Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization

Muzaffar Eusuff , Kevin Lansey & Fayzul Pasha

To cite this article: Muzaffar Eusuff , Kevin Lansey & Fayzul Pasha (2006) Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, Engineering Optimization, 38:2, 129-154, DOI: 10.1080/03052150500384759

To link to this article: https://doi.org/10.1080/03052150500384759

Published online: 25 Jan 2007.

Engineering Optimization
Vol. 38, No. 2, March 2006, 129–154

Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization

MUZAFFAR EUSUFF†, KEVIN LANSEY*‡ and FAYZUL PASHA‡

†Department of Water Resource Sacramento, California, USA
‡Department of Civil Engineering and Engineering Mechanics, The University of Arizona, Tucson, Arizona 85721, USA

(Received 29 September 2004; revised 15 March 2005; in final form 10 July 2005)

A memetic meta-heuristic called the shuffled frog-leaping algorithm (SFLA) has been developed for solving combinatorial optimization problems. The SFLA is a population-based cooperative search metaphor inspired by natural memetics. The algorithm contains elements of local search and global information exchange. The SFLA consists of a set of interacting virtual population of frogs partitioned into different memeplexes. The virtual frogs act as hosts or carriers of memes where a meme is a unit of cultural evolution. The algorithm performs simultaneously an independent local search in each memeplex. The local search is completed using a particle swarm optimization-like method adapted for discrete problems but emphasizing a local search. To ensure global exploration, the virtual frogs are periodically shuffled and reorganized into new memplexes in a technique similar to that used in the shuffled complex evolution algorithm. In addition, to provide the opportunity for random generation of improved information, random virtual frogs are generated and substituted in the population.The algorithm has been tested on several test functions that present difficulties common to many global optimization problems. The effectiveness and suitability of this algorithm have also been demonstrated by applying it to a groundwater model calibration problem and a water distribution system design problem. Compared with a genetic algorithm, the experimental results in terms of the likelihood of convergence to a global optimal solution and the solution speed suggest that the SFLA can be an effective tool for solving combinatorial optimization problems.

Keywords: Combinatorial optimization; Meta-heuristic; Memetic algorithm; Memes; Genes; Shuffled frog-leaping algorithm

1. Introduction

In many science and engineering problems, it is important to find the minimum or maximum of a function of many variables. For some problems, efficient analytically based algorithms exist, such as linear programming, for obtaining globally optimal solutions. However, for discrete and/or combinatorial optimization problems, no such efficient algorithms are available for a general problem. It is often necessary to use heuristics, or so-called meta-heuristics, because

*Corresponding author. Email: lansey@engr.arizona.edu exact algorithms, such as branch-and-bound and dynamic programming, may be limited by computational requirements.

In mathematical programming, a heuristic refers to a procedure that seeks a solution to an optimization problem but does not guarantee that it will find one. Heuristics use a heuristic function to guide the search, as a human would do. The heuristic search can be either an informed search or a blind search. Greedy algorithms employ blind searches that use little information about the system. The heuristics that use the heuristic function in a strategic way (*i.e.* an informed search) are referred to as meta-heuristics.

A meta-heuristic is a general framework for heuristics for solving hard global optimization problems. Most meta-heuristics are based on some kind of natural phenomenon. Examples of commonly used meta-heuristics are briefly described below.

(i) *Ant colony optimization.* Based on the foraging behavior exhibited by ant colonies in their search for food, Dorigo *et al.* (1996) developed a meta-heuristic known as the ant colony optimization (ACO) algorithm. ACO is inspired by the fact that ants are able to find the shortest route between their nest and a food source despite being nearly blind. This result is achieved by using a chemical, pheromone, as trails that act as an indirect communication form. Ants deposit the chemical trails whenever they travel. Although the path taken by individual ants from the nest in the search for food is random, when many ants are searching for food at the same time, the overall paths are affected by the trails laid by the group of ants. As more ants travel on paths with higher pheromone intensities, the pheromone on these paths builds up more, increasing the probability that other ants choose that path. The ACO algorithm makes use of the approach taken by the colony of cooperating individuals and adopts a stochastic decision-making policy using local information.

(ii) *Genetic algorithms.* Genetic algorithms (GAs), introduced by Holland (1975), are inspired by the genetic mechanisms of natural species evolution. GAs are based on the theory of Darwinian evolution and survival of the fittest. Genotypes of individuals are generally coded as chromosome-like bit strings. The basic analogy of GAs is that a set of solutions represents a population, each solution represents an individual within the population, and the objective function value of each solution represents that individual's fitness. A new population, or generation, arises from the previous generation through competition based on fitness. Transition rules are applied on a stochastic basis so that the fittest individuals have the highest probability of being selected to produce the next generation. After the selection process, genetic operators, such as crossover and mutation, are applied to complete the transition from one generation to the next.

(iii) *Artificial neural networks*. Artificial neural network (ANN) technology, inspired by neurobiological (the brain's structure) theories of massive interconnection and parallelism, is well suited to such tasks as pattern recognition and discrimination. One of the first researchers to extend the application of neural networks into optimization was Hopfield (1984), and it has been successfully applied to a variety of combinatorial optimization problems. The ANN learns to solve a problem by developing a memory capable of associating a large number of input parameters with a resulting set of outputs or effects. The ANN develops a solution system by training using examples given to it.

(iv) *Particle swarm optimization*. Particle swarm optimization (PSO), which is based on how birds flock, is described in a later section.

(v) *Shuffled complex evolution algorithm*. The shuffled complex evolution (SCE) algorithm, which is based on evolution, is also described in a later section.

(vi) *Simulated annealing*. Simulated annealing algorithms (van Laarhoven and Aarts 1987) are based on an analogy with the thermodynamic process of cooling solids. When a material is melted and allowed to cool slowly (*i.e.* annealed), a crystal lattice emerges that minimizes the energy of the system. Simulated annealing uses this analogy to search for 'minimum-energy' configurations of the decision variables, where energy is represented by the objective function value for a given solution. The transition rules from one state to the next are applied stochastically, so that occasionally an inferior solution will be chosen over a better solution. The probability that this occurs is larger at early stages of the annealing process than at later stages, when the temperature of the system has cooled.

Upon investigation it has been found that the ideas used in the SCE algorithm and PSO can be combined to design an improved meta-heuristic to solve discrete and/or combinatorial problems. The new meta-heuristic is called the shuffled frog-leaping algorithm (SFLA) and is based on the memetics of living beings.

The remainder of the paper is organized as follows: section 2 describes the basic concepts of memetics; section 3 presents short descriptions of the SCE algorithm and PSO; section 4 discusses the development of the SFLA meta-heuristic and the algorithm structure; section 5 includes a discussion on the sensitivity tests of SFLA performance to its parameters; section 6 presents experimental evaluations of the SFLA and comparison with a GA; section 7 provides the conclusion to the paper. Appendix A details some experimental test problems.

2. Basic concept of the memetic algorithm and memetics

A memetic algorithm (MA) is a population-based approach for a heuristic search in optimization problems. This class of algorithms is gaining acceptance, in particular, given that well-known large combinatorial optimization problems have been solved by MAs while other meta-heuristics have failed.

The first use of the term memetic algorithms in the computing literature appeared in work by Moscato (1989). He presented a heuristic that combined simulated annealing for a local search with a competitive and cooperative game between agents interspersed with the use of a crossover operator. The term memetic algorithm comes from 'meme' (Dawkins 1976). Meme (pronounced 'meem') is a contagious information pattern that replicates by parasitically infecting human and/or animal minds and altering their behavior, which causes them to propagate the pattern. The actual content(s) of a meme, called memotype(s), are analogous to the chromosome(s) of a gene. An idea or information pattern is not a meme until it causes someone to replicate it or to repeat it to someone else. All transmitted knowledge is memetic. Dawkins (1976), who coined the word in his book *The Selfish Gene*, defines the meme as simply a unit of intellectual or cultural information that survives long enough to be recognized as such and that can pass from mind to mind. Examples of memes are songs, ideas, catch phrases, clothes fashions and ways of making pots or of building arches. Just as genes propagate themselves in the gene pool via sperm or eggs, memes propagate themselves in the meme pool by leaping from brain to brain via a process that, in the broad sense, can be called imitation. If a scientist hears or reads about a good idea, he passes it to his colleagues and students. He mentions it in his articles and his lectures and may try to improve upon it. If the idea catches on, it can be said to propagate itself, spreading from brain to brain.

2.1 *Comparison of memes and genes*

The implicit goals of genes and memes are different to the degree that they use different mechanisms for spreading from one vehicle to another one. Memes will be positively selected mainly for increased communicability among the hosts (described as frogs in the SFLA). Genes will be selected mainly for sexual reproducibility (Heylighen 1992).

Memetic and genetic evolution are subjected to the same basic principles, *i.e.* possible solutions are created, selected according to some measure of fitness, combined with other solutions and possibly mutated. Memetic evolution, however, is a much more flexible mechanism. Genes can only be transmitted from parents or parent in the case of asexual reproduction to offspring. Memes can, in principle, be transmitted between any two individuals. Since genes are transmitted between generations, higher organisms may take several years to propagate. Memes can be transmitted in the space of minutes. Gene replication is restricted by the rather small number of offspring from a single parent, whereas the number of individuals that can take over a meme from a single individual is almost unlimited. Moreover, it seems much easier for memes to undergo variation since the information in the nervous system is more plastic than that in deoxyribonucleic acid and since individuals can come into contact with many different sources of novel memes. Therefore, meme spreading is much faster than gene spreading. The other difference between ‘memes’ and ‘genes’ is that memes are processed and possibly improved by the person that holds them – something that cannot happen to genes.

