# Optimization Research: Enhancing the Robustness of Large-Scale Multiobjective Optimization in Construction

Amr Kandil[1]; Khaled El-Rayes[2]; and Omar El-Anwar[3]

**Abstract:** Many construction planning problems require optimizing multiple and conflicting project objectives such as minimizing construction time and cost while maximizing safety, quality, and sustainability. To enable the optimization of these construction problems, a number of research studies focused on developing multiobjective optimization algorithms (MOAs). The robustness of these algorithms needs further research to ensure an efficient and effective optimization of large-scale real-life construction problems. This paper presents a review of current research efforts in the field of construction multiobjective optimization and two case studies that illustrate methods for enhancing the robustness of MOAs. The first case study utilizes a multiobjective genetic algorithm (MOGA) and an analytical optimization algorithm to optimize the planning of postdisaster temporary housing projects. The second case study utilizes a MOGA and parallel computing to optimize the planning of construction resource utilization in large-scale infrastructure projects. The paper also presents practical recommendations based on the main findings of the analyzed case studies to enhance the robustness of multiobjective optimization in construction engineering and management.

**DOI:** 10.1061/(ASCE)CO.1943-7862.0000140

**CE Database subject headings:** Optimization models; Parallel processing; Resource management; Housing; Multiple objective analysis; Linear analysis; Algorithms; Construction industry.

**Author keywords:** Optimization models; Parallel processing; Resource management; Housing; Multiple objective analysis; Linear analysis; Algorithms.

## Introduction

The construction industry in the United States is one of the largest sectors of the economy, contributing $1,175 billion to the nation's economy [U.S. Bureau of Economic Analysis (BEA) 2007]. What makes this contribution even more significant is the impact that construction has on other sectors of the economy including the manufacturing sector (Carty 1995). Construction engineering and management research is of great importance to the viability and continued success of this important industry (Levitt 2007). The importance of construction research is underscored by the marginal gains in productivity that the construction industry has achieved in comparison to manufacturing industries (Carty 1995). These small productivity gains could in part be attributed to the increased complexity of construction projects, which requires the development of more effective methods for managing those projects.

[1]Assistant Professor, Division of Construction Engineering and Management, School of Civil Engineering, Purdue Univ., 550 Stadium Mall Dr., West Lafayette, IN 47907 (corresponding author). E-mail: akandil@purdue.edu

[2]Associate Professor. Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana Champaign, 3127 NCEL, 205 N. Mathews Ave., Urbana, IL 61801. E-mail: elrayes@uiuc.edu

[3]Assistant Professor, Dept. of Construction Management, Univ. of Washington, Seattle, WA, 98195. E-mail: elanwar@uw.edu

Note. This manuscript was submitted on August 4, 2008; approved on August 19, 2009; published online on December 15, 2009. Discussion period open until June 1, 2010; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Construction Engineering and Management***, Vol. 136, No. 1, January 1, 2010. ©ASCE, ISSN 0733-9364/2010/1-17–25/$25.00.

One of the main complexities that construction projects face is the presence of multiple and conflicting objectives that need to be achieved for the successful completion of the project (Burns et al. 1996). These objectives and their relative importance vary from one project to another, and they often include minimizing construction time and cost while maximizing safety, quality, and sustainability. Construction engineers and managers often need to analyze and optimize the impact of their decisions on these important project objectives. To support construction engineers and managers in this critical and challenging task, a number of research studies focused on developing optimization models that utilize multiobjective optimization algorithms (MOAs) (El-Rayes and Kandil 2005; Hyari and El-Rayes 2006; Kandil and El-Rayes 2006a; Lee and Kim 2007; Liu et al. 1997; Liu and Frangopol 2005; Chan and Hu 2002). These algorithms could be classified into two main types: (1) naturally inspired algorithms that mimic a natural process in their optimization approach and (2) analytical algorithms that utilize a mathematical approach to the optimization problem. Regardless of the MOA used, the developed models need to be (1) efficient to enable the optimization of large-scale real-life construction problems in a practical and feasible time and (2) effective in identifying optimal or near optimal solutions to the optimization problem. These two criteria are collectively referred to as the robustness of the algorithm, and are often challenging to evaluate and attain.

## Objectives

The main objective of this paper is to outline a number of methods for evaluating and attaining robustness in construction multiobjective optimization models. To accomplish this objective, this paper presents: (1) a review of recent construction multiobjective optimization research and approaches to increase the robustness of optimization algorithms; (2) two large-scale case studies to analyze the robustness of naturally inspired and analytical MOAs; and (3) recommendations for increasing the robustness of construction multiobjective optimization models.

## Literature Review

Existing research studies that focused on construction multiobjective optimization have used a number of naturally inspired and analytical optimization algorithms. This section reviews and discusses recent research efforts that used (1) naturally-inspired algorithms such as genetic algorithms (GAs), ant-colony optimization (ACO), and particle swarm optimization (PSO) and (2) analytical optimization algorithms such as linear, integer, and dynamic programming. The review also covers a number of approaches that were used to enhance the robustness of optimization algorithms such as parallel and distributed computing.

### *Naturally Inspired Optimization Algorithms*

Naturally inspired optimization algorithms have been developed in an effort to computationally mimic natural processes that lead to optimal results (Goldberg 1989). For example, GAs were first introduced by Holland (1975), and are used as search and optimization algorithms that mimic genetic operations based on Darwin's theory of natural selection (Goldberg 1989; Marler and Arora 2004). They adopt the survival of the fittest and the structured exchange of genetic materials among population members over successive generations as a basic mechanism for the search process (Goldberg 1989; El-Rayes and Kandil 2005). GAs have been widely used to solve multiobjective construction engineering and management problems, including (1) deterministic time-cost tradeoff analysis for construction project planning (Feng et al. 1997; Kandil and El-Rayes 2006b; Kasaeian et al. 2007; Schaumann et al. 1998; Zheng et al. 2004); (2) stochastic time-cost tradeoff analysis (Feng et al. 2000; Zheng and Ng 2005); (3) time-cost-quality tradeoff analysis (Kandil and El-Rayes 2006a); (4) linear repetitive projects scheduling (Hyari and El-Rayes 2006); (5) site layout planning (El-Rayes and Khalafallah 2005; Khalafallah and El-Rayes 2006); (6) lighting design optimization for nighttime highway construction (El-Rayes and Hyari 2005); (7) postdisaster temporary housing arrangements optimization (El-Anwar and El-Rayes 2007); (8) repair and rehabilitation planning in bridge management systems (Lee and Kim 2007; Liu et al. 1997; Liu and Frangopol 2005); (9) production scheduling for precast plants (Chan and Hu 2002); and (10) double sampling plan optimization for construction quality inspection (Cheng and Chen 2007). Due to this wide utilization of GAs in construction multiobjective optimization studies, the first and second case studies in this paper will focus on analyzing the robustness of this optimization technique.

