have 7 resource utilization alternatives each. This string is composed of a total of 15 bits, since the 7 alternatives are represented by 3 binary bits.

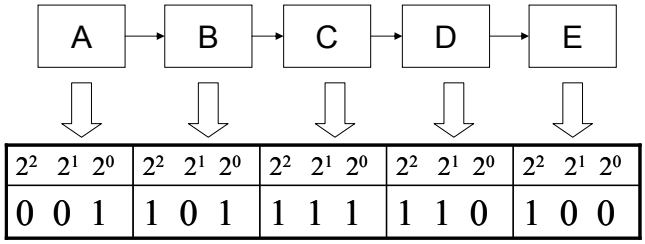

Figure 2. Binary String Formulation.

The current model is formulated to enable the optimization of construction cost and duration. These two equations are used to evaluate the fitness of each string/solution generated during the GA operations as shown in equations 1 and 2.

$$\text{Minimize Project Time} = \sum_{i=1}^{I} T_i^n \quad (1)$$

Where, T_i^n = duration of activity (i) on the critical path using resource utilization (n). In this model, project duration is estimated using newly developed algorithms for the scheduling of highway construction (El-Rayes 2001, El-Rayes and Moselhi 2001).

$$\text{Minimize Project Cost} = \sum_{i=1}^{I} [(M_i^n + D_i^n \times R_i^n) + (B_i^n)] \quad (2)$$

Where, M_i^n = material cost of activity (i) using resource utilization (n); D_i^n = duration of activity (i) using resource utilization (n); R_i^n = daily cost rate in $/day of resource utilization (n) in activity (i); B_i^n = subcontractor lump sum cost for resource utilization (n) in activity i, if any.

2.2 Genetic Algorithm Implementation

The present multi-objective GA is implemented in three main stages: 1) GA initialization; 2) function evaluation; and 3) evolution, as shown in Figure 3. The GA initialization generates a set of random solutions to form the initial population of feasible solutions. The function evaluation stage evaluates the project costs and durations for the different solutions in each of the generated populations. The fitness values calculated by this stage are then used by the evolution stage to generate new populations of feasible solutions. The evolution stage performs four main functions: 1) calculates the Pareto optimal rank of all solutions in each population; 2) calculates the crowding distance of all solutions in each population; 3) creates child populations; and 4) creates combined populations, as shown in Figure 3. First, Pareto optimal rank is calculated by ranking the solutions in the population according to their domination of other solutions, where a solution is identified as dominant if it is better than all other solutions in all of the considered optimization objectives simultaneously. Second, the crowding distance of each solution is calculated, by evaluating the closeness of neighboring solutions to the solution considered. The crowding distance values help the algorithm spread the obtained solutions over a wider Pareto optimal front instead of converging to points that cover only a small part of the tradeoff curve (Deb et al. 2001). Third, a child population is created using selection, crossover and mutation based on the calculated optimal rank and crowding distance. Fourth, child and parent populations are then combined to form a new combined population. This new combined population acts as a vehicle for elitism, where good solutions of the initial parent population are passed on to the following generations to avoid loss of good solutions once they are found (Deb et al. 2001). The new combined population (Ng) is then sorted using the niched comparison rule and the top solutions from the combined population are kept to form the parent population of

the next generation. This sorting rule selects solutions with higher Pareto optimal ranks and breaks ties between solutions with the same rank by favoring solutions with higher crowding distances.