3. Description of the shuffled complex evolution algorithm and particle swarm optimization

As noted, the SFLA extends two meta-heuristics applicable to continuous optimization problems; the SCE algorithm and PSO. The principles in those algorithms evolved to a new meta-heuristic for solving hard combinatorial optimization problems. As a background, summary descriptions of the SCE algorithm and PSO are presented below.

3.1 *Shuffled complex evolution algorithm*

The SCE algorithm (Duan *et al.* 1992) combines the ideas of controlled random search (CRS) algorithms (Price 1978, 1983, 1987) with the concepts of competitive evolution (Holland 1975) and shuffling.

3.1.1 Some important features of the shuffled complex evolution approach

(i) The philosophy behind the SCE algorithm is to treat the global search as a process of natural evolution. The population is partitioned into several communities (complexes), each of which is permitted to evolve independently. After a defined number of evolutions, complexes are forced to mix, and new communities are formed through the process of shuffling. This strategy helps to improve the solution by sharing the information and properties independently gained by each community.

(ii) Each member of a community (complex) is a potential parent with the ability to participate in a process of reproduction. At each evolutionary step, only subsets of the complex, called subcomplexes, are considered. The subcomplex is like a pair of parents except that there are more than two members.

(iii) To ensure that the evolution process is competitive, it is required to have higher probabilities that better parents contribute to the next generation of offspring. The use of a triangular probability distribution ensures this fairness.

(iv) The simplex algorithm (Nelder and Mead 1965), a direct search method, is used to generate the offspring, *i.e.* to evolve the subcomplexes. This strategy uses the information contained in the subcomplex to direct the evolution in an improved direction.

(v) Offsprings are also introduced at random locations within the feasible space to ensure that the process of evolution does not become trapped in poor regions or miss promising areas of the solution space.

(vi) Each new offspring replaces the worst point of the current subcomplex, rather than the worst point of the entire population. This substitution ensures that every community member has at least one opportunity to evolve before being discarded or replaced. Thus, none of the information contained in the community is ignored.

(vii) The SCE method is designed to improve on the best features of the CRS method (*i.e.* global sampling and complex evolution), by incorporating the powerful concepts of competitive evolution and complex shuffling. Both of these techniques help to ensure that the information contained in the sample is efficiently and thoroughly exploited. They also ensure that the information set does not become degenerate.

3.2 *Particle swarm optimization*

PSO is a population-based memetic-evolution-motivated meta-heuristic. PSO was originally designed and developed by Eberhart and Kennedy (1995) and it was extended by Eberhart *et al.* (1996) and Kennedy (1997).

3.2.1 Some important features of the particle swarm optimization approach

(i) Particle swarm optimization is motivated from the simulation of social behavior; *e.g.* synchrony of flocking behavior is thought to be a function of birds' efforts to maintain an optimum distance between themselves and their neighborhood.

(ii) In PSO, the particles (birds) undergo a memetic evolution while they fly. Each particle adjusts its flying state according to its own flying experience and its companions' flying experience.

(iii) A population of particles (birds) is randomly initialized with position and velocities. Each particle is treated as a point in a D-dimensional space. The ith particle is represented as $X_I = (x_{i1}, x_{i2}, \ldots, x_i)$. The best previous position (the position giving the best fitness value) of the ith particle is recorded and represented as $P_I = (p_{i1}, p_{i2}, \ldots, p_{iD})$. The index of the best particle among all the particles in the population is represented by P_g. The rate of the position change (velocity) for particle i is represented as $V_I = (v_{i1}, v_{i2}, \ldots, v_{iD})$.

(iv) The performance of each particle is measured according to a predefined fitness function that is problem specific.

(v) The particles' velocity and position are changed by

$$v_{id} = wv_{id} + c_1\mathrm{rand1}()(p_{id} - x_{id}) + c_2\mathrm{rand2}()(p_{gd} - x_{id}), \quad d = 1, \ldots, D, \tag{1}$$

$$x_{id} = x_{id} + v_{id}, \quad d = 1, \ldots, D, \tag{2}$$

where c_1 and c_2 are two positive constants, rand1() and rand2() are two random numbers in the range [0, 1] and w is the inertia weight. Equation (1) is used to calculate the particle's new velocity according to its previous velocity and the distances of its current position from its own best experience (position) and the group's best experience. The particles then fly toward a new position according to equation (2).

(vi) The inertia weight w is employed to control the impact of the previous history of velocities on the current velocity. It directs the trade-off between global (wide-ranging) and local (nearby) exploration abilities of the 'flying points'. A larger inertia weight w facilitates global exploration (searching for new areas) while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area.

4. Shuffled frog-leaping algorithm – a memetic meta-heuristic

4.1 *Philosophy behind the development of the shuffled leaping-frog algorithm*

A number of scientists have created computer simulations of various interpretations of the movement of organisms in a bird flock or fish school. Reynolds (1987) and Heppner *et al.* (1990) presented simulations of bird flocking. Both models relied heavily on manipulation of inter-individual distances, *i.e.* memetic evolution. In reference to the fish schooling, Wilson (1975), a sociologist, wrote, 'In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantages of competition for food items, whenever the resource is unpredictably distributed in patches.' This statement suggests that social sharing of information among the population offers an evolutionary advantage. This hypothesis is fundamental to the development of the frog-leaping algorithm.

Clearly, development of ideas works in a similar way. Individual research teams may be attempting to solve the same problem but using different approaches. These teams work on their own improvements and contact others to introduce better information and knowledge. Occasionally large groups may organize (e.g. conferences) and the alignment of research teams may change to different paradigms for solving the problem. A similar analogy can be made to the engineering design process that iteratively and incrementally improves technology with better information from other designs.

4.2 *The shuffled frog-leaping algorithm meta-heuristic framework*

The SFLA has been designed as a meta-heuristic to perform an informed heuristic search using a heuristic function (any mathematical function) to seek a solution of a combinatorial optimization problem. It is based on evolution of memes carried by the interactive individuals, and a global exchange of information among themselves. The following subsections discuss the framework in detail.

4.2.1 Virtual population of frogs. Traditional evolutionary algorithms rely on the concept of population. A population is a set of individuals. Each individual has an associated fitness value that measures the goodness of the individual. Time is divided into discrete steps called time loops.

In the SFLA, the individuals are not so important and the population is seen as hosts of memes, *i.e.* a memetic vector. Each host carries a single meme (consisting of memotype(s)). As noted earlier, memes and memotypes are analogous to genes and chromosomes respectively.

In this view, the SFLA does not enumerate the individuals belonging to it; rather, it uses an abstract model, called a virtual population.

Consider a group of frogs leaping in a swamp; the swamp has a number of stones at discrete locations on to which the frogs can leap to find the stone that has the maximum amount of available food. The frogs are allowed to communicate with each other, so that they can improve their memes using others' information. Improvement of a meme results in altering an individual frog's position to be optimum by changing the leaping steps (memotype(s)) of each frog. Here, the alteration of memotype(s) is only allowed to be a discrete value by analogy with the frogs and their discrete positions.

4.3 *Algorithm details*

The SFLA is a combination of deterministic and random approaches. The deterministic strategy allows the algorithm to use response surface information effectively to guide the heuristic search as in PSO. The random elements ensure the flexibility and robustness of the search pattern. The search begins with a randomly selected population Ω of frogs covering the entire swamp. The population is partitioned into several parallel communities (memeplexes) that are permitted to evolve independently to search the space in different directions. Within each memeplex, the frogs are infected by other frogs' ideas; hence they experience a memetic evolution. Memetic evolution improves the quality of the meme of an individual and enhances the individual frog's performance towards a goal. To ensure that the infection process is competitive, it is required that frogs with better memes (ideas) contribute more to the development of new ideas than frogs with poor ideas. Selecting frogs using a triangular probability distribution provides a competitive advantage to better ideas. During the evolution, the frogs may change their memes using the information from the memeplex best or the best of the entire population. Incremental changes in memotype(s) correspond to a leaping step size and the new meme corresponds to the frog's new position. After an individual frog has improved its position, it is returned to the community. The information gained from a change in position is immediately available to be further improved upon. This instantaneous accessibility to new information separates this approach from a GA that requires the entire population to be modified before new insights are available. Again, the visual representation of frogs is used as the analogy here but, in terms of spreading of ideas, the analogy can be teams of inventors or researchers improving a concept or engineers iteratively advancing a design.

After a certain number of memetic evolution time loops, the memeplexes are forced to mix and new memeplexes are formed through a shuffling process. This shuffling enhances the quality of the memes after being infected by frogs from different regions of the swamp. Migration of frogs (cross-fertilization of ideas and/or designs) accelerates the searching procedure sharing their experience in the form of infection and it ensures that the cultural evolution towards any particular interest is free from regional bias. This major transfer of ideas can result in open academic forums or as new inventions and designs are made public.

4.3.1 Steps of the shuffled frog-leaping algorithm. The SFLA meta-heuristic strategy is summarized in the following steps.