Another important naturally inspired optimization technique is ACO which was developed by Dorigo and Maniezzo (1996). This algorithm models the natural process by which ant colonies are able to find the shortest route between their nest and a source of food. This process, therefore, simulates the use of pheromone trails, which ants deposit whenever they travel as a form of indirect communication (Elbeltagi et al. 2005). The effectiveness of this optimization technique was tested in a number of multiobjective construction optimization models including: (1) Xiong and Kuang (2008) that used ACO in solving time-cost tradeoff problems for construction project planning; and (2) Afshar et al. (2007) that used ACO to solve time-cost-quality tradeoff problems. A similar optimization technique called PSO was developed by Kennedy and Eberhart (1995). PSO is also a naturally inspired algorithm that imitates the social behavior of a flock of migrating birds trying to reach an unknown destination. Therefore, the optimization process in PSO involves: (1) local search, where birds use intelligence to learn from their own experience and (2) global search, where birds use social interaction to learn from the experience of other birds in the flock (Elbeltagi et al. 2005). An example of a construction multiobjective optimization model that used PSO is that of Yang (2007) that used PSO to solve the time-cost tradeoff problems for construction project planning.

### *Analytical Optimization Algorithms*

A number of construction multiobjective optimization models have also used analytical algorithms. Many of these models used linear, integer, and dynamic programming algorithms. For example, Burns et al. (1996) presented an optimization model that used a hybrid method that integrated linear and integer programming for finding optimal time-cost trade-offs. The model was applied in two main stages: (1) linear programming stage to find lower bounds of the tradeoffs and (2) integer programming stage to obtain the exact tradeoff. Another model that used linear and integer programming was developed by Mattila and Abraham (1998). This model was dedicated to the optimization of repetitive projects and attempted to level resource utilization for this type of projects. Gomar et al. (2002) introduced a model that attempts to optimize the allocation of mulitskilled work crews using integer linear programming. The developed model optimized the allocation of multiskilled workforces in order to increase labor efficiency on the project and was validated using CII Model Plant Data. Ipsilandis (2007) used a parametric objective function to develop a multiobjective linear programming model for scheduling linear repetitive projects. In this model, different optimal solutions can be generated by changing the values of the objective function parameters.

A number of dynamic programming models have also been developed to optimize resource utilization in repetitive construction projects. These models focused on (1) minimizing the overall project cost by striking an optimal balance between minimizing the project direct cost and duration (Moselhi and El-Rayes 1993; El-Rayes and Moselhi 2001) and (2) minimizing the total combined bid in A+B highway contracting methods (El-Rayes 2001).

### *Parallel and Distributed Computing*

Naturally inspired and analytical optimization algorithms have relative strengths that make each of them better suited for certain types of optimization problems. However, when both types of algorithms are applied to large and more complex problems the time required to solve the problem increases, decreasing their efficiency (Balla and Lingireddy 2000; Cantú-Paz 1997a). This led to a number of attempts to increase efficiency in solving large and complex problems using a number of approaches including: (1) global parallelization and (2) coarse-grained parallelization (Cantú-Paz 1997a).

Global parallel optimization algorithms have a single search space (i.e., not segmented) and behave in a way similar to the serial ones (Cantú-Paz 1997b). These algorithms utilize the manager/worker paradigm or parallel computing. This paradigm

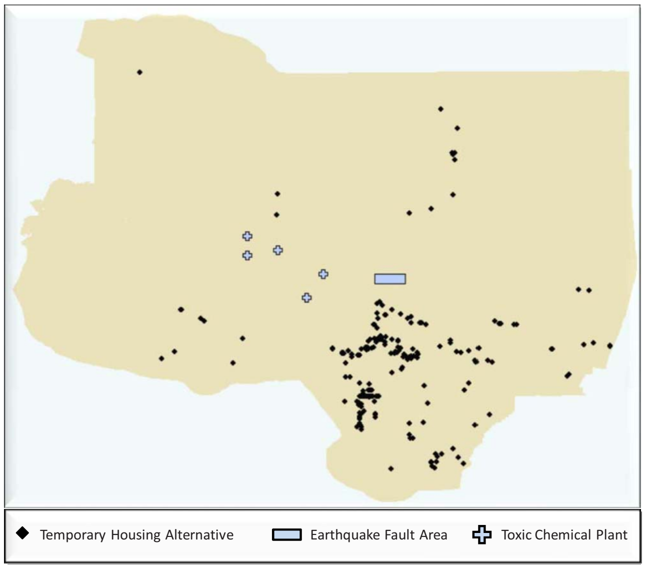

**Fig. 1.** Impacted area by the earthquake

is reported to provide a number of advantages including the increased flexibility in program structuring, and the ease of the data partitioning, mapping and load balancing procedures. Despite these advantages this environment was also reported to suffer from efficiency losses as the number of used processors increases (De Santiago and Law 2000; Balla and Lingireddy 2000; Alonso et al. 2000). The second category of parallel optimization algorithms is the coarse-grained parallel algorithms, which is also known as island or distributed GAs. These algorithms implement a form of data parallelism in which data are distributed over the processors and each processor implements a separate copy of the program independently and occasionally communicates with the other processors (Thiagarajan and Aravamuthan 2002).

## First Case Study: Temporary Housing Planning

In order to analyze and compare the robustness of naturally inspired and analytical MOAs, a large-scale case study of a postdisaster temporary housing planning is analyzed. The case study requires decision-makers to identify optimal plans of temporary housing arrangements for 20,000 displaced families after a major earthquake. There are 222 available temporary housing alternatives in and around the impacted area with a total capacity of 39,001 families, as shown in Fig. 1. Accordingly, there is a very large number of possible plans of temporary housing arrangements, which needs a robust optimization methodology to analyze in a reasonable and practical computational time.

