A MODIFIED SHUFFLED-FROG-LEAPING ALGORITHM FOR OPTIMIZING BRIDGE-DECK REPAIRS

E. Elbeltagi
Associate Professor, Structural Engineering Department, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt
Email: eelbelta@mans.edu.eg

ABSTRACT

Infrastructure assessment and renewal have been in the center of attention worldwide. While the infrastructure is the foundation for economic growth, a large percentage of existing facilities are rapidly deteriorating due to age, outdated technology, and insufficient capacity for population growth. Prioritizing infrastructure facilities for repair purposes, the allocation of the limited funds, and the selection of appropriate repair methods, however, is a complex task, particularly when hundreds of facilities are involved. In this case, the complexity and huge problem size render traditional mathematical optimization tools as inadequate.

Recently, a new breed of evolutionary algorithms has evolved with non-traditional optimization tools with great potential for application in the infrastructure management domain. Evolutionary algorithms, such as shuffled frog leaping, are stochastic search methods that mimic natural biological evolution and/or the social behavior of species. Such algorithms have been developed to arrive at optimum or near-optimum solutions to complex and large-scale optimization problems which cannot be solved by traditional mathematical optimization techniques.

As a new and promising technique, the shuffled frog leaping algorithm draws its formulation from two other search techniques: the local search of the “particle swarm optimization” technique; and the competitiveness mixing of information of the “shuffled complex evolution” technique. In this paper, a modified shuffled frog leaping algorithms is presented and applied on a network of bridges to optimize the repair decisions for the bridge decks. The objective function is formulated to minimize the life-cycle cost of bridge deck repairs. The system’s implementation on a spreadsheet program is briefly highlighted. Also, the results of using the modified shuffled frog leaping are compared with that of the genetic algorithms.

Keywords: Bridge Decks, Optimization, Life Cycle Cost, Shuffled Frog Leaping, Evolutionary Algorithms.

INTRODUCTION

Optimization and prioritization of infrastructure maintenance works for repair have always posed a computational challenge due to the complexity and scale of the problem. This challenge is a result of the exponential growth of solution space, often resulting in an unmanageable process using traditional mathematical optimization techniques. Such mathematical optimization techniques tend to be complex and computationally intensive.
Infrastructure maintenance works include: bridges, roads, sewer systems, water supply networks and many other works. This paper focuses on the maintenance of a network of bridge decks. Bridges are very important links in road infrastructure networks and it is important to keep them well maintained. Lack of preserving bridges resulted in increasing repair costs to a point where repairing deteriorated bridges is often more expensive than building new ones (Miyamoto et al. 2001). The increase in deteriorated bridges and maintenance costs, in view of the limited funds, has motivated many agencies to develop Bridge Management Systems such as Pontis (FHWA 2001) and Bridgit (Hugh 1995) to support decision makers in allocating their limited funds to top-priority bridges.

While the selection of bridges for prioritization is considered a network-level decision, selection of repair type is a project-level decision. Dealing with network and project levels separately may lead to a non-optimal decision. Both network-level and project-level decisions are inter-related and should be integrated together (Thompson et al. 2003). Considering both levels is a challenging task that leads to a complex optimization problem. For example, if the number of bridges is N, the planning horizon is T years, and the number of repair alternatives is R, then the number of possible combinations is $R^{NT}$. Therefore, the complexity of the problem grows exponentially with the increase in numbers of bridges, making traditional optimization techniques not able to handle such huge problem. This is the motivation for recent research efforts for developing new and innovative techniques using evolutionary-based approaches for searching optimum or near-optimum solutions.

Evolutionary Algorithms are stochastic search methods that mimic natural biological evolution and/or social behavior of species. Examples include how ants find the shortest route to a source of food and how birds find their destination during migration (Løvbjerg 2002). The first evolutionary-based technique introduced in the literature was the Genetic Algorithms (Holland 1975). Genetic Algorithms (GAs) were developed based on the Darwinian principle of “survival of the fittest” and the natural process of evolution through reproduction. Despite their benefits, GAs may require long processing time for a near-optimum solution to evolve. Also, not all problems lend themselves well to a solution with GAs (Joglekar and Tungare 2004).

In an attempt to reduce processing time and improve the quality of solutions, other Evolutionary Algorithms (EAs) have been introduced during the last decade, including various GA improvements and a recently developed technique: shuffled frog-leaping. In general, EAs share a common approach for their application to a given problem. The problem usually requires some representation to suit each method, then, the evolutionary search algorithm is applied iteratively to arrive at optimum or near-optimum solution (Elbeltagi et al. 2005).