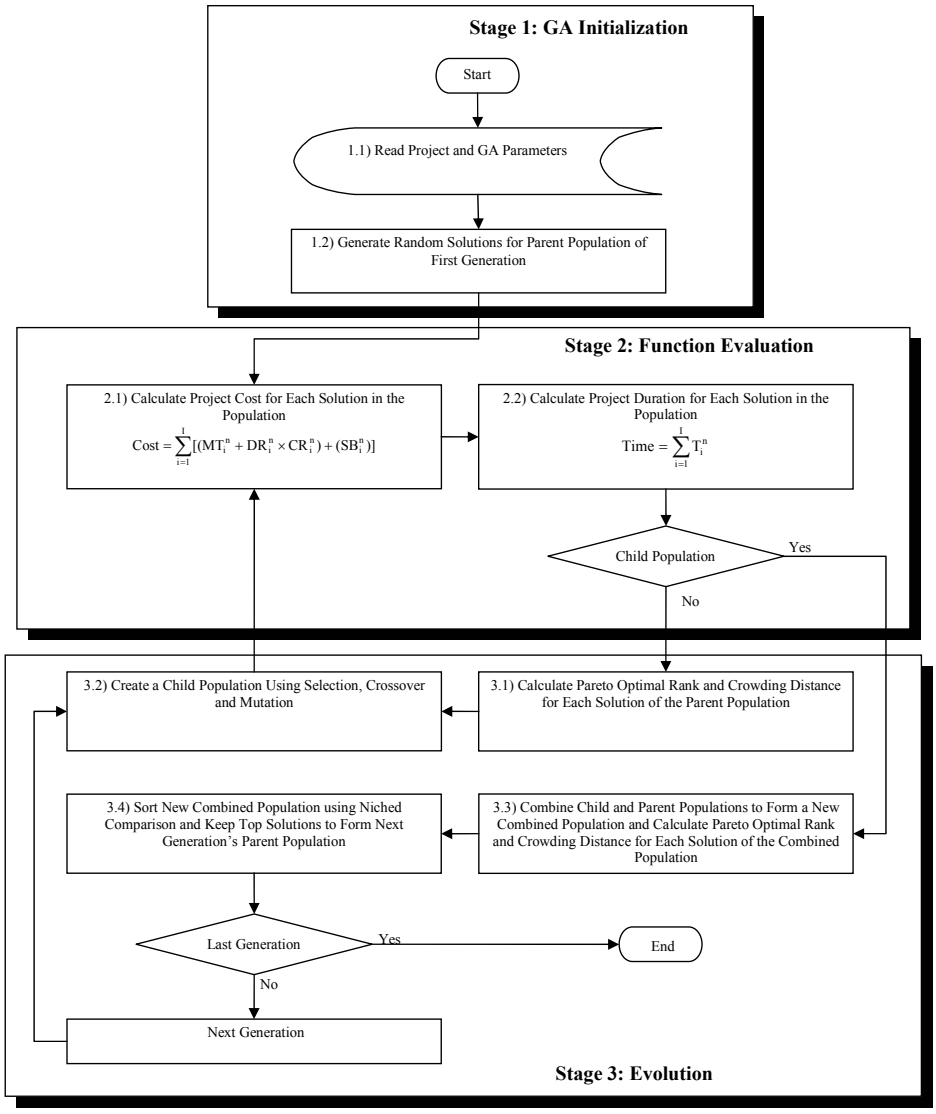

Figure 3. Multi-objective Genetic Algorithm.

3 Distributed Computing Environment

The developed model relies on a robust distributed computing environment that makes the optimization of large scale construction projects feasible. The design of this environment follows the global parallel GA paradigm in which a manager processor administers the operation of a number of worker processors that only perform the function evaluation stage of the developed multi-objective GA. The manager processor executes all stages of the developed multi-objective GA, and performs a number of administrative tasks such as distributing sub-populations of solutions to worker processors, and gathering the fitness values associated with each solution in the population. These tasks require communication between the manager and

worker processors, which is implemented using the message passing interface (MPI) (Snir et al 1998). The main reason for selecting MPI to implement this environment is its wide acceptance as a standard for distributed programming in both academia and the industry, and its support for portability (Gropp et al. 1999). This makes the developed environment operable on a wide range of computing systems that includes Linux clusters, and networks of workstations. The current distributed computing environment is implemented in two steps: 1) structuring the multi-objective GA for parallelization; and 2) applying the MPI communication functions.

3.1 Structuring the Parallel Genetic Algorithm

The execution of the present distributed environment consists of a number of independent processes that execute dissimilar code on separate processors. Processes executed by the manager processor implement all three stages of the developed multi-objective GA, while processes performed by worker processors execute the function evaluation stage. These processes communicate through calls to MPI functions that pass messages between processes using a communicator (Gropp et al.1999). The present distributed environment is implemented in the following six steps:

1- Include the MPI header file, which contains the definitions and prototypes of all the MPI functions used in the implementation of the environment, at the beginning of the multi-objective GA code (Girianna 2002).