Global exploration

Step 0 *Initialize*. Select m and n, where m is the number of memeplexes and n is the number of frogs in each memeplex. Therefore, the total sample size F in the swamp is given by $F = mn$.

Step 1 *Generate a virtual population*. Sample F virtual frogs $\boldsymbol{U}(1), \boldsymbol{U}(2), \ldots, \boldsymbol{U}(F)$ in the feasible space $\Omega \subset \Re^d$, where d is the number of decision variables (*i.e.* number of memotype(s) in a meme carried by a frog). The ith frog is represented as a vector of decision variable values $\boldsymbol{U}(i) = (U_i{}^1, U_i{}^2, \ldots, U_i^d)$. Compute the performance value $f(i)$ for each frog $\boldsymbol{U}(i)$.

Step 2 *Rank frogs*. Sort the F frogs in order of decreasing performance value. Store them in an array $\boldsymbol{X} = \{\boldsymbol{U}(i), f(i), i = 1, \ldots, F\}$, so that $i = 1$ represents the frog with the best performance value. Record the best frog's position $\boldsymbol{P}_X$ in the entire population (F frogs) (where $\boldsymbol{P}_X = \boldsymbol{U}(1)$).

Step 3 *Partition frogs into memeplexes*. Partition array $\boldsymbol{X}$ into m memeplexes $\boldsymbol{Y}^1, \boldsymbol{Y}^2, \ldots, \boldsymbol{Y}^m$, each containing n frogs, such that

$$\begin{aligned} \boldsymbol{Y}^k = [&\boldsymbol{U}(j)^k, f(j)^k | \boldsymbol{U}(j)^k = \boldsymbol{U}(k + m(j-1)), f(j)^k \\ &= f(k + m(j-1)), \quad j = 1, \ldots, n], \qquad k = 1, \ldots, m; \end{aligned} \tag{3}$$

e.g., for $m = 3$, rank 1 goes to memeplex 1, rank 2 goes to memeplex 2, rank 3 goes to memeplex 3, rank 4 goes to memeplex 1, and so on.

Step 4 *Memetic evolution within each memeplex*. Evolve each memeplex $\boldsymbol{Y}^k$, $k = 1, \ldots, m$ according to the frog-leaping algorithm outlined below.

Step 5 *Shuffle memeplexes*. After a defined number of memetic evolutionary steps within each memeplex, replace $\boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^m$ into $\boldsymbol{X}$, such that $\boldsymbol{X} = \{\boldsymbol{Y}^k, k = 1, \ldots, m\}$. Sort $\boldsymbol{X}$ in order of decreasing performance value. Update the population best frog's position $\boldsymbol{P}_X$.

Step 6 *Check convergence*. If the convergence criteria are satisfied, stop. Otherwise, return to step 3. Typically, the decision on when to stop is made by a prespecified number of consecutive time loops when at least one frog carries the 'best memetic pattern' without change. Alternatively, a maximum total number of function evaluations can be defined.

Local exploration: frog-leaping algorithm

In step 4 of the global search, evolution of each memeplex continues independently N times. After the memeplexes have been evolved, the algorithm returns to the global exploration for shuffling. Below are details of the local search for each memeplex.

Step 0 Set $im = 0$ where im counts the number of memeplexes and will be compared with the total number m of memeplexes. Set $iN = 0$ where iN counts the number of evolutionary steps and will be compared with the maximum number N of steps to be completed within each memeplex.

Step 1 Set $im = im + 1$.

Step 2 Set $iN = iN + 1$.

Step 3 *Construct a submemeplex*. The frogs' goal is to move towards the optimum ideas by improving their memes. As stated earlier, they can adapt the ideas from the best frog within the memeplex $\boldsymbol{Y}^{im}$ or from the global best. Regarding the selection of the memeplex best, it is not always desirable to use the best frog because the frogs' tendency would be to concentrate around that particular frog which may be a local optima. So, a subset of the memeplex called a submemeplex is considered. The submemeplex selection strategy is to give higher weights to frogs that have higher performance values and less weight to those with lower performance values. The weights are assigned with a triangular probability distribution, *i.e.* $p_j = 2(n + 1 - j)/n(n + 1)$, $j = 1, \ldots, n$, such that within a memeplex the frog with the best performance has the highest probability $p_1 = 2/(n + 1)$ of being selected for the submemeplex, and the frog with the worst performance has the lowest probability $p_n = 2/n(n+1)$. Here, q distinct frogs are selected randomly from n frogs in each memeplex to form the submemeplex array $\boldsymbol{Z}$. The submemeplex is sorted so that frogs are arranged in order of decreasing performance ($iq = 1, \ldots, q$). Record the best ($iq = 1$) frog's position and worst ($iq = q$) frog's position in the submemeplex as $\boldsymbol{P}_{\mathrm{B}}$ and $\boldsymbol{P}_{\mathrm{W}}$ respectively. The concept of a submemeplex is illustrated in figure 1.

Step 4 *Improve the worst frog's position.* The step and new position are computed for the frog with worst performance in the submemeplex by

$$\begin{aligned}\text{step size}\,\boldsymbol{S} &= \min\{\text{int}[\text{rand}(\boldsymbol{P}_{\mathrm{B}} - \boldsymbol{P}_{\mathrm{W}})], \boldsymbol{S}_{\max}\} \quad \text{for a positive step,}\\ &= \max\{\text{int}[\text{rand}(\boldsymbol{P}_{\mathrm{B}} - \boldsymbol{P}_{\mathrm{W}})], -\boldsymbol{S}_{\max}\} \quad \text{for a negative step,}\end{aligned}$$

where rand is a random number in the range [0, 1] and $\boldsymbol{S}_{\max}$ is the maximum step size allowed to be adopted by a frog after being infected. Note that the step size has dimensions equal to the number of decision variables. The new position is then computed by

$$\boldsymbol{U}_{(q)} = \boldsymbol{P}_W + \boldsymbol{S}. \tag{4}$$

If $\boldsymbol{U}_{(q)}$ is within the feasible space Ω, compute the new performance value $f_{(q)}$. Otherwise go to step 5. If the new $f_{(q)}$ is better than the old $f_{(q)}$: *i.e.*, if the evolution produces a benefit, then replace the old $\boldsymbol{U}_{(q)}$ with the new $\boldsymbol{U}_{(q)}$ and go to step 7. Otherwise go to step 5. An illustration of memetic evolution (infection) in a submemeplex is shown in figure 2.

Step 5 If step 4 cannot produce a better result, then the step and new position are computed for that frog by

$$\begin{aligned}\text{step size}\,\boldsymbol{S} &= \min\{\text{int}[\text{rand}(\boldsymbol{P}_X - \boldsymbol{P}_W)], \boldsymbol{S}_{\max}\} \quad \text{for a positive step,}\\ &= \max\{\text{int}[\text{rand}(\boldsymbol{P}_X - \boldsymbol{P}_W)], -\boldsymbol{S}_{\max}\} \quad \text{for a negative step,}\end{aligned}$$

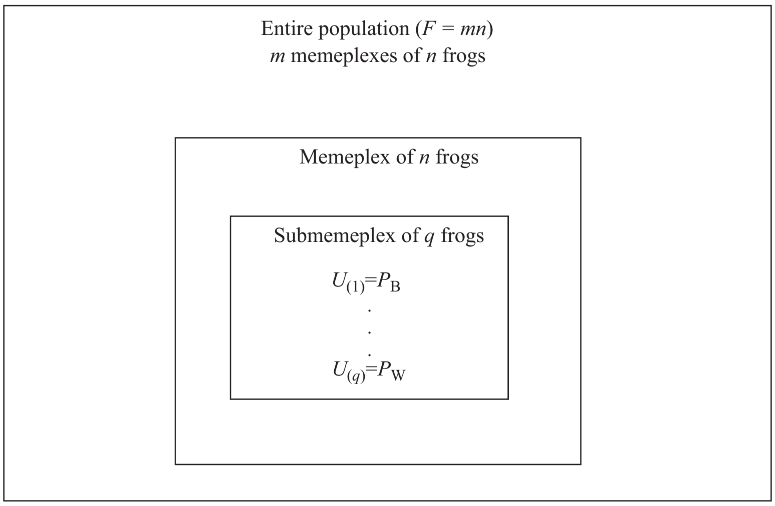

Entire population ($F = mn$)
m memeplexes of n frogs
Memeplex of n frogs
Submemeplex of q frogs
$U_{(1)}$=P_{B}
.
.
.
$U_{(q)}$=P_{W}

Figure 1. An illustration of the concept of a submemeplex. P_{B} and P_{W} correspond to the positions $U_{(1)}$ and $U_{(q)}$ of frogs with the best and worst performances respectively in the submemeplex.