This paper presents an approach to optimize the prioritization of infrastructure maintenance works using the modified shuffled frog leaping algorithm. Bridge decks are used as an example of infrastructure maintenance works. The paper starts with describing a simplified framework for bridge deck repair decision considering both network and project levels. A review of the SFL algorithm is then presented. To facilitate the model implementation, a macro program has been written on spreadsheet program. Performance comparison between the GAs and modified SFL is presented in terms of processing time and solution quality for problems with different number of bridges.

**TYPICAL ASSET MANAGEMENT SYSTEM**

The components of a typical asset management system include (Figure 1): time-dependent deterioration model, repair cost model, and improvement model; constraints; and decision support module (Hegazy et al. 2004). In this study, a bridge deck management system (BDMS) is used for applying the modified SFL algorithm. In the BDMS, the condition rating system used was developed by the FHWA (1998), which uses a scale from 0 to 9, where 9 is the best condition; also it assumes that the bridge is non-serviceable when the condition rating reaches 3. Markov chain deterioration
model is used to predict the future bridge deck conditions (Jiang 1990). The transition probability matrix was developed based on the FHWA condition rating. Three repair options were, also, considered for bridge decks: light repair, medium repair, and extensive repair. The repair cost is estimated as a percentage of the initial cost; 28.5%, 65% and 100% respectively (Seo 1990), in addition to the do nothing option. The impact of applying a repair option on improving the bridge deck condition is presented in Hegay et al (2004). For example, to raise the bridge condition from level 3 to level 5, medium repair is required, while to raise it to level 7, extensive repair should be applied. BDMS considers practical constraints imposed on a network of bridges at both the project level and the network-level. It, also, takes into account other constraints such as governmental, political, and user predefined constraints (Hegazy et al. 2004). All the above models are integrated and linked with a life cost cycle (LCC) optimization module to optimally select the repair strategy while considering different practical constraints in order to minimize the Total Life Cycle Cost (TLCC).

As an optimization problem, an objective function is constructed by summing up the present value of the annual cost of repairs for all bridges. The objective function, as such, is to minimize the TLCC, while maintaining acceptable bridge condition.

\[
\text{Min } \ TLCC = \sum_{t=1}^{T} \sum_{i=1}^{N} \frac{C_{ti}}{(1+r)^t}
\]  

(1)

Where, \( C_{ti} \) = repair cost of bridge \( i \) at time \( t \); \( r \) = discount rate; \( T \) = number of years; and \( N \) = number of bridges. This objective function is subjected to the following constraints: 1) yearly \( LCC \) should be \( \leq \) yearly budget limits; 2) condition rating of individual bridges \( \geq 3 \) (minimum acceptable level to FHWA); 3) network overall condition rating is \( \geq \) pre-defined user desirable value; 4) repair method used in a specific year for a specific bridge = user-forced value; and 5) the number of repair visits to

Fig. 1: Components of a Typical Asset Management System
a certain bridge can be constrained to a user-desirable maximum number. The number of variables equal \( N \times T \) and the solution is structured as a string of elements equal to the number of variables as shown in Figure 2. Each element can take an integer value from 0 to 3 corresponding to one of the repair options (0 = do nothing; 1 = light repair; 2 = medium repair; and 3 = extensive repair).

![Fig. 2: Typical Solution Representation](image)

To facilitate the implementation process and the experimentation using the modified SFL algorithm, the proposed bridge deck management system was implemented on a commercial spreadsheet program. In this study, Microsoft Excel software is selected for the implementation of the proposed model because of its ease of use and its powerful programming language. The BDMS was coded using the VBA Macro Language of Excel; various procedures were coded to form a complete BDMS in a user friendly interface. Figure 3 shows the input data for bridge network. More details about the model development and its implementation on spreadsheet program can be found in Hegazy et al. (2004).

![Fig. 3: Main User Interface for the BDMS](image)
MODIFIED SHUFFLED-FROG-LEAPING ALGORITHM

As a new and promising technique, the shuffled frog leaping algorithm draws its formulation from two other search techniques: the local search of the "particle swarm optimization" technique (Kennedy and Eberhart 1995); and the competitiveness mixing of information of the "shuffled complex evolution" technique (Duan and Gupta 1993). The SFL is a heuristic search algorithm. It attempts to balance between a wide scan of a large solution space and also a deep search of promising locations for a global optimum. As such, in the SFL, the population consists of a set of frogs (solutions) each having the same solution structure as shown in Figure 2. A solution to a given problem is represented in the form of a string (Figure 2), called "frog", consisting of a set of elements, which hold a set of values for the optimization variables (in our problem the repair option at each year during the planning horizon for a facility).