2- Initialize MPI communication using an MPI initialization call at the beginning of the main function of the multi-objective GA code.

3- Set the rank of the processes in the communicator by calling the MPI_rank function. The process rank is the identification number given to each process in the communicator.

4- Set the communicator size by calling the MPI_Comm_size function, surveys the number of processes that will be executed by the environment.

5- Include the statements and functions that execute the different stages of the multi-objective GA and the different MPI communication functions.

6- Terminate communications by calling MPI_finalize function after all the MPI function calls have been made (Pacheco 1998).

3.2 Applying the MPI Communication Functions

There are two main MPI communication functions used in the development of the present distributed environment: 1) the MPI_Scatter function; and 2) the MPI_Gather function. MPI_Scatter divides the message sent at the sending process into x equal segments then sends the x messages to all receiving processes, and also receives the sent messages from sending processes. The MPI_Scatter function is used in the present environment for sending individuals from the manager processor to the worker processors as shown in Figure 4. The send buffer at the manager processor contains the decoded values of the whole population. The corresponding receive buffer at each of the worker processors accepts the decoded value of a single individual. The MPI_Gather function on the other hand has the same syntax as MPI_Scatter, but behaves the exact opposite way. This function takes the same parameters as MPI_Scatter, but collects the messages arriving to the receiving process and assembles them in the receive buffer instead of sending them. The MPI_Gather function is used in the present environment to return fitness values of the evaluated solutions to the manager processor as shown in Figure 4. The send buffer at each worker process contains the value of the objective functions evaluated for each solution. The receive buffer at the manager processor contains the objective function values for all solutions in the population which equals the population size (N) multiplied by the number of objective functions (O).

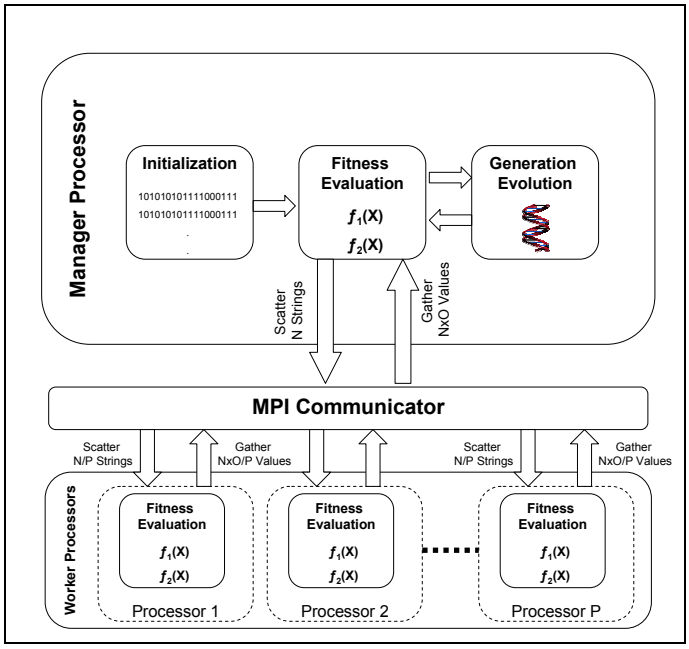

Figure 4. Parallelized implementation of the model

4 Measures of Performance

The current model was tested at the University of Illinois's Turing Linux Cluster, which consists of 208 dual-processor machines (for a total of 416 processors) with two 1 GHz Pentium III processors and 1 GB of RAM each. The cluster was accessed through a 1.5 GHz quad-processor Pentium III Xeon front-end server. The primary network connecting the cluster machines is a high-bandwidth, low-latency Myrinet network. In addition, all machines in the cluster are also connected by a 100 Mbs switched, full-duplex Ethernet and there is a 1 Gbs link between the front-end and the primary switch (CSE 2003).

Different numbers of processors were used for the performance evaluation tests, and the developed algorithm was tested using three sample construction projects composed of 180, 360, and 720 activities respectively. Each of these construction projects required a different string length and hence a different population size (Reed et al. 2002). The performance of the implemented model was evaluated using three main measures namely the elapsed time, parallel speed-up, and parallel GA efficiency.