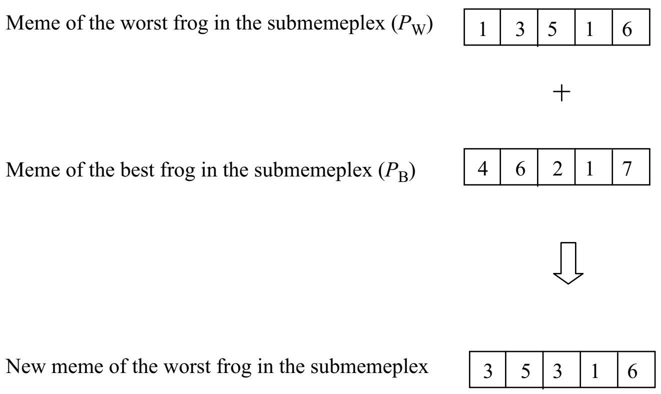

Meme of the worst frog in the submemeplex (P_W)
1 3 5 1 6
+
Meme of the best frog in the submemeplex (P_B)
4 6 2 1 7
New meme of the worst frog in the submemeplex
3 5 3 1 6

Figure 2. Illustration of the memetic evolution in a submemeplex. Each frog carries one meme and each meme consists of five memotypes. For example, if random number rand is 0.7 and the components of $\boldsymbol{S}_{\max}$ are all 3, the first memotype is changed according to $(1 + \min\{\text{int}[\text{rand}(4 - 1)], 3\} = 3)$, and the third memotype is changed according to $(5 + \max\{\text{int}[\text{rand}(2 - 5)], -3\} = 3)$.

and the new position is computed by equation (4). If $\boldsymbol{U}_{(q)}$ is within the feasible space Ω, compute the new performance value $f_{(q)}$; otherwise go to step 6. If the new $f_{(q)}$ is better than the old $f_{(q)}$, *i.e.* if the evolution produces a benefit, then replace the old $\boldsymbol{U}_{(q)}$ with the new $\boldsymbol{U}_{(q)}$ and go to step 7. Otherwise go to step 6.

Step 6 *Censorship*. If the new position is either infeasible or not better than old position, the spread of defective meme is stopped by randomly generating a new frog r at a feasible location to replace the frog whose new position was not favorable to progress. Compute $f(r)_{-}$ and set $\boldsymbol{U}_{(q)} = r$ and $f_{(q)} = f(r)$.

Step 7 *Upgrade the memeplex*. After the memetic change for the worst frog in the submemeplex, replace $\boldsymbol{Z}$ in their original locations in $\boldsymbol{Y}^{im}$. Sort $\boldsymbol{Y}^{im}$ in order of decreasing performance value.

Step 8 If $iN < N$, go to step 2.

Step 9 If $im < m$, go to step 1. Otherwise return to the global search to shuffle memeplexes.

Steps 4 and 5 of the local search are similar in philosophy to PSO. A descent direction is identified for a particular frog and the frog is moved in that direction. Here, however, since the global search is also introduced in the shuffle operation, only the local minimum is used rather than the complete population best (step 4) unless no improvement is made (step 5). Since a descent direction is implicitly applied, it may be fruitful to perform a line search rather than a random step but the simpler approach is taken here.

4.4 *Typical pattern of memetic evolution of the shuffled frog-leaping algorithm*

Figures 3(a)–(d) illustrate the progress of local and global explorations. In figure 3(a), a population of 15 virtual frogs with different memes is randomly selected within the boundary of the swamp. They have been equally partitioned into three memeplexes denoted by open triangles, full squares and full circles respectively. In each memeplex, the frogs with poor performance are affected by frogs with better performance and improve the quality of their memes. A sufficient number N of memetic evolution cycles is allowed during a time loop to ensure that all the frogs in a memeplex have an opportunity to exercise their ideas. Figure 3(b) shows the locations of the frogs at the end of one time loop. It can be seen that each of the memeplexes evolved independently and led to three different local searches.

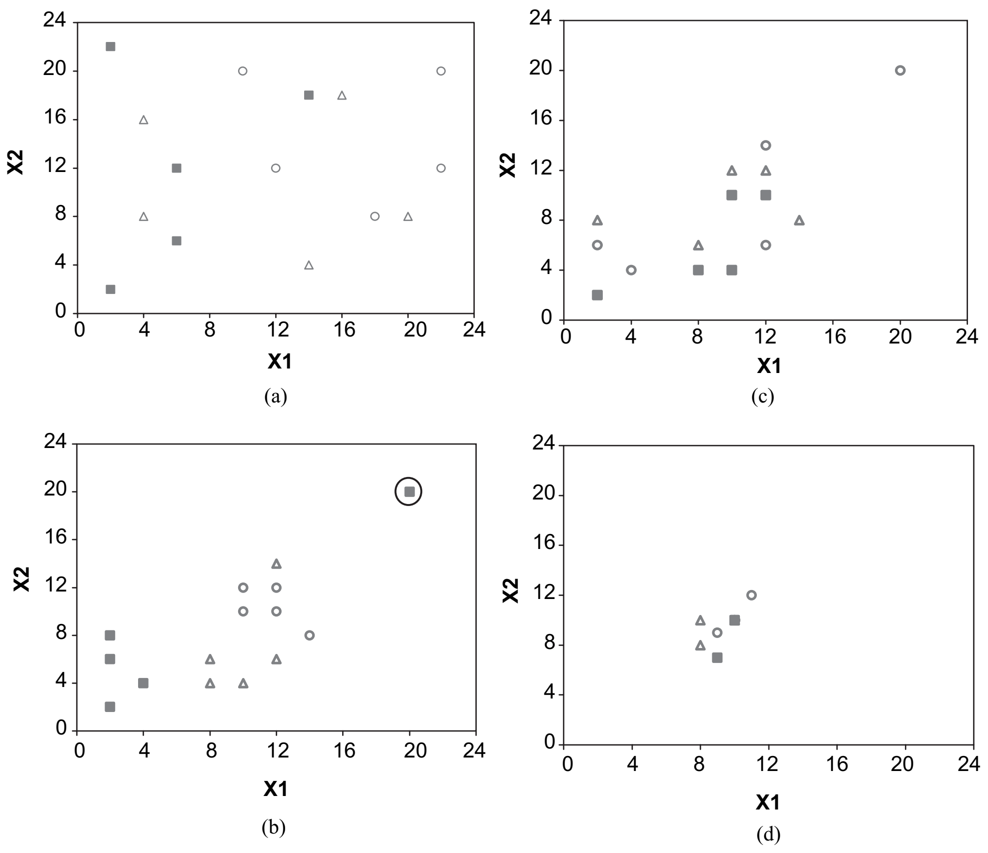
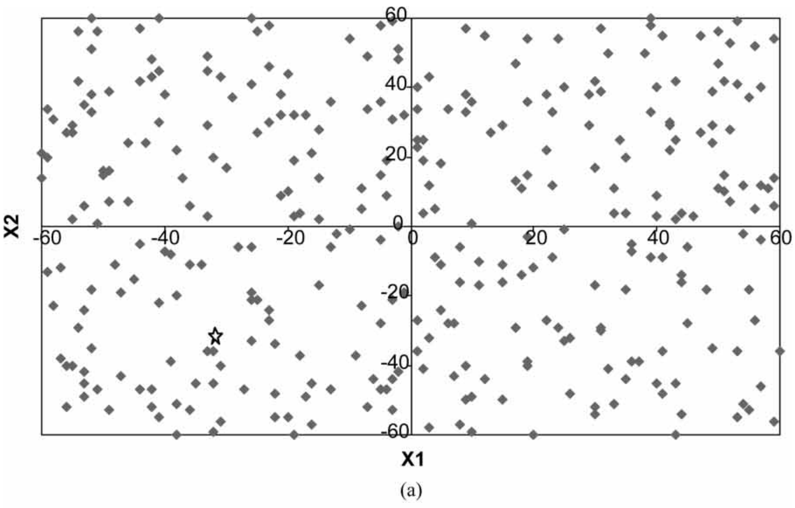


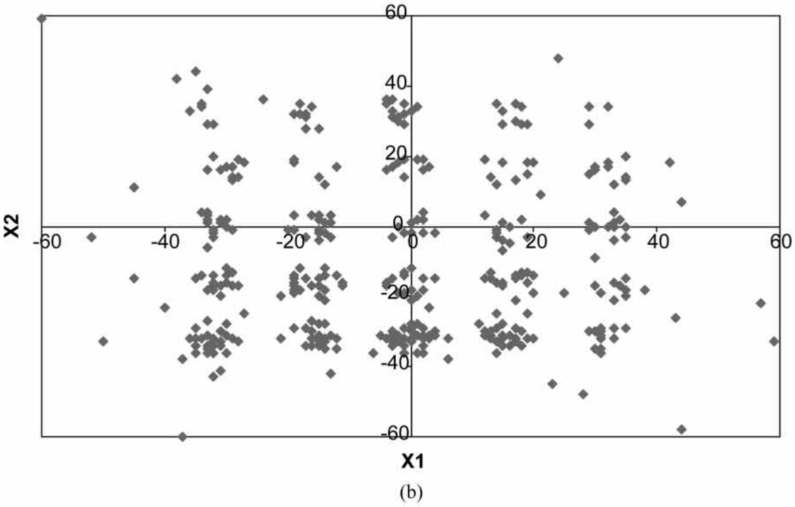
Figure 3. Illustration of the SFLA on a small scale: (a) initial population of virtual frogs (at the beginning of time loop 1); (b) independently evolved memeplexes (at the end of time loop 1); (c) shuffled frogs (at the beginning of time loop 2); (d) independently evolved memeplexes (at the end of time loop 2).

During the evolution it is possible to find a frog with an idea that is not favorable to the community, and that triggers the censorship and results a new virtual frog with a new set of memes. In figure 3(b), the circled frog in the memeplex indicated by full squares is an example of censorship.