The whole population of frogs is then partitioned into subsets referred to as memeplexes. The different memeplexes are considered as different cultures of frogs that are located at different places in the solution space (i.e. global search). Each culture of frogs performs a deep local search. Within each memeplex, the individual frogs hold information, that can be influenced by the information of their frogs within their memeplex, and evolve through a process of change of information among frogs from different memeplexes. After defined number of evolution steps, information is passed among memeplexes in a shuffling process (Eusuff and Lansey, 2003). The local search and the shuffling processes (global relocation) continue until a defined convergence criterion is satisfied (Eusuff and Lansey, 2003).

As explained, the SFL formulation places emphasis on both global and local search strategies, which is one of its major advantages. As shown in Figure 4a, the SFL algorithm starts with an initial population of "P" frogs created randomly. Frog $i$ is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iS})$; where $S$ represents the number of variables. Afterwards, the frogs are sorted in a descending order according to their fitness. Then, the entire population is divided into m memeplexes, each containing n frogs (i.e., $P = m \times n$). In this process, the first frog goes to the first memeplex, the second frog goes to the second memeplex, frog $m$ goes to the $m$ memeplex, and frog $m+1$ goes to the first memeplex, etc.

Within each local memeplex (Figure 4b), the frogs with the best and the worst fitness are identified as $X_b$ and $X_w$, respectively. Also, the frog with the global best fitness (the overall best frog) is identified as $X_g$. Then, an evolutionary process is applied to improve only the frog with the worst fitness (not all frogs) in each cycle. Accordingly, each frog updates its position to catch up with the best frog as follows:

$$\text{Change in frog position } (D_i) = \text{rand()} \cdot (X_b - X_w)$$

(2)

$$\text{New position } X_w = \text{current position } X_w + D_i; \quad D_{\text{max}} \geq D_i \geq -D_{\text{max}}$$

(3)

Where $\text{rand()}$ is a random number between 0 and 1; and $D_{\text{max}}$ is the maximum allowed change in frog's position. If this process produces a better solution, it replaces the worst frog. Otherwise, the calculations in Equations 2 and 3 are repeated with respect to the global best frog (i.e., $X_g$ replaces $X_b$). If no improvement becomes possible in this case, then a new solution is randomly generated to replace the worst frog. The calculations then continue for a specific number of iterations (Eusuff and Lansey, 2003). Accordingly, the main parameters of SFL are: population size $P$; number of memeplexes; number of generations for each memeplex before shuffling; number of shuffling iterations; and maximum step size, $D_{\text{max}}$. The SFL algorithm has been proven to perform better than
other evolutionary algorithms as reported by Elbeltagi et al. (2005). For more discussions about the SFL and its variations can be found in Elbeltagi et al. (2006).

In the SFL algorithm, each memeplex is allowed to evolve independently to locally search at different regions of the solution space. As well, shuffling all the memeplexes and re-dividing them again into a new set of memeplexes results in a global search through changing the information between memeplexes. As such, the SFL algorithm attempts to balance between a wide search of the solution space and a deep search of promising locations that are close to a local optimum. The modified SFL algorithm is formed by multiplying the right-hand side of Equation 2 by what is called search-acceleration factor, \( C \), to form (Elbeltagi et al. 2006):

![Flowchart of the Shuffled-Frog-Leaping Algorithm](image)
Change in frog position \( (D) = \text{rand()} \cdot C \cdot (X_{b} - X_{w}) \) (4)

Assigning a large value to the factor \( C \) at the beginning of the evolution process will accelerate the global search by allowing for a bigger change in the frog’s position and accordingly will widen the global search area. Then, as the evolution process continues and a promising location is identified, the search-acceleration factor, \( C \), will focus the process on a deeper local search as it will allow the frogs to change its positions. The search-acceleration factor, which can be a positive constant value, linear, or nonlinear function of time, provides the means to balance between global and local search (Elbeltagi et al. 2006).

**EXPERIMENTATION WITH THE MODIFIED SFL ALGORITHM**

Using the VBA Macro programming language of Microsoft Excel, the SFL algorithm was coded and integrated with the BDMS described earlier. The modified SFL algorithm was used to optimize the repair decisions for bridge networks. Experiments were carried out using different numbers of bridges: 10, 50, 100, and 250. For comparison purposes with the GAs, the same experiments were repeated using the GAs. Ten trial runs were performed for each case using the GAs and the SFL. All experiments took place on a 1.8 MHz laptop PC machine.