4.1 Elapsed Time

The elapsed time (T_p) is the time required by the parallel model to perform the GA functions explained in Figures 3 and 4. Elapsed time is composed of two main components: 1) evaluation time (E_t), which estimates the time required for function evaluation by processors; and 2) the total communication time (C_t), which is the time required to perform the communication functions in the model. Elapsed time is given by equation 3 (Cantú-Paz 2000).

$$T_p = E_t + C_t = \left(\frac{S \times T_f}{p^x}\right) + (\alpha + \beta \times p) \quad (3)$$

Where, S= number of individuals in the GA population; p= number of processors used; x= empirical factor associated with efficiency of the implementation; T_f= time required for the

evaluation of a single individual; α= communication time constant which depends on the number of individuals in the population and the size of GA string; β= slope of the communication function which depends on the number of individuals in the population and network latency.

The total elapsed time per generation for the developed model was measured for each of the tested projects using a varying number of processors as shown in Figure 5. The time savings per generation were found to be greatest for the larger project. The magnitude of these time savings is amplified by the fact that the number of generations needed to optimize larger problems is larger than that of a smaller one. This makes the application of the developed model more efficient on large problems.

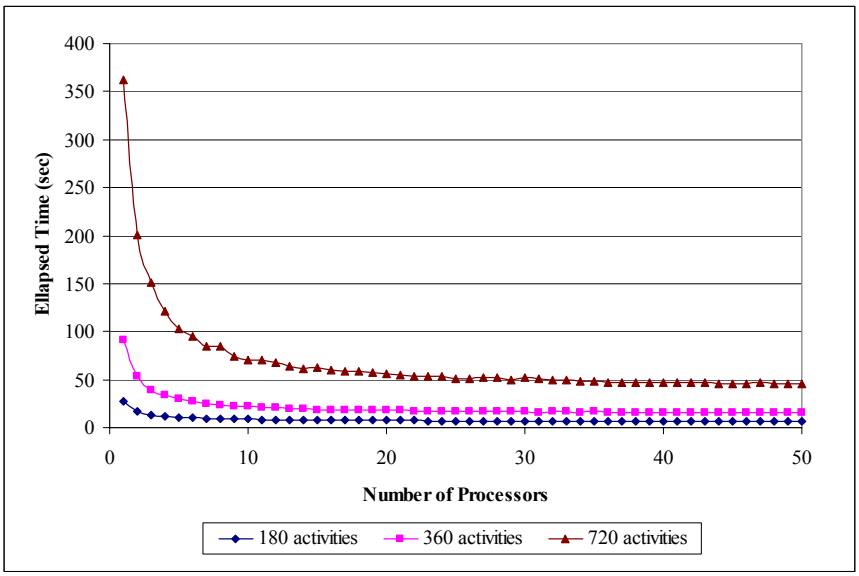

Figure 5. Total Elapsed Time per Generation.

4.2 Parallel Speed up

Parallel speed up (ψ) is a measure of effectiveness that compares the performance of the parallel GA to that of a serial GA, and can be calculated using equation 4 (Cantú-Paz 2000).

$$\psi = \frac{T_s}{T_p} \tag{4}$$

Where, ψ= parallel speedup; T_p= elapsed time of the parallel GA; and T_s= elapsed time of the serial GA.

The parallel speedup of the developed model was measured for the same set of projects and number of processors shown in Figure 5. The results show that a parallel speedup of 8 and 4 times the speed of the serial algorithm can be achieved for the 720-activity and 180-activity projects, respectively. These obtained parallel speedups clearly displayed Amdahl's effect, which predicts that speedup would increase as the problem size increases (Quinn 2004).

4.3 Parallel GA Efficiency

Parallel GA efficiency is used to measure the deviation of the parallel GA implementation from ideal conditions, and is calculated as shown in equation 7 (Cantú-Paz 2000).

$$E = \frac{T_s}{p \times T_p} \quad (5)$$

Where, p= number of processors used; T_p= elapsed time of the parallel GA; and T_s= elapsed time of the serial GA.

Ideal conditions occur if the gains made by the addition of each new processor do not diminish. This condition is also called linear speed up, which rarely occurs in reality due to the presence of network latencies and other factors that lead to loss of efficiency. The 720-activity project was found to have a higher efficiency than the remaining projects, which can be attributed to the higher speedups larger projects achieve.