After N evolutionary cycles the frogs are shuffled to form the new memeplexes shown in figure 3(c). The new memeplexes now have a diversity of ideas. Again, the memeplexes are evolved independently for a predefined number of cycles. Figure 3(d) shows the three memeplexes at the end of another time loop. All the memeplexes (except a few frogs) are clustering near the optimum location in the swamp. Eventually all the frogs will be infected by the unique optimal idea that represents the optimum problem solution.

Figures 4(a)–(f) show the progress of the algorithm for an actual problem (DeJong's F_5 function (details are given later)). At the end of the search the frogs will tend towards homogeneity. The positions of the frogs in the figures are defined according to their memes. At the beginning of the food search (time loop 1), frogs are randomly scattered in the swamp (in the range from –66 to +66). The unknown optimum food spot (indicated with an open star in figure 4(a)) is their ultimate goal. After time loop 4 the frogs are already being infected by better ideas, have started to adopt some common ideas and begin to cluster around a local optimum (figure 4b). As evolution and shuffling continue, frogs start to explore more new





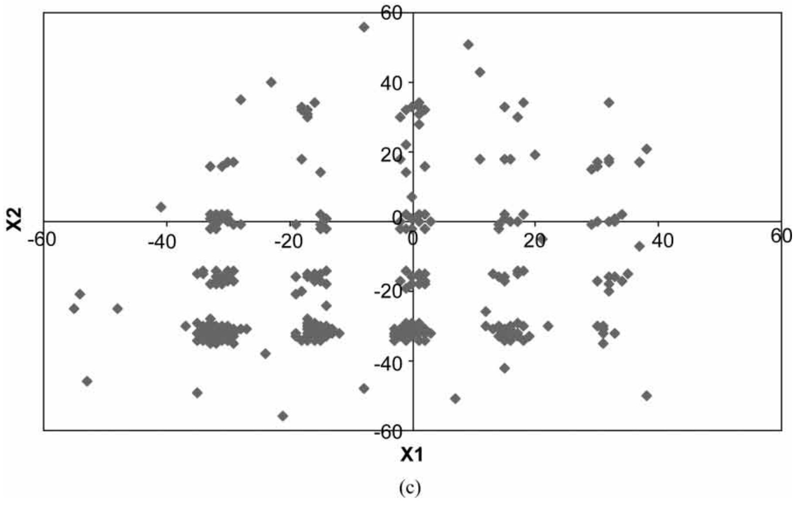
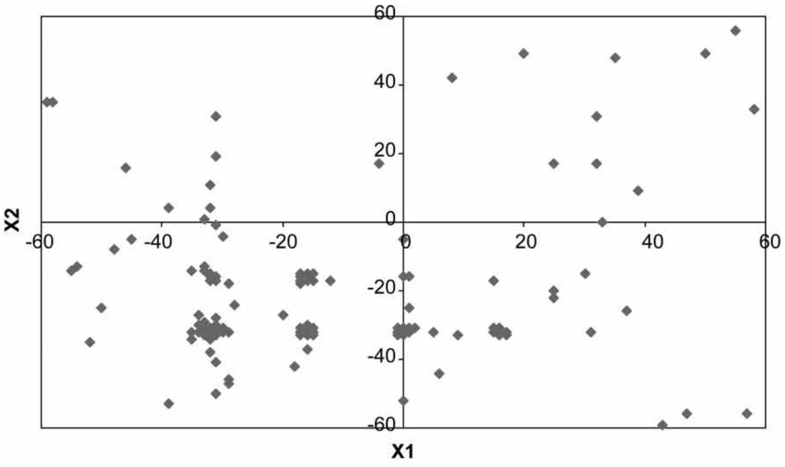
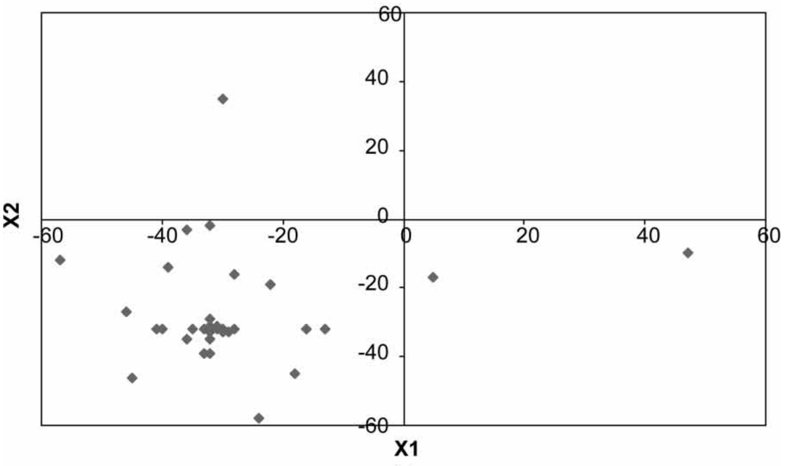


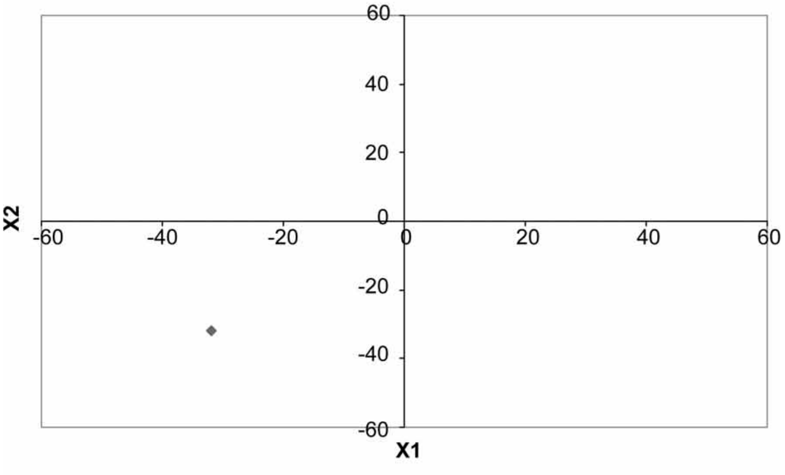
Figure 4. Typical SFLA search pattern for DeJong's F_5 function (Fn5), showing the positions of the virtual frogs at the ends of time loops (a) 1, (b) 4, (c) 6, (d) 10, (e) 15 and (f) 20.



(d)



(e)



(f)

Figure 4. Continued.

ideas (time loops 6, 10 and 15) (figures 4(c)–(e)). Finally, at time loop 20, all the frogs are infected by the same ideology and gather at the optimum location.

5. Optimization parameters in the shuffled leaping-frog algorithm

Like all heuristics, parameter selection is critical to SFLA performance. SFLA has five parameters: the number m of memeplexes, the number n of frogs in a memeplex, the number q of frogs in a submemeplex, the number N of evolution or infection steps in a memeplex between two successive shufflings and the maximum step size $\boldsymbol{S}_{\max}$ allowed during an evolutionary step. At this point in the development of this meta-heuristic, no clear theoretical basis is available to dictate parameter value selection.

Based on experience, the sample size F (the number m of memeplexes multiplied by the number n of frogs in each memeplex), in general, is the most important parameter. An appropriate value for F is related to the complexity of the problem. The probability of locating the global (or near-global) optima increases with increasing sample size. However, as the sample size increases, the number of function evaluations to reach the goal increases, hence making it more computationally burdensome. In addition, when selecting m, it is important to make sure that n is not too small. If there are too few frogs in each memeplex, the advantage of the local memetic evolution strategy is lost. The response of the algorithm performance to q is that, when too few frogs are selected in a submemeplex, the information exchange is slow, resulting in longer solution times. On the other hand, when too many frogs are selected, the frogs are infected by unwanted ideas that trigger the censorship phenomenon that tends to lengthen the search period. The fourth parameter, N, can take any value greater than 1. If N is small, the memeplexes will be shuffled frequently, reducing idea exchange on the local scale. On the other hand, if N is large, each memeplex will be shrunk into a local optimum. The fifth parameter, $\boldsymbol{S}_{\max}$, is the maximum step size allowed to be adopted by a frog after being infected. $\boldsymbol{S}_{\max}$ actually serves as a constraint to control the SFLA's global exploration ability. Setting $\boldsymbol{S}_{\max}$ to a small value reduces the global exploration ability making the algorithm tend to be a local search. On the other hand, a large $\boldsymbol{S}_{\max}$ may result in missing the actual optima, as it is not fine-tuned.

Although, at present, there is no guidance to select proper values for the parameters, experimental results presented in the next section provide a better understanding of the parameters' impacts.

6. Experimental evaluations of the shuffled leaping-frog algorithm

Two types of experimental evaluation of the SFLA were performed. Firstly, the algorithm was tested on several mathematical test functions that are considered as benchmarks. Secondly, the algorithm was used as an application tool in groundwater model calibration. The SFLA performance was compared with the GA and published results, where available. In addition to the work presented here the present authors have applied the SFLA to literature water distribution system problems with promising results (Eusuff and Lansey 2003).

6.1 *Mathematical test problems and results*

Initially, the SFLA was used to solve five of DeJong's test problems. These test functions consist of a set of benchmarks that is considered by many to be a standard for performance comparisons of evolutionary algorithms and reflect different aspects of the global optimization problem. However, the continuous variables are considered in discrete space here. The details of these problems are presented in appendix A.