The performance of the two algorithms is compared using four criteria: 1) the percentage of success (i.e., how many trials out of 10 were able to provide a solution without violating the allowable yearly budget and the constraints on the condition of individual bridges and the whole network); 2) the best solution obtained; 3) the average solution \( TLCC \) of all successful trails; and 4) the average processing time to reach the best solution for successful trails.

Using GAs in several initial experiments, GAs parameters were set as follow: crossover probability = 0.8; mutation probability = 0.08; population size is 100; and number of generations = 500. The GAs optimization module was kept running till no improvement was made in the objective function for 10 consecutive generation cycles (10 × 500 generations). Also, in several other initial experiments with the modified SFL, its parameters were set as follows: number of memplexes = 20; number of frogs per memplex = 10; and number of iteration per memplex = 20 (Elbeltagi et al. 2005). The SFL optimization module was kept running till there was no improvement made in the objective function for 10 consecutive shuffling cycles (10 × 20 × 10 generations).

**DISCUSSION OF RESULTS**

The results of using the GAs and the modified SFL algorithms on different sets of bridges (10, 50, 100, and 250) are shown in Figures 8 and 9. For the experiments with large networks of bridges (50, 100, and 250 bridges), the networks were constructed by copying the 10-bridge network several times. As such, the solution obtained from the 10-bridge (which reached a near optimum solution) was used as a reference to measure the success in the larger networks.

The results of Figure 5 show that in the case of 10 bridges, both GAs and SFL were able to obtain solutions that satisfy all the constraints in all the trial runs. The lowest \( TLCC \) ($5,400,000) was obtained by the modified SFL. Also, the average \( TLCC \) obtained by the SFL ($5,520,000) was much less than that of the GAs ($6,883,000).

On the experiments with 50 bridges, the GAs was able to achieve only 80% success, as compared to 100% using the modified SFL. It is noted that, the best and the average \( TLCC \) obtained using the modified SFL were also less than those obtained using the GAs. The same trend in the results was
also obtained in the experiments with 100 bridges. The modified SFL outperformed the GAs in both the success rate and the value of the TLCC.

Experimenting with a larger bridge network of 250 bridges, the GAs was not able to provide any feasible solution, while the modified SFL achieved a success rate of 60%. A comparison of the processing time results is also shown in Figure 6. As the number of bridges increased, the processing time increased exponentially. In general, however, the modified SFL proved to be much faster than the GAs. In the case of 50 bridges, for example, the average processing time of the GAs was 34min:17sec, as compared to 11min:22sec using the modified SFL algorithm. It is noted that since the GAs was not able to provide any feasible solution for the case of 250 bridges, the time to reach a successful solution was not recorded (Figure 6).
Based on this discussion, the complexity of the problem substantially increases as the number of bridges increases. The variables, constraints, and the whole solution space become too large. For example, in the case of 250 bridges, the number of possible solutions is $4^{1250}$, which is extremely large.

From the extensive experimentations of this research, problem size still represents a huge challenge for optimizing infrastructure works maintenance and repair decisions. While the integrated model coupled with the modified SFL algorithm were reasonably capable of optimizing decisions for up to 250 bridges, it may not perform well if the model is expanded to the case of multiple bridge components (e.g., deck, substructure, and superstructure). In addition to experimenting with other evolutionary algorithms, some modeling adjustments may need to be applied to handle large scale problems.

SUMMARY AND CONCLUDING REMARKS

A BDMS that integrates both project-level and network-level decisions was developed utilizing GAs and modified SFL techniques for optimizing maintenance and repair activities for bridge decks. The BDMS is flexible and allows a bridge deck to be selected for repair more than once during the planning horizon. The BDMS is implemented on a spreadsheet program to utilize its familiar interface and powerful functions. Ten trial runs with different number of bridges were experimented with to evaluate the performance of both the GAs and the modified SFL. The results of the experiments showed that the modified SFL was noticeably better in performance than GAs. This is due to its use of a step term in Equation 4 to adjust and refine solutions (i.e., deeper local search). This is opposed to the GAs where the crossover exchanges large portions of the parent chromosome to cause slower refinement to solutions, particularly with a small mutation rate.

APPENDIX I. Pseudocode for SFL algorithm

Begin;
    Generate random population of $P$ solutions (individuals);
    For each individual $i \in P$: calculate fitness $(i)$;
    Sort the whole population $P$ in descending order of their fitness;
        Divide the population $P$ into $m$ memeplexes;
        For each memplex;
            Determine the best and worst individuals;
            Improve the worst individual position using Eqs. 4 and 3;
            Repeat for a specific number of iterations;
        End;
    Combine the evolved memeplexes;
    Sort the population $P$ in descending order of their fitness;
    Check if termination = true;
End;

REFERENCES