The present case study includes two main optimization objectives, namely: (1) maximizing safety of displaced families and (2) minimizing total public expenditures (TPEs) on temporary housing. First, the safety of displaced families is maximized by minimizing the vulnerability of temporary housing to potential postdisaster hazards, which includes aftershocks at the earthquake fault area and hazardous material releases at the locations of five toxic chemical plants, as shown in Fig. 1. To this end, a safety index ($\mathrm{SI}_j$) is computed for each temporary housing alternative $j$, which accounts for (1) potential hazards; (2) temporary housing type, such as travel trailers, mobile homes, and tent camps; (3) distance between each temporary housing alternative and the potential hazards; (4) expected level of damage in each temporary housing in case the potential hazard occurs; and (5) the probability of each potential hazard to occur given that the natural disaster has occurred. In order to maximize the safety of the displaced families, they should be assigned to temporary housing facilities with the highest $\mathrm{SI}_j$, which can range from 0 (i.e., unsafe) to 1 (i.e., completely safe). Second, TPEs on temporary housing is minimized by computing the monthly ownership or lease cost ($C_j$) of each temporary housing alternative $j$ and assigning families to the least expensive alternatives.

The following sections describe (1) the model formulation; (2) the model implementation using multiobjective GAs (MOGA); (3) the model implementation using weighted integer programming; and (4) the comparison between the optimization results of each optimization technique, as shown in Fig. 2.

### *Model Formulation*

The multiobjective optimization model is formulated in three main steps: (1) designing decision variables; (2) formulating optimization objective; and (3) modeling the optimization constraints, as shown in Fig. 2.

First, the decision variables are designed to represent the possible configurations of temporary housing arrangements. To this end, each decision variable $x_j$ represents the number of displaced families assigned to temporary housing alternative $j$. Accordingly, the number of decision variables is initially equal to the number of available temporary housing alternatives ($J$), and in the present case study there are 222 decision variables.

Second, the two main optimization objectives are formulated using Eqs. (1) and (2)

$$\mathrm{SI} = \frac{1}{n} \times \sum_{j=1}^{J} (\mathrm{SI}_j \times x_j) \tag{1}$$

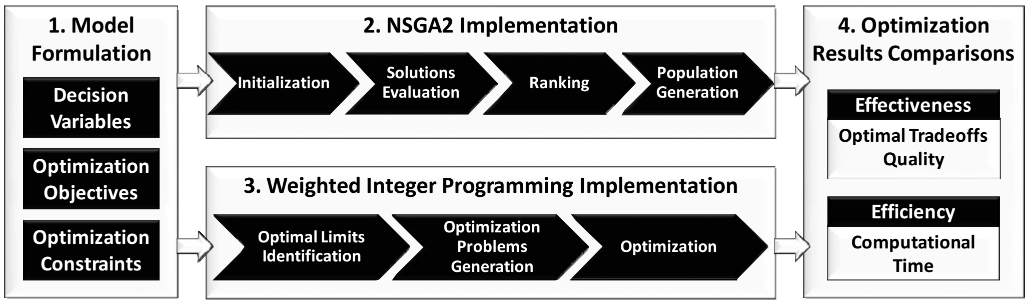

**Fig. 2.** Temporary housing planning case study

$$\text{TPE} = \sum_{j=1}^{J} (C_j \times x_j) \qquad (2)$$

where SI=average safety index per displaced family for a specific configuration of temporary housing arrangements; $n$=number of displaced families in need of temporary housing; $J$=number of temporary housing alternatives; $\text{SI}_j$=safety index for temporary housing alternative $j$; $x_j$=number of displaced families assigned to temporary housing alternative $j$; TPE=TPE on temporary housing for a specific configuration of temporary housing arrangements; and $C_j$=monthly cost of each temporary housing alternative $j$ per house family.

Third, three optimization constraints are formulated to ensure the practicality of generated optimal solutions. These constraints are designed to ensure that (1) the capacity of any temporary housing alternative should not be exceeded; (2) all families should be housed; and (3) number of families assigned to any temporary housing should be positive integers.

## *Model Implementation Using Multiobjective Genetic Algorithms*

Nondominated Sorting GA II (NSGA2) is used to optimize this large-scale optimization problem, because of its important characteristics such as fast nondominated sorting, crowding, elitism, and constrained-domination principle; in addition to its superior performance compared to other MOGAs (D'Souza and Simpson 2002; Deb et al. 2001; Deb 2005; Weile et al. 1996). The model is implemented in four main phases: (1) initialization; (2) solutions evaluation; (3) ranking; and (4) population generation, as shown in Fig. 2.

First, the initialization phase is designed to generate a population of random solutions (chromosomes), where each population in this case study is specified to consist of 2,000 solutions. Each solution consists of a number of real-coded genes, where each gene represents one of decision variables $(x_j)$. Therefore, in the present case study each solution consists of 222 genes. In addition, in order to fulfill the first optimization constraint the value of each gene $(x_j)$ is set to range between 0 and the capacity $(\text{cap}_j)$ of temporary housing alternative j, which ensures that the number of families assigned to any temporary housing will not exceed its capacity.

Second, the solution evaluation phase is designed to evaluate each generated solution based on its performance in (1) the first objective function, which represents maximizing safety for displaced families and is computed using Eq. (1); (2) the second objective function, which represents minimizing TPEs on temporary housing and is computed using Eq. (2); and (3) a constraint compliance function ensuring that the sum of the number of families assigned to all temporary housing alternatives is equal to the number of displaced families in order to fulfill the second optimization constraint. It should be noted that any solution not fulfilling that constraint is considered infeasible.

Third, the ranking phase is used to rank all the solutions based on the constrained-domination principle. In this approach, the evaluated solutions are ranked based on their domination of other solutions, where solution $s_1$ is said to dominate solution $s_2$ if any of the following conditions is true: (1) solution $s_1$ is feasible and solution $s_2$ is infeasible; (2) both solutions are infeasible, but solution $s_1$ has a smaller constraint violation; or (3) both solutions are feasible, but solution $s_1$ outperforms solution $s_2$ in one of the objective functions and its performance in the other objective function is at least as good as solution $s_2$ (Deb et al. 2001).