For each test problem, a total of 20 runs with different combinations of the SFLA's parameters were executed. In each run, the stopping criterion was that at least one frog would carry the 'so far best memetic pattern' for ten consecutive time loops (shuffles). In 95% of all runs the criterion was met although the required number of function evaluations varied between the runs and problems.

For comparison, the test functions were also solved using Evolver (1997), a GA-based optimizer. Again, a total of 20 runs was made for each problem with different combinations of initial genes, numbers in the population, and crossover and mutation rates. In each run, the stopping criterion was that the best function value would not change for ten consecutive generations. The GA failed to find the optimal values in 20% of the cases. Figure 5 summarizes the comparisons between the SFLA and the GA for the test functions. The vertical bars represent the average number of function evaluations to find the global optima in successful runs.

For more robust testing of the SFLA, another set of six well-established test functions with discrete variables were selected and solved using both the SFLA and the GA (see appendix A). These functions and the impact of the SFLA parameter selection were analysed in more detail than the first set in terms of the SFLA's robustness in finding the global optimal solution and its efficiency in terms of the number of function evaluations required to find the global solution. Like any heuristic algorithm, the performance of finding global optimum using the SFLA largely depends on the selection of algorithm parameters. Different sets of parameter values alter the chances of converging to the global optimum but will also vary the number of function evaluations required. Therefore, compromise between computation costs and the probability of success is necessary.

To study the influence of different parameters, an enumerative study was completed that varied all the SFLA parameters except the maximum step size $S_{\max}$. A preliminary study was made for different $S_{\max}$ values, and it was found that that the success rate of achieving optimal solution was highest at 100% of the variable range (results not shown); hence it was not included in the parameter sensitivity experiments. In the experiments, the number m of memeplexes was varied in the range from 10 to 50 with an increment of 10 and from 70 to 100 with an increment of 30. Values for the number n of frogs in each memeplex were 10, 20,

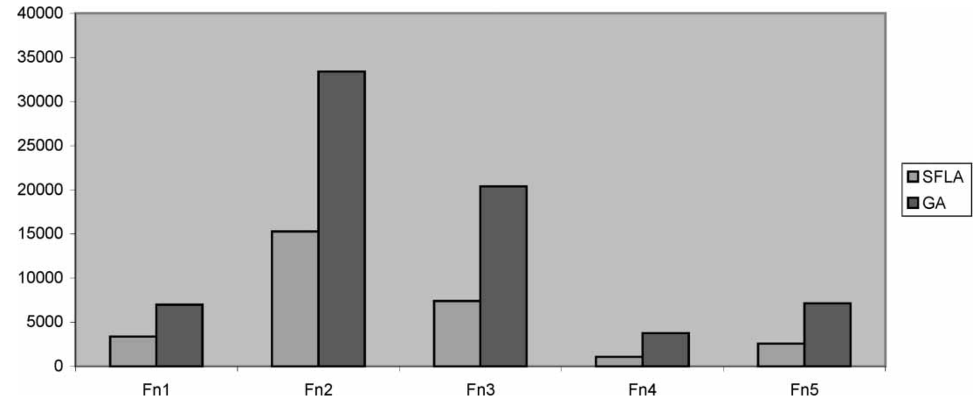


Figure 5. Number of function evaluations for the SFLA and the GA to reach the optimal solution of DeJong's test functions (the average number in successful runs).

Table 1. Comparison of the test results for the SFLA and the GA.

	Number of function evaluations[†] required	
Test problem	By the SFLA	By the GA
Gear	2 074	Failed[‡]
Cutting stock	1 097	20 000
Trim loss	1 227	Failed
Traveling salesman	1 080	10 000
Simpleton25	76 666	2 979
Simpleton50	28 167	10 000

[†]Minimum number of function evaluations to find the optimal solution.
[‡]Failed to find the solution.

30, 40, 50, 70, 100, 150, 200 and 300. The number q of frogs in a submemeplex was varied from 5 to 20 with an increment of 5 and the number N of evolutions between shuffles was varied from 5 to 35 with an increment of 5. The same parameters ranges were used for all the problems except the Simpleton25 and Simpleton50 problems. It was observed that the optimal value of Simpleton25 and Simpleton50 was hard to achieve with the range of m from 10 to 100. Therefore, for these two problems the range of m was extended up to 300.

For comparison with the GA over a range of parameters, many combinations of crossover and mutation rates were considered, such as crossover rate from 0.2 to 0.9 with an increment of 0.05, and the mutation rate from 0.01 to 0.15 in increments of 0.001.

Table 1 lists the comparative results summary of the test problems solved by the SFLA and the GA. The SFLA was able to find the optimal solutions to all the problems. Moreover, it is evident that the SFLA performed well, solving the first four problems; however, it did not show better results than the GA for the two Simpleton problem. The solutions to these problems are at the boundaries of the feasible region and may be the reason for not converging to the optimal solutions.

6.1.1 Results and effect of parameter selection.

In this section, the effect of parameter selection on the results of the optimization runs for the test problems are presented. Firstly,

Table 2. Test results on the sensitivity of the SFLA performance to different parameters when applied to DeJong's F_5 function (Fn5). Base values are $m = 20$, $n = 20$, $q = 5$, $N = 15$ and $S_{\max} = 45\%$ of the variable bounds.

Sensitivity to numbers m of memeplexes				Sensitivity to number n of frogs in a memeplex			
	Number of evaluations required				Number of evaluations required		
m	One optima[†]	All optima[‡]	Comments	n	One optima	All optima	Comments
5	Not found	Not found	Best[§] = 1.99	5	2714	4 558	Success = 70%
10	2283	13 506	Success* = 80%	10	3507	9 844	Success = 80%
20	3143	21 484	Success = 90%	20	3143	21 484	Success = 90%
25	2641	30 692	Success = 90%	25	2664	28 292	Success = 90%
30	1720	41 706	Success = 100%	30	8200	52 631	Success = 100%

[†]One optima indicates the event when at least one frog is infected by the global optimal idea.
[‡]All optima indicate the event when all the frogs are infected by the global optimal idea.
[§]The global optimal (0.998) was not found; rather the local best was determined.
*The success rate was measured on the basis of the number of global optimal solutions found in ten runs.

Table 3. Test results on the sensitivity of the SFLA performance to different parameters for DeJong's F_5 function. Base values are $m = 20, n = 20, q = 15, N = 15$ and $\boldsymbol{S}_{\max} = 45\%$ of the variable bounds.

Sensitivity to number q of frogs in a submemeplex				Sensitivity to number N of evolution steps in a memeplex			
	Number of evaluations required				Number of evaluations required		
q	One optima[†]	All optima[‡]	Comments	N	One optima	All optima	Comments
5	3143	21 484	Success[§] = 90%	5	3391	9 297	Success = 100%
10	2103	11 317	Success = 100%	10	3995	11 034	Success = 100%
15	1070	8 491	Success = 100%	15	1070	8 491	Success = 100%
20	5237	10 466	Success = 100%	20	3950	11 098	Success = 100%

[†] One optima indicates the event when at least one frog is infected by the global optimal idea.
[‡] All optima indicate the event when all the frogs are infected by the global optimum idea.
[§] The success rate was measured on the basis of the number of global optimal solutions found in ten runs.

for DeJong's functions, it is understood that different problems will probably have different best parameters. For example, the results for DeJong's F_5 function are presented in tables 2 and 3. Secondly, the results of the sensitivity study for the second problem set are presented in tables 4– 6. Table 4 lists the number of runs for the different sets of parameters, and tables 5 and 6 list the summaries of the runs.

Although, on the basis of experiment results, substantial information was found on the range of parameter values, it must be realized that the selection is problem specific. However, as an initial guidance for parameter value selection, for a problem with variables up to 15–20, the following ranges can be recommended and may result in a good probability of success. Recommended parameter values ranges are $100 \leq m \leq 150$, $30 \leq n \leq 100$, $20 \leq N \leq 30$ and $q = 20$. Also based on experience, $\boldsymbol{S}_{\max}$ should be 100%.

Table 4. Summary results of problems solved by the SFLA.

	Value for the following problems					
	Gear	Cutting stock	Traveling salesman	Trim loss	Simpleton25	Simpleton50
Number of variables	4	6	6	8	25	50
Total runs	1960	1820	1960	1960	2520	2520
Successful runs	1198	1465	1258	1842	286	328
Success rate (%)	61.1	77.2	64.2	94.0	11.4	13.0

Table 5. Range of highest probabilities of successfully finding the global optimal solution for four test problems.

		Parameter range with the highest success rate			
Parameter	Range evaluated	Gear	Cutting stock	Traveling salesman	Trim loss
m	10–100	100	100	100	10–100
n	10–300	30–300	70–300	10–300	150–300
N	5–35	10–35	5–35	25–35	5–35
q	5–20	5–20	5–20	5–20	5–20
Number of runs		192	140	120	588
Success rate (%)		100	95	97	100

Table 6. Range of highest probabilities of successfully finding the global optimal solution for the two Simpleton test problems.