5 Endnotes

A robust multi-objective GA model was developed for optimizing project cost and duration in large construction projects using a distributed computing environment. The model was developed in three stages that: 1) developed a robust multi-objective GA; 2) implemented an efficient distributed computing environment; and 3) evaluated the performance of the model using four measures . The measures of performance employed showed that the developed parallel environment was capable of optimizing large construction projects up to 8 times faster than a serial GA. Further, large construction projects were found to have higher efficiencies than smaller ones. This makes the current distributed optimization model capable of efficiently analyzing large-scale real-life construction projects that were previously infeasible to consider.

6 References

Adeli, H. (2000) “High Performance Computing for Large-Scale Analysis, Optimization, and Control,” *Journal of Aerospace Engineering*, ASCE, 13(1), 1-10.

Adeli, H. and Karim, A. (1997). "Scheduling/Cost Optimization and Neural Dynamics Model for Construction Projects," J. Constr. Engrg. Mgmt., ASCE, 123(4), 450-458.

Alonso, J., Alvarriuz, F., Guerrero, D., Hernández, V., Ruiz, P., Vidal, A., Martinez, F., Vercher, J., and Ulanicki, B. (2000). “Parallel Computing in Water Network Analysis and Leakage Minimization” *J. Water Resour. Plng. and Mgmt*, ASCE, 126(4), 251-260.

Balla, M., and Lingireddy, S. (2000) “Distributed Genetic Algorithm Model on Network of Personal Computers” *J. Comp. Civ. Engrg*., ASCE, 14(3), 199-205.

Burns, S., Liu, L., and Feng, C. (1996) "The LP/IP hybrid Method for Construction Time-Cost Trade-Off Analaysis," *J. Constr. Mgmt. Econ.*, 14, 265-276.

Cantú-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publishers, Boston, MA.

Cantú-Paz, E. (1997) “A Survey of Parallel Genetic Algorithms” *ILLGAL Report No. 97003*, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana Champaign, Urbana, Illinois, USA.

University of Illinois at Urbana Champaign Computation Science and Engineering Program (CSE) (2003) “The Turing Linux Cluster.” http://turing.cse.uiuc.edu.

Chan, W., Chua, D., and Kannan, G. (1996) “Construction Resource Scheduling with Genetic Algorithms” *J. Constr. Engrg. Mgmt.*, ASCE, 122(2), 125-132.

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, LTD, New York, NY.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2001). "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-objective Optimization." *KANGAL Report 200001*, Genetic Algorithm Laboratory, Indian Institute of Technology, Kanpur, India.

Easa, S. (1989) "Resource Leveling in Construction by Optimization," *J. Constr. Engrg. Mgmt.*, ASCE, 115(2), 302-316.

El-Rayes, K. (2001) "Optimum Planning of Highway Construction Under the A + B Bidding Method," *J. Constr. Engrg. Mgmt.*, ASCE, 127(4), 261-269.

El-Rayes, K., and Moselhi, O. (2001) "Optimizing Resource Utilization for Repetitive Construction Projects," *J. Constr. Engrg. Mgmt.*, ASCE, 127(1), 18-27.

Feng, C., Liu, L., and Burns, S. A. (1997). "Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems ." *J. Comp. Civ. Engrg.*, ASCE, 11(3), 184-189.

Feng, C., Liu, L. and Burns, S. (2000) "Stochastic Construction Time-Cost Trade-Off Analysis," *J. Comp. Civ. Engrg.*, ASCE, 14(2), 117-126.

Gomar, J., Haas, C. and Morton, D. (2002) "Assignment and Allocation Optimization of Partially Multiskilled Workforce," *J. Constr. Engrg. Mgmt.*, ASCE, 128(2), 103-109.

Gropp, W., Lusk, E., and Thakur, R. (1999). *Using MPI-2: Advanced Features of the Message Passing Interface*, Massachusetts Institute of Technology, Cambridge, MA.

Girianna, M. (2002). "Dynamic Signal Coordination Model for a Network with Oversaturated Intersections," *Phd/thesis*, Univ. of Illinois at Urbana Champaign, Urbana, Illinois, USA.