Fourth, the population generation phase is designed to generate a new improved generation using selection, crossover, and mutation. In this phase, the values of the off-springs genes are forced to be integers in order to fulfill the third optimization constraint. Phases (2)–(4) are then repeated until a predetermined number of generations are completed. In the present case study, the model generated 300,000 generations. Each solution in the first rank in last generation is a nondominated optimal solution providing an optimal tradeoff between maximizing safety and minimizing expenditures.

## *Model Implementation Using Weighted Integer Programming*

The linear nature of optimization objectives and constraints in this case study enables the utilization of analytical optimization techniques, such as integer programming. Weighted integer programming is used in order to simultaneously optimize the two main objectives, where each of the optimization objectives is first normalized (Marler and Arora 2004) and then their weighted sum is maximized, as shown in Eq. (3). Moreover, the three optimization constraints are fulfilled using Eqs. (4)–(6)

$$\text{maximize}\ W_{\text{SI}} \times \frac{\text{SI}_{\max} - \text{SI}}{\text{SI}_{\max} - \text{SI}_{\min}} - W_{\text{TPE}} \times \frac{\text{TPE} - \text{TPE}_{\min}}{\text{TPE}_{\max} - \text{TPE}_{\min}} \qquad (3)$$

subject to Constraint 1: capacities of temporary housing alternatives should not be exceeded

$$x_j \leqslant \text{cap}_j, \quad \forall\, j = 1,2 \ldots J \qquad (4)$$

Constraint 2: all displaced families should be housed

$$\sum_{j=1}^{J} x_j = n, \quad \forall\, j = 1,2 \ldots J \qquad (5)$$

Constraint 3: All assignments should have positive integer values

$$x_j \geqslant 0, \quad x_j \text{ is an integer}, \quad \forall\, j = 1,2 \ldots J \qquad (6)$$

where $W_{\text{SI}}$=relative weight of maximizing safety of displaced families; $\text{SI}_{\max}$ and $\text{SI}_{\min}$=maximum and minimum values of average safety index per displaced family among the optimal solutions, respectively; SI=average safety index per displaced family for a specific configuration of temporary housing arrangements, which can be computed using Eq. (1); $W_{\text{TPE}}$=relative weight of minimizing TPEs on temporary housing; $\text{TPE}_{\max}$ and $\text{TPE}_{\min}$ =maximum and minimum values of TPEs among the optimal solutions, respectively; TPE=TPE on temporary housing for a specific configuration of temporary housing arrangements; $x_j$=number of displaced families assigned to temporary housing alternative $j$; $\text{cap}_j$=capacity of temporary housing alternative $j$ in number of families; $J$=number of temporary housing alternatives; and $n$=number of displaced families in need of temporary housing.

In order to generate the optimal tradeoffs between maximizing safety and minimizing expenditures, the weighted integer programming model is implemented in three main phases: (1) optimal limits identification; (2) optimization problems generation; and (3) optimization, as shown in Fig. 2.

First, the optimal limits identification phase is designed to identify $\text{SI}_{\max}$, $\text{SI}_{\min}$, $\text{TPE}_{\max}$, and $\text{TPE}_{\min}$, which is necessary for

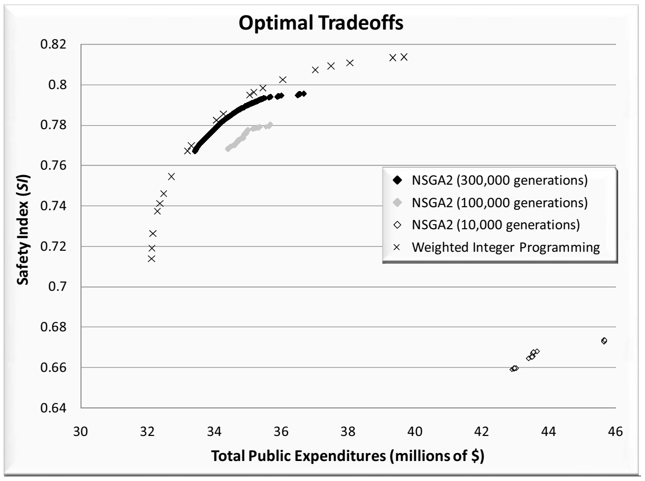

**Fig. 3.** Optimization results

normalizing the optimization objectives as shown in Eq. (3). $\mathrm{SI}_{\max}$ is identified by maximizing Eq. (1) using integer programming and the corresponding value of TPEs is stored as $\mathrm{TPE}_{\max}$. Similarly, $\mathrm{TPE}_{\min}$ is identified by minimizing Eq. (2) using integer programming and the corresponding value of average safety index is stored as $\mathrm{SI}_{\min}$.

Second, the optimization problems generation phase is designed to generate unique combinations of relative weights ($W_{\mathrm{SI}}$ and $W_{\mathrm{TPE}}$) for the two optimization objectives, where the sum of those weights is always equal to 1 (i.e., $W_{\mathrm{SI}}$ and $W_{\mathrm{TPE}}=1$). Each unique combination represents a unique integer programming optimization problem, which is sent to the optimization phase to generate a unique optimal solution. The number of unique combinations, and accordingly the number of unique optimal solutions, is equal to $k+1$, where $k$ is equal to 100 divided by the value of the weight increment, which is set by the decision-maker. In the present case study a weight increment of 5% is used, and therefore the optimization problems generation phase generates 21 unique combinations of relative weights.

Third, the optimization phase is designed to generate a unique optimal tradeoff between maximizing safety and minimizing expenditures for each unique combination of relative weights by maximizing Eq. (3) using weighted integer programming. In the present case study the optimization phase generates 21 unique optimal solutions each representing a unique tradeoff between the two main optimization objectives.

### *Comparison of Analyzed Optimization Techniques*

This section compares the results obtained from the MOGAs using NSGA2 and the weighted integer programming. The results are compared based on two criteria: (1) effectiveness that evaluates the quality of the generated optimal tradeoffs; and (2) efficiency that measures the required computational time, as shown in Fig. 3.