Parameter	Range evaluated	Simpleton25	Simpleton50
m	10–300	300	300
n	20–300	20–300	20–300
N	35	35	35
q	5–20	5–20	5–20
Number of runs		36	36
Success rate (%)		67	64

Overall, the success rate of the algorithm is satisfactory and encouraging. In general, the success rate of the algorithm rises with the increase in the number of frogs in the population and number of frogs in a subcomplex, but at the cost of number of function evaluation required to find the solution. It was also noted that the success rate is more sensitive to the number of memeplexes than to the number of frogs in a memeplex, which reinforces the idea of exploring more regions in the domain. This finding is most evident in the Simpleton problems. Here, the success rate decreased with an increase in the number of frogs in a memeplex above a certain value. This may be due to the nature of this type of problem; however, further research is recommended on this feature of the SFLA. The Simpleton problems indicates the difficulty of efficiently converging to an optimal solution near the boundary of the solution space. In the algorithm described, except when a random point is generated, the search will remain within the range of the set of feasible points. Thus, if all boundary points are not included in the initial set, movement to the boundary is a random occurrence. Although this scheme is successful in terms of convergence, other approaches may be more computationally efficient and should be studied.

6.2 *Groundwater model calibration problem*

To investigate the performances of the SFLA and the GA, both of these algorithms were used to calibrate a steady-state hypothetical groundwater model (figure 6), which was modelled after a real field site. The purpose of the calibration was to find the location of extraction wells at five predetermined places with known extraction rates. The objective of the calibration was to find the extraction well locations such that the calibrated water levels match the water levels shown in table 7, minimizing the least-squares errors (equation (5)). The advantage of this example is that it is a realistic problem and the global optimum solution is known.

6.2.1 Model site description. The finite-difference groundwater model grid as shown in figure 6 covers an area of 2964.2 m by 3223.3 m (9725 ft by 10 575 ft) and consists of 34 rows, 36 columns and five layers. Small cells of size 38.1 m by 38.1 m (125 ft by 125 ft) are located in the vicinity of the recharge cells while all other cells have dimensions of 121.92 m by 121.92 m (400 ft by 400 ft).

Boundary conditions of the aquifer include the river along part of the eastern side (elevation, 396.34 m (1300 ft)) and constant fluxes along the rest of the boundaries. The constant fluxes were applied in the form of extraction wells. The extraction rates for the boundary were determined in a previous study by an automated calibration method. Natural recharge to the aquifer is 0.61 mm day^{-1} (0.002 ft day^{-1}) and the applied recharge rate in the recharge basins is 0.91 m day^{-1} (3 ft day^{-1}).

The bottom elevation of the unconfined layer is variable, ranging from 64.0 m to 73.15 m (210 to 240 ft). Underlying this layer is a confining layer with a bottom elevation of 60.96 m

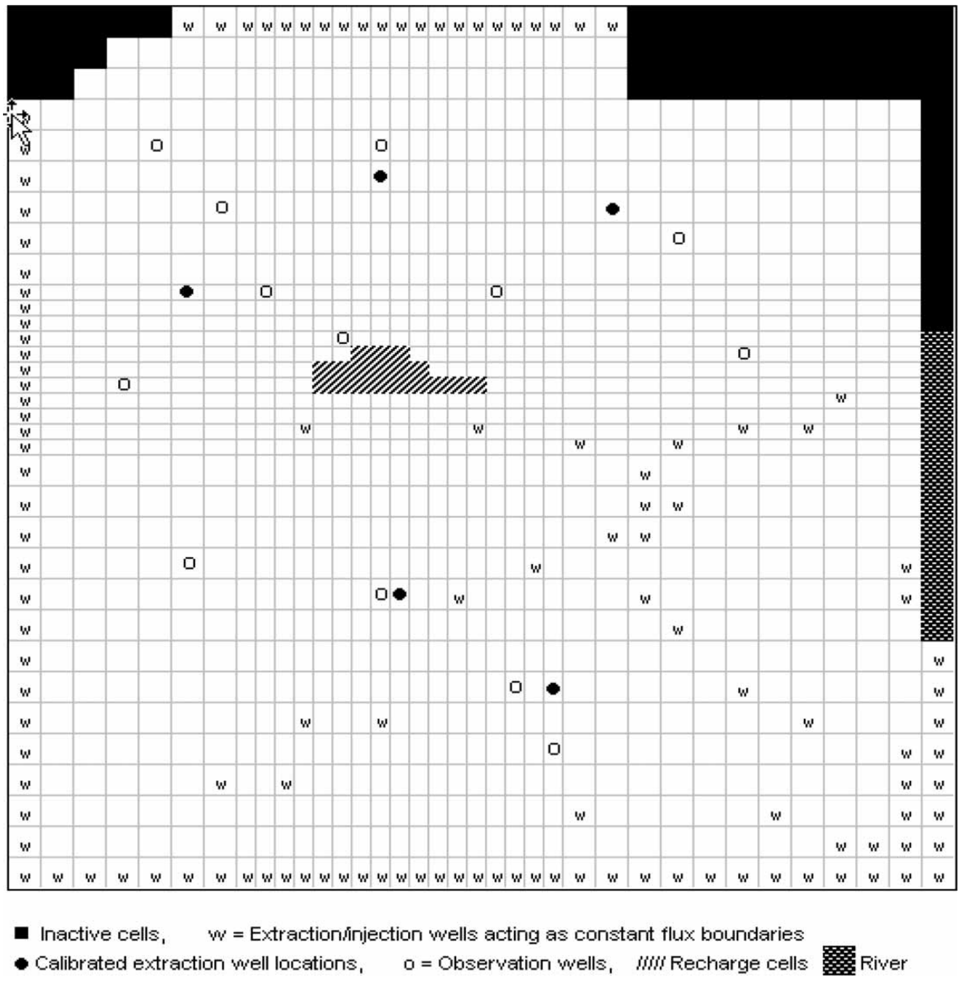

■ Inactive cells,
w = Extraction/injection wells acting as constant flux boundaries
● Calibrated extraction well locations,
o = Observation wells,
///// Recharge cells
River

Figure 6. Groundwater model grid for the calibration problem.

Table 7. Observation well locations and observed water level data used for the calibration of the groundwater model.

Observation well locations		Water levels
Row	Column	(ft)
7	7	1301.0
8	28	1354.0
5	15	1333.0
30	24	1330.0
5	5	1249.0
14	30	1379.0
16	4	1396.0
10	9	1365.0
13	13	1453.0
10	21	1389.0
25	15	1398.0
28	22	1352.0
24	6	1369.0

(200 ft) and a vertical conductivity of 0.000 152 m day^{-1} (0.0005 ft day^{-1}). Underlying this layer are four confined aquifers 15.24 m (50 ft) thick that are separated from each other by thin aquitard layers. The hydraulic conductivity of the top layer varies between 12.19 and 18.29 m day^{-1} (between 40 and 60 ft day^{-1}). The recharge basin cells have a conductivity of 18.29 m day^{-1} (60 ft day^{-1}). The transmissivity of the bottom four layers is 116.13 m^2 day^{-1} (1250 ft^2 day^{-1}). The porosities are 0.35 for layers 1 and 4, 0.30 for layers 2 and 3, and 0.20 for layer 5.

6.2.2 Problem statement.

The GA and the SFLA were linked with a groundwater simulation model, MODFLOW (Harbaugh and McDonald 1996). The responses of the different decision variables (extraction well locations) were reflected in the simulated water levels. The heuristic function (objective function) value was calculated using the observed water level values H_o and the simulated water level values H_s. Mathematically, the heuristic function is

$$\text{minimize } H_d = \sum_{i=1}^{N_w} (H_{si} - H_{oi})^2, \tag{5}$$

where N_w is the number of observation wells. The constraints are the equations of the physical system of the model (*i.e.* the groundwater equations). The ranges on the extraction well locations are listed in table 8.

6.2.3 Results and discussion.

Ten calibration runs with different optimization parameters were completed using the SFLA and the GA. The SFLA successful rate was in 100% of all runs while the GA found the true solution in 70% of its runs. For the fastest result, the SFLA determined the well locations in 926 evaluations, whereas the GA found the solution in the 2281st evaluation. On average, the SFLA and the GA required about 1200 iterations and 2500 iterations respectively. Figure 7 shows the SFLA and the GA solution convergence for the groundwater model calibration problem.

Computational time required to solve this type of simulation–optimization problem is governed primarily by the time required for simulation. In this study, a single

Table 8. Possible extraction well locations with the lower and upper bounds on cell locations with respective extraction rates.

		Exact well locations									
		Well 1		Well 2		Well 3		Well 4		Well 5	
		Row 10	Column 6	Row 7	Column 26	Row 6	Column 15	Row 25	Column 16	Row 28	Column 24
Bounds[†]	Lower	2	2	2	20	2	13	18	2	18	21
	Upper	17	12	17	33	17	19	33	20	33	33
Extraction rates (m^3/day)		4.3×10^4		21.3×10^4		18.4×10^4		3.4×10^4		2.8×10^4	
Extraction rates (ft^3/day)		1.5×10^6		7.5×10^6		6.5×10^6		1.2×10^6		1.0×10^6	

[†]For example, the bounds were set as (2, 2) and (17, 12) for the calibrated location (10, 6) of well 1.