Hegazy, T. and Petzold, K. (2003) " Genetic Optimization for Dynamic Project Control," *J. Constr. Engrg. Mgmt*, ASCE, 129(4), 396-404.

Hegazy, T., and Wassef, N. (2001) "Cost Optimization in Projects with Repetitive Nonserial Activities" *J. Constr. Engrg. Mgmt.*, ASCE, 127(3), 183-191.

Hegazy, T., and Ersahin, T. (2001) "Simplified Spreadsheet Solutions. II: Overall Schedule Optimization, " *J. Constr. Engrg. Mgmt.*, ASCE, 127(6), 469-475.

Hegazy, T. (1999). "Optimization of Resource Allocation and Leveling Using Genetic Algorithms." *J. Constr. Engrg. Mgmt.*, ASCE, 125(3), 167-175.

Hiroyasu, T., Miki, M., Watanabe, S., Kamiura, J., and Okuda, T. (2002) "MOGADES: Multi-Objective Genetic Algorithm with Distributed Environment Scheme." *Special Report*, Doshisha University, Kyoto, Japan.

Liberatore, M., Pollack-Johnson, B., and Smith, C. (2001) "Project Management in Construction: Software Use and Research Directions" *J. Constr. Engrg. Mgmt.*, ASCE, 127(2), 101-107.

Leu, S., and Hwang, S. (2001) "Optimal Repetitive Scheduling Model with Shareable Resource Constraint." *J. Constr. Engrg. Mgmt.*, ASCE, 125(6), 420-427.

Leu, S., and Yang, C. (1999) "GA-Based Multicriteria Optimal Model for Construction Scheduling." *J. Constr. Engrg. Mgmt.*, ASCE, 127(4), 270-280.

Li, H. and Love, P. (1997) "Using Improved Genetic Algorithms to Facilitate Cost Optimization," *J. Comp. Civ. Engrg.*, ASCE, 13(3), 233-237.

Li, H., Cao, J., and Love, P. (1999) “Using Machine Learning and GA to Solve Time-Cost Trade-Off Problems.” *J. Constr. Engrg. Mgmt.*, ASCE, 125(5), 347-353.

Mattila, K. and Abraham, D. (1998) "Resource Leveling of Linear Schedules Using Integer Linear Programming," *J. Constr. Engrg. Mgmt.*, ASCE, 124(3), 232-244.

Maxwell, D., Back, E. and Toon, J. (1998) "Optimization of Crew Configuration Using Activity-Based Costing," *J. Constr. Engrg. Mgmt.*, ASCE, 124(2), 162-168.

Pacheco, P. S. (1998) “A user's guide to MPI,” *Technical report*, Department of mathematics, University of San Francisco, San Francisco, CA.

Quinn, M.J. (2004). *Parallel Programming in C with MPI and OpenMP*, McGraw Hill, New York, NY.

Reed, P., Minsker B. S., and Goldberg, D. E. (2002). "Simplifying Multi-objectiveOptimization: An Automated Design Methodology for the Nondominated Sorted Genetic Algorithm-II." *Water Resources Research*, Submitted.

Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J. (1998). *MPI: The Complete Reference, Volume1, The MPI Core*, Massachusetts Institute of Technology, Cambridge, MA.

Sziveri, J., and Topping, B. (2000) “Transient Dynamic Nonlinear Analysis Using MIMD Computer Architectures.” *J. Comp. Civ. Engrg.*, ASCE, 14(2), 79-91.

Thakur,R., Gropp,W., and Lusk, E. (1997) “Users Guide for ROMIO: A High-Performance, Portable MPI-IO Implementation.” *Technical Report ANL/MCS-TM-234*, Mathematics and Computer Science Division, Argonne National Laboratory, Chicago, IL.

Watanabe, S., Hiroyashu, T. and Miki, M. (2002) “Parallel Evolutionary Multi-Criterion Optimization for Block Layout Problems,” *Special Report*, Doshisha University, Kyoto, Japan.

Zitzler, E., Laumanns, M., and Thiele, L. (2001) “SPEA2: Improving the Strength Pareto Evolutionary Algorithm.” *TIK-Report 103*, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.