First, the effectiveness of the two techniques can be compared using Fig. 3 that shows the optimal tradeoffs identified by NSGA2 after 10,000, 100,000, and 300,000 generations using a populations size of 2,000; and those identified by weighted integer programming using a 5% increment. As shown in Fig. 3, the optimization results of using NSGA2 illustrate the following: (1) NSGA2 could reach near optimal solutions after 300,000 generations; (2) NSGA2 did not converge yet and has the potential to identify better results with more generations, but the rate of improvement in the quality of solutions is decreasing; and (3) all the generated solutions are concentrated at the middle portion of the Pareto front. On the other hand, the optimization results of using weighted integer programming illustrate the following: (1) it could identify optimal solutions; and (2) the solutions are distributed along the Pareto front. These results illustrate that weighted integer programming outperforms NSGA2 in terms of the quality of generated optimal tradeoffs between maximizing safety and minimizing expenditures for the temporary housing planning problem.

Second, the efficiency of the two techniques can be compared using the total computation time for optimizing this large-scale temporary housing problem on a 2.6 GHz Intel Pentium 4 processor with 512 MB of RAM. The total computation time required by the NSGA2 to complete 300,000 generations was approximately 90 h; while that required by the weighted integer programming was less than one minute. These results clearly illustrate that the computational efficiency of the weighted integer programming model outperform those of NSGA2 for this particular large-scale temporary housing planning problem. It should be noted, however, that the efficiency of NSGA2 can be significantly improved using parallel computing paradigms, especially for large-scale construction optimization problems that cannot be modeled using analytic optimization algorithms. The following case study analyzes one of these problems and investigates the potential of parallel computing to enhance the computation efficiency.

## Second Case Study: Construction Resource Optimization

This case study is analyzed to investigate the robustness of naturally inspired MOAs and parallel computing paradigms in optimizing resource utilization in large-scale infrastructure projects. The optimization is performed using a MOGA called the Non-dominated Sorting GA II (NSGA2). Despite its advantages and flexibility in enabling multiobjective optimization for linear and nonlinear problems, NSGA2 often requires extensive computation time to solve large-scale optimization problems. To investigate and enhance this critical robustness issue, this case study focuses on: (1) model formulation and implementation using NSGA2; (2) the utilization of parallel computing paradigms to enhance the robustness of the optimization model; and (3) analyzing the efficiency and effectiveness of the implemented NSGA2 model and parallel computing paradigms.

### *Model Formulation and Implementation*

An optimization model is formulated to support construction engineers and managers in identifying optimal resource utilization plans that provide optimal tradeoffs between construction time and cost. Accordingly, the main decision variables considered in this model for each construction activity in the project include: (1) construction method, which represents the different types of materials and/or methods that can be used in the construction of the activity; (2) crew formation, which models available sizes and configurations for construction crews; and (3) crew overtime policy, which represents available overtime hours and nighttime shifts. These three main decision variables are then combined into one called resource utilization option (n) in order to reduce the complexity of the optimization problem. The main goal of the optimization algorithm employed to this problem is then to select

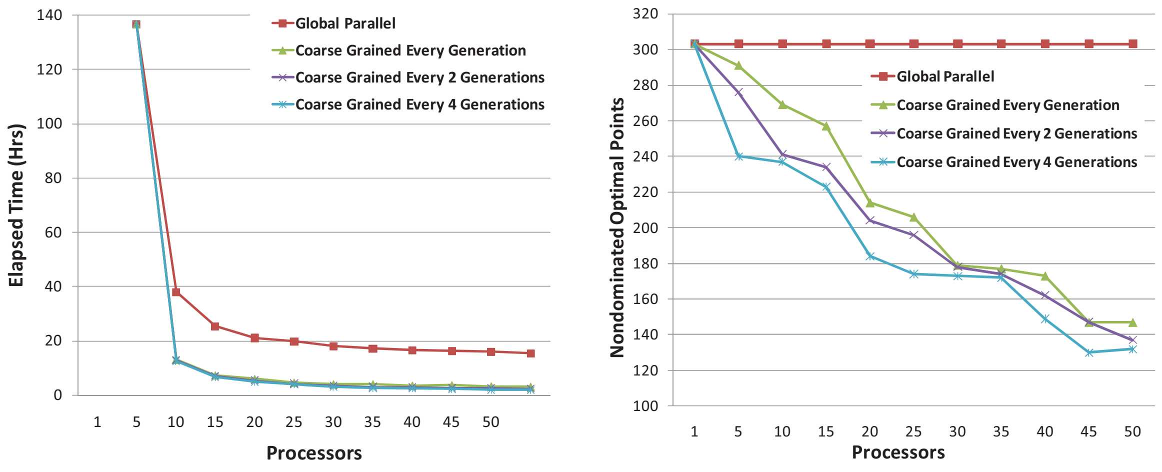

**Fig. 4.** Efficiency and effectiveness of parallel computing approaches

the optimal resource utilization option for each activity in the project. Since the employed naturally inspired MOA is a MOGA, these resource utilization are modeled in a GA string with each location of that string representing a resource utilization option of one of the project activities.

The fitness of the formulated GA strings needs to be evaluated in order for NSGA2 stages to be implemented. The two main objectives considered in the present case study were project cost and time, as shown in Eqs. (7) and (8)

$$\text{Minimize Project Cost} = \sum_{i=1}^{t} [(M_i^n + D_i^n \times R_i^n) + (B_i^n)] \quad (7)$$

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 dollars per day of resource utilization ($n$) in activity ($i$); $B_i^n$=subcontractor lump sum cost for resource utilization ($n$) in activity ($i$), if any

$$\text{Minimize Project Time} = \sum_{i=1}^{t} T_i^n \quad (8)$$

where $T_i^n$=duration of activity ($i$) on the critical path using resource utilization ($n$). In order to determine the critical path in project activity networks, the critical path method (CPM) is employed to calculate activity times and floats. The utilization of the CPM requires the implementation of graph traversal algorithm which makes the representation of this objective in closed mathematical form infeasible which precludes the utilization of analytic optimization algorithms. Accordingly, this formulated model was implemented as an NSGA2 model using a similar procedure to that described earlier in the first case study.

## *Parallel Computing to Enhance Robustness*

To examine the efficiency of the developed MOGA, a preliminary analysis was performed to measure the needed GA processing time. In this analysis, the MOGA was used to simultaneously minimize the duration and cost for a large-scale project that includes 720 activities. The resource utilization for this project was optimized using a single computer/processor in order to identify, for each activity in the project, an optimal selection of construction method, crew formation and/or overtime policy. The results of this preliminary analysis indicated that the GA computations on a single processor/computer required approximately 137 h for optimizing this preliminary analysis. The findings of this analysis confirm the need for methods to enhance the efficiency of the developed MOGA. Accordingly, two parallel programming paradigms are implemented and tested to analyze their impact on improving the efficiency of the developed MOGA. The two parallel computing paradigms that were analyzed in this case study are global parallelization and coarse-grained parallelization.

The global parallel GA uses the global paradigm of parallel computing for increasing the efficiency of the MOGA (De Santiago and Law 2000). In this approach, one processor is assigned the role of a manager processor that manages the computations on the remaining worker processors. The manager processor starts by implementing the initialization task of the MOGA, and then breaks down the generated population of solutions into a number of subpopulations and equally distributes them to all the worker processors in the used computing cluster. The worker processors are then used to evaluate the fitness (i.e., project duration and cost) of all solutions in their subpopulations and return their evaluated solutions to the manager processor. The manager processor then performs the generation evolution task of the multiobjective optimization module in order to create the next generation of solutions. The generation evolution and fitness evaluation tasks are repeated iteratively using the manager and worker processors until the convergence criteria are met. The implementation of this parallel GA for a construction project size of 720 activities was able to reduce the overall computational time from 136.50 h on a single processor to 19.72 h using a cluster of 20 processors, as shown in Fig. 4. Despite this significant time saving, the performance of the global parallel computing paradigm used in this module was reported in the literature to suffer from efficiency losses especially as the number of processors increases (De Santiago and Law 2000), which was the main motivation to examine the second paradigm of coarse-grained parallel computing.

The course-grained parallel GA on the other hand, employs data parallelism by dividing the evaluated population of solutions into a number of subpopulations called demes that are evolved independently on a number of parallel processors. Accordingly, all processors involved in the computations perform all the tasks of the MOGA in a completely nonhierarchical fashion, which eliminates the need for a single processor to manage and distribute the computations (Cantú-Paz 1997a; Cantú-Paz 2000; Noda et al. 2002; Adeli 2000). The nonhierarchical structure of the module also leads to the elimination of synchronized communications, which also allows for further time savings and speedups (Cantú-Paz 2000). The asynchronous communications implemented in this framework also prevent slower processors in the computation from impeding the progress of the remaining faster processors in the cluster. In order to compensate for the independent evolution of the subpopulation (i.e., demes) in this paradigm, the present algorithm enables each processor to share and exchange the best found solutions in its deme with the remaining processors in the cluster using a procedure called the migration process (Cantú-Paz 1997b).

### Comparison of Parallel GA Paradigms

The performance of the aforementioned parallel GA paradigms was evaluated in this case study using 44 experiments that tested various combinations of: (1) eleven different clusters of parallel processors that ranged from 1 to 50 with an increment of 5 processors, as shown in Fig. 4; (2) two migration rates in the coarse-grained parallel module that specify the transfer of the top 25% and 75% of the solutions in each deme; and (3) three migration intervals that require the execution of the migration procedure every one, two, and four generations, as shown in Fig. 4. The main metric used to judge the efficiency of the two parallel computing approaches is the Elapsed time of the algorithm, while the main metric for judging effectiveness of the approach was the number of nondominated points obtained. The used number of parallel processors as well as the specified migration rate and interval affected both the GA efficiency and effectiveness (Noda et al. 2002). First, increasing the number of parallel processors led to: (1) an increase in the GA efficiency due to the decrease in the overall computational time as shown in Fig. 4; and (2) a decrease in the GA effectiveness due to the reduced size of evaluated subpopulations/demes. Second, increasing the migration rate and interval produced: (1) reduced GA efficiency due to the increase in migration and communication times among processors; and (2) improved GA effectiveness due the increase in the exchange rate and interval of best found solutions among analyzed demes leading to improvements in the explorative and exploitive power of the algorithm (Cantú-Paz 2000; Noda et al. 2002).

Additionally, the above results show that the coarse-grained parallel approach did not lose effectiveness as its efficiency increased, while the coarse-grained approach showed a tradeoff between its efficiency and effectiveness. The coarse-grained approach, however, was able achieve higher increases in efficiency as the number of processors used increased. These results show the feasibility of parallel computing paradigms and their potential to enhance the robustness of the developed MOGA.

## Recommendations to Enhance Robustness and Research Rigor

The above explained case studies showed that the performance of MOAs and parallel computing approaches varied depending on the type and size of the construction optimization problem addressed. Therefore, the main findings of the above two case studies are used to develop a number of recommendations to enhance the robustness and research rigor in this critical area of construction research. These recommendations focus on enhancing: (1) the formulation of the optimization problem; (2) the selection of optimization algorithm; and (3) the testing and validation of optimization algorithms.

### Formulation of Optimization Problem

The formulation of construction multiobjective optimization problems is one of the important stages that has a significant impact on the effectiveness and efficiency of the optimization process. Accordingly, the following recommendations can be used to enhance robustness whenever they are feasible and can be applied.

1. Minimize the number of decision variables, if possible. Decreasing the number of decision of variables by possibly combining them (similar to the decision variables combined in the second case study of this paper) decreases the complexity of the optimization and allows for it to be solved more efficiently.
2. Minimize or relax constraints if possible. The reduction of optimization constraints has the potential of creating a non-constrained optimization problem, which is particularly useful in naturally inspired optimization algorithms because many of those algorithms do not have fully mature mechanisms for constraint handling.
3. Favoring objective functions that could be modeled in closed mathematical form. This allows the employment of analytical algorithms. The use of objective functions that are modeled through a computer algorithm or simulation may preclude the use of analytical optimization algorithms.

### Selection of the Optimization Algorithm

The selection of the optimization algorithm should be based on the type and nature of the problem. Based on the conducted analysis of the two case studies, the following recommendations can be applied to enhance the robustness of the optimization process.

1. The first case study illustrated that the optimization effectiveness and efficiency of analytic optimization algorithms outperform those of the analyzed naturally inspired one. Therefore, the use of analytic optimization algorithms can be preferred, whenever possible, over naturally inspired algorithms.
2. The second case study illustrated that many construction optimization problems may not be good candidates for analytic optimization algorithms due to the mathematical formulation of its objectives. Accordingly, these problems can best be optimized using naturally inspired optimization algorithms in combination with parallel computing paradigms to enhance the computation time and optimization efficiency, especially in large-scale construction optimization problems.
3. Optimization algorithms require extensive testing and validation to ensure that they function properly. Many optimization algorithms are currently present in proprietary or open-source libraries that facilitate the development of optimization models. Therefore, researchers should favor those implementations, available from trusted sources, over developing their own code if possible. If not, researchers should use well tested code as a benchmark for their developed implementations.

### Testing and Validation

The testing and validation of optimization models is a challenging task. The main challenge in many construction optimization problems is that the true optimal solutions are not really known, and if know from other research studies, that solution may not be a true optimum. In order to overcome this challenge, the following recommendations can be applied whenever possible.

1. The validation could be performed by analyzing the optimization problem using multiple optimization algorithms to evaluate the relative performance of the algorithms giving preference to the results of algorithms with better performance on standard problems of equivalent complexity.
2. If an analytical optimization algorithm requiring an initial starting point, the algorithm should executed with multiple starting points to ensure convergence to a true optimal solution.
3. If a small-scale problem exists such that all solutions could be enumerated to find its optimal solution, that problem should be used initially for testing the used optimization algorithm before testing the algorithm on large-sized problems.

## Conclusions

This paper presented a review of current research efforts in the field of construction MOAs and a detailed analysis of two large-scale case studies that were designed to evaluate the robustness of naturally inspired and analytical MOAs. The first case study aimed at optimizing the planning of postdisaster temporary housing. This case study first used a MOGA, however the processing time was found to be very large. The case study then used an analytical optimization algorithm that was found to outperform the MOGA in both effectiveness and efficiency. The second case study aimed at optimizing construction resource utilization in large-scale infrastructure projects. In this case study, a MOGA was used for performing the optimization process. In order to increase the robustness of this algorithm, two parallel computing approaches were used. The results of these approaches illustrated that significant efficiency increases could be achieved, and that in some cases, tradeoffs exist between efficiency and effectiveness. The paper also presented a number of recommendations enhance the robustness of MOAs and the level of rigor in the research performed in this field. The recommendations focused on: (1) problem formulation; (2) selection of the optimization algorithm; and (3) testing and validation methods. These recommendations are expected to enhance the development and utilization of multiobjective construction optimization models, however further research is needed to analyze and address other causes that still limit the widespread utilization of optimization techniques in the construction industry.

## Acknowledgments

This material is based upon work supported by the Qatar National Research Fund under Award No. QNRF-NPRP26-6-7-2. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the writers and do not necessarily reflect the views of the Qatar National Research Fund.

## References

Adeli, H. (2000). "High performance computing for large-scale analysis, optimization, and control." *J. Aerosp. Eng.*, 13(1), 1–10.

Afshar, A., Kaveh, A., and Shoghli, O. (2007). "Multi-objective optimization of time-cost-quality using mutli-colony ant algorithm." *Asian J. Civil Engineering (Building and Housing)*, 8(2), 113–124.

Alonso, J., et al. (2000). "Parallel computing in water network analysis and leakage minimization." *J. Water Resour. Plann. Manage.*, 126(4), 251–260.

Balla, M., and Lingireddy, S. (2000). "Distributed genetic algorithm model on network of personal computers." *J. Comput. Civ. Eng.*, 14(3), 199–205.

Burns, S. A., Liu, L., and Feng, C. (1996). "The LP/IP hybrid method for construction time-cost trade-off analysis." *Constr. Manage. Econom.*, 14, 265–276.

Cantú-Paz, E. (1997a). "A survey of parallel genetic algorithms." *ILLGAL Rep. No. 97003*, Illinois Genetic Algorithms Laboratory, Univ. of Illinois at Urbana Champaign, Urbana, Ill.

Cantú-Paz, E. (1997b). "Designing efficient master-slave parallel genetic algorithms." *ILLGAL Rep. No. 97004*, Illinois Genetic Algorithms Laboratory, Univ. of Illinois at Urbana Champaign, Urbana, Ill.

Cantú-Paz, E. (2000). *Efficient and accurate parallel genetic algorithms*, Kluwer Academic, Boston.

Carty, G. (1995). "Construction." *J. Constr. Engrg. Manage.*, 121(3), 319–328.

Chan, W., and Hu, H. (2002). "Production scheduling for precast plants using a flow shop sequencing model." *J. Comput. Civ. Eng.*, 16(3), 165–174.

Cheng, T., and Chen, Y. (2007). "A GA mechanism for optimizing the design of attribute double sampling plan." *Autom. Construct.*, 16, 345–353.

D'Souza, B., and Simpson, T. (2002). "A genetic algorithm based method for product family design optimization." *Proc., DETC 2002: 2002 ASME Design Engineering Technical Conf.*, ASME, New York.

De Santiago, E., and Law, K. H. (2000). "A distributed implementation of an adaptive finite-element method for fluid problems." *Comput. Struc.*, 74(1), 97–119.

Deb, K. (2005). "Real-coded genetic algorithms." *Kanpur Genetic Algorithm Laboratory, Indian Institute of Technology, Kanpur, India*, ⟨http://www.iitk.ac.in/kangal/resources.shtml⟩ (Nov. 12, 2005).

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2001). "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization." *KANGAL Rep. No. 200001*, Genetic Algorithm Laboratory, Indian Institute of Technology, Kanpur, India.

Dorigo, M., and Maniezzo, V. (1996). "ColorniA. Ant system: Optimization by a colony of cooperating agents." *IEEE Trans. Syst. Man Cybern.*, 26(1), 29–41.

El-Anwar, O., and El-Rayes, K. (2007). "Post-disaster optimization of temporary housing efforts." *Proc., ASCE Construction Research Congress*, ASCE, Reston, Va.

El-Rayes, K. (2001). "Optimum planning of highway construction under the A+B bidding method." *J. Constr. Eng. Manage.*, 127(4), 261–269.

El-Rayes, K. and Hyari, K. (2005). "Optimal lighting arrangements for nighttime highway construction projects." *J. Constr. Eng. Manage.*, 131(12), 1292–1300.

El-Rayes, K. and Kandil, A. (2005). "Time-cost-quality trade off analysis for highway construction." *J. Constr. Eng. and Manage.*, 131(4), 477–486.

El-Rayes, K. and Khalafallah, A. (2005). "Tradeoff between safety and cost in planning construction site layouts." *J. Constr. Eng. Manage.*, 131(11), 1186–1195.

El-Rayes, K., and Moselhi, O. (2001). "Optimizing resource utilization for repetitive construction projects." *J. Constr. Eng. Manage.*, 127(1), 18–27.

Elbeltagi, E., Hegazy, T., and Grierson, D. (2005). "Comparison among five evolutionary-based optimization algorithms." *Adv. Eng. Inf.*, 19(1), 43–53.

Feng, C., Liu, L., and Burns, S. (1997). "Using genetic algorithms to solve construction time-cost trade-off problems." *J. Comput. Civ. Eng.*, 11(3), 184–189.

Feng, C., Liu, L., and Burns, S. (2000). "Stochastic construction time-cost trade-off analysis." *J. Comput. Civ. Eng.*, 14(2), 117–126.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Reading, Mass.

Gomar, J., Haas, C. and Morton, D. (2002). "Assignment and allocation optimization of partially multiskilled workforce." *J. Constr. Eng. Manage.*, 128(2), 103–109.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, Mich.

Hyari, K. and El-Rayes, K. (2006). "Optimal planning and scheduling for repetitive construction projects." *J. Manage. Eng.*, 22(1), 11–19.

Ipsilandis, P. (2007). "Multiobjective linear programming model for scheduling linear repetitive projects." *J. Constr. Eng. Manage.*, 133(6), 417–424.

Kandil, A. and El-Rayes, K. (2006a). "MACROS: Multi-objective automated construction resource optimization system." *J. Manage. Eng.*, 22(3), 126–134.

Kandil, A. and El-Rayes, K. (2006b). "Parallel genetic algorithms for optimizing resource utilization in large-scale construction projects." *J. Constr. Eng. Manage.*, 132(5), 491–498.

Kasaeian, A., Shoghli, O., and Afshar, A. (2007). "Nondominated archiving genetic algorithm for multi-objective optimization of time-cost trade-off." *Proc., 8th WSEAS Int. Conf. on Evolutionary Computing*, WSEAS.

Kennedy, J., and Eberhart, R. (1995). "Particle swarm optimization." *Proc., IEEE Int. Conf. on Neural Networks*, IEEE, Piscataway, N.J., 1942–1948.

Khalafallah, A. and El-Rayes, K. (2006). "Optimizing airport construction site layouts to minimize wildlife hazards." *J. Manage. Eng.*, 22(4), 176–185.

Lee, C., and Kim, S. (2007). "GA-based algorithm for selecting optimal repair and rehabilitation methods for reinforced concrete (RC) bridge decks." *Autom. Constr.*, 16, 153–164.

Levitt, R. (2007). "CEM research for the next 50 years: Maximizing economic, environmental, and societal value of the built environment." *J. Constr. Eng. Manage.*, 133(9), 619–628.

Liu, C., Hammad, A., and Itoh, Y. (1997). "Multiobjective optimization of bridge deck rehabilitation using a genetic algorithm." *Microcomput. Civ. Eng.*, 12, 431–443.

Liu, M., and Frangopol, D. (2005). "Bridge annual maintenance prioritization under uncertainty by multiobjective combinatorial optimization." *Comput. Aided Civ. Infrastruct. Eng.*, 20, 343–353.

Marler, R. T., and Arora, J. S. (2004). "Survey of multi-objective optimization methods for engineering." *Struct. Multidiscip. Optim.*, 26(6), 369–395.

Mattila, K., and Abraham, D. (1998) "Resource leveling of linear schedules using integer linear programming." *J. Constr. Eng. Manage.*, 124(3), 232–244.

Moselhi, O., and El-Rayes, K. (1993). "Scheduling of repetitive projects with cost optimization." *J. Constr. Eng. Manage.*, 119(4), 681–697.

Noda, E., Coelho, A., Ricarte, I., Yamakami, A., and Freitas, A. (2002). "Devising adaptive migration policies for cooperative distributed genetic algorithms." *Proc., 2002 IEEE Int. Conf. on Systems, Man and Cybernetics*, IEEE, Piscataway, N.J.

Schaumann, E. J., Balling, R., and Day, K. (1998). "Genetic algorithms with multiple objectives." *American Institute of Aeronautics and Astronautics, Inc., Rep. No. A98-39919*, AIAA, Reston, Va.

Thiagarajan, G., and Aravamuthan, V. (2002). "Parallel strategies for element-by-element preconditioned conjugate gradient solver using high performance Fortran for unstructured finite element applications on Linux clusters." *J. Comp. Civ. Eng.*, 16(1), 1–10.

U.S. Bureau of Economic Analysis (BEA). (2007). *Annual industry reports*, ⟨http://www.bea.gov/industry/gpotables/gpo_action.cfm?anon=54811&table_id=19025&format_type=0⟩ (June 1, 2007).

Weile, D. S., Michielssen, E., and Goldberg, D. E. (1996). "Genetic algorithm design of pareto-optimal broadband microwave absorbers." *IEEE Trans. Electromagn. Compat.*, 38(3), 518–525.

Xiong, Y., and Kuang, Y. (2008). "Applying an ant colony optimization algorithm-based multiobjective approach for time-cost trade-off." *J. Constr. Eng. Manage.*, 134(2), 153–156.

Yang, I. (2007). "Using elitist particle swarm optimization to facilitate bicriterion time-cost trade-off analysis." *J. Constr. Eng. Manage.*, 133(7), 498–505.

Zheng, D., and Ng, S. (2005). "Stochastic time-cost optimization model incorporating fuzzy sets theory and nonreplaceable front." *J. Constr. Eng. Manage.*, 131(2), 176–186.

Zheng, D., Ng, T., and Kumaraswamy, M. (2004). "Applying a genetic algorithm-based multiobjective approach for time-cost optimization." *J. Constr. Eng. Manage.*, 130(2), 168–176.