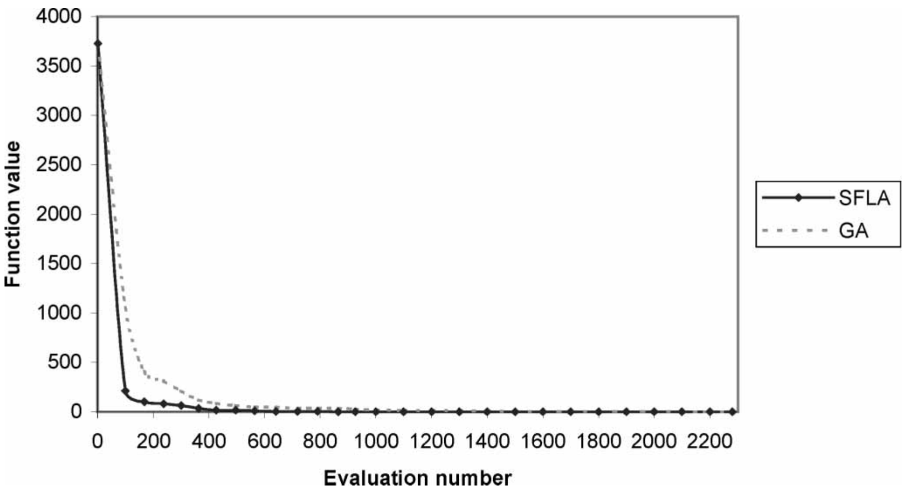


Figure 7. The SFLA and GA convergences of the best solutions for the groundwater model calibration problem.

simulation–optimization cycle took about 3 s on a 1.5 GHz personal computer. Therefore, for a solution requiring about 1000 iterations, it required a computational time of 50 min.

7. Conclusions

This paper described the SFLA, a memetic meta-heuristic to solve combinatorial optimization problems. A memetic algorithm is a population-based approach for heuristic search. The SFLA starts with a sample virtual population of frogs, leaping in a swamp, searching for the optimum location of food. Frogs act as hosts of memes. A meme is considered as a unit of ideas or cultural evolution. Each meme consists of memotype(s). During the memetic evolution the memes of the frogs are changed, resulting in a change in their position towards the goal. The change in or evolution of memes occurs when the frogs are infected by other better ideas.

The SFLA performance was compared with a GA for a series of test problems. The results for 11 theoretical test problems (functions) and two applications show that the SFLA performed better than or at least comparable with the GA for almost all problem domains and was more robust in determining the global solution. Four realistic engineering problems were also solved and results compared with literature results from several optimization algorithms. Again the results were extremely promising in terms of robustness of finding solutions and solution speed.

The SFLA is quite general but relies strongly on suitable model parameters. No clear guidance is available to direct the selection of the parameters. Analysis with respect to the performance of the algorithm showed that different problems had different sets of optimal parameters. Given a specific problem, a particular set of parameters is probably most appropriate. Despite all these facts, based on the experimental results, the present authors have suggested a set of ranges to choose the parameter values. Further study and extensive experimental testing is required to establish a basis for the parameter selection. The results for the Simpleton problems identified the need for fine tuning of the algorithm to determine the solution near the boundary. Additional work is ongoing in this regard.

In summary, the SFLA is a promising robust meta-heuristic. Although the SFLA has been developed and tested for combinatorial problems, it appears capable of being extended to mixed-integer problems. Also, like GAs and many meta-heuristics, the SFLA is very suitable for parallelization and its potential should be investigated.

Acknowledgements

The present authors acknowledge the financial support provided by the following agencies: Tucson Water, City of Tucson, Arizona, Sanitation Districts of Los Angeles County, American Water Works Association Research Foundation, City of Phoenix, Arizona, Subregional Operators group (Phoenix-area cities), Water Replenishment District of Southern California, City of Riverside, California, City of Los Angeles Department of Water and Power.

References

Dawkins, R., *The Selfish Gene*, 1976 (Oxford University Press: Oxford).

Deb, K. and Goyal, M., Optimizing engineering designs using a combined genetic search, in *Proceedings of the 7th International Conference on Genetic Algorithms*, edited by T. Back, pp. 521–528, 1997 (Morgan Kaufmann: San Mateo, CA).

Dorigo M., Maniezzo, V. and Colorni, A., Ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst., Man Cybernetics B*, 1996, **26**(1), 29–41.

Duan, Q., Sorooshian, S. and Gupta, V., Effective and efficient global optimization for conceptual rainfall-runoff models. *Water Resources Res.*, 1992, **28**(4), 1015–1031.

Eberhart, R.C. and Kennedy, J., A new optimizer using particles swarm theory. *Proceeedings of the 6th International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995, pp. 39–43, 1995 (IEEE Service Center: Piscataway, NJ).

Eberhart, R.C., Dobbins, R.W. and Simpson, P., *Computational Intelligence PC Tools*, 1996 (Academic Press: Boston, MA).

Eusuff, M.M. and Lansey, K.E., Optimization of water distribution network design using the shuffled frog leaping algorithm (SFLA). *J. Water Resources Planning Mgmt*, Am. Soc. Civ. Engrs, 2003, **129**(3), 210–225.

Evolver, *Evolver – the Innovative Optimizer for Windows*, 1997 (Palisade Corporation: New York).

Harbaugh, A.W. and McDonald, M.G., User's documentation for MODFLOW-96, an update to the US Geological Survey modular finite-difference groundwater flow model. US Geological Survey Open-File Report 96-485, 1996.

Harjunkoski, I., Westerlund, T., Porn, R. and Skrifvars, H., Different transformations for solving non-convex trim loss problems by MINLP. *Eur. J. Operational Res.*, 1998, **105**, 594–603.

Heppner, F. and Grenander, U., A stochastic nonlinear model for coordinated bird flocks, in *The Ubiquity of Chaos*, edited by S. Krasner, pp. 233–238, 1990 (AAAS Publications: Washington, DC).

Heylighen, F., Selfish memes and the evolution of cooperation. *J. Ideas*, 1992, **2**(4), 77–84.

Holland, J.H., *Adaptation in Natural and Artificial Systems*, 1975 (University of Michigan Press: Ann Arbor, MI).

Hopfield, J., Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natn. Acad. Sci. USA*, 1984, **81**, 3088–3092.

Kennedy, J., The particle swarm: social adaptation of knowledge. *Proceedings of the IEEE International Conference on Evolutionary Computation*, Indianapolis, IN, USA, 1997, pp. 303–308, 1997 (IEEE Service Center: Piscataway, NJ).

Moscato, P., On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concurrent Computation Program 158-79, California Institute of Technology, Pasadena, CA, 1989.

Nelder, J.A. and Mead, R., A simplex method for function minimization. *Comput. J.*, 1965, **7**, 308–313.

Price, W.L., A controlled random search procedure for global optimization, in *Toward Global Optimization 2*, edited by L.C. Dixon and G.P. Szego, pp. 71–84, 1978 (North-Holland: Amsterdam).

Price, W.L., Global optimization by controlled random search. *J. Optimization Theory Applic.*, 1983, **40**, 333–348.

Price, W.L., Global optimization algorithms for a CAD workstation. *J. Optimization Theory Applic.*, 1987, **55**, 133–146.

Reynolds, C.W., Flocks, herds and schools: a distributed behavioral model. *Comput. Graphics*, 1987, **21**(4), 25–34.

Speilberg, K., Richards, E., Smith, B.T., Laporte, G. and Boffey, B.T., *Integer Programming and Network Models*, 2000 (Springer: Berlin).

van Laarhoven, P. and Aarts, E.H.L., *Simulated Annealing: Theory and Applications*, 1987 (Kluwer: Dordrecht).

Wilson, E.O., *Sociobiology: The New Synthesis*, 1975 (Belknap Press: Cambridge, MA).

Appendix A: Experimental test problems details

A.1 *DeJong's test functions* (http://solon.cma.univie.ac.at/~vpk/math/funcs.html)

A.1.1 Fn1: DeJong's F_1 function (three-dimensional paraboloid)

$$F(x) = \sum_{i=1}^{3} x_i^2$$
$$\text{s.t.} \quad -512 \leq x_i \leq 512.$$

This three-dimensional function is a simple, nonlinear, convex, unimodal and symmetric function. This test function is intended to be a performance measure of the general efficiency of an algorithm. The global minimum solution is $F(x) = 0$ at (0, 0, 0).

A.1.2 Fn2: DeJong's F_2 function (Rosenbrock's saddle)

$$F(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$
$$\text{s.t.} \quad -2018 \leq x_1, x_2 \leq 2018.$$

This two-dimensional function can be quite difficult for algorithms that are unable to identify promising search directions with little information. The difficulty lies in transiting a sharp narrow ridge that runs along the top of the saddle in the shape of a parabola. The global minimum solution is $F(x) = 0$ at (1, 1).

A.1.3 Fn3: extended Rosenbrock's function

$$F(x) = \sum_{i=1}^{9} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2$$
$$\text{s.t.} \quad -100 \leq x_i, x_{i+1} \leq 100.$$

This function is similar to Fn2 but tests an algorithm's capability to deal with more variables. The global minimum solution is $F(x) = 0$ at $(x_i = 1, i = 1, \ldots, 9)$.

A.1.4 Fn4: Schaffer's F_6 function (Rastrigin function)

$$F(x) = F(x) = 0.5 - \frac{\sin^2(x_1^2 + x_2^2)^{0.5} - 0.5}{1.0 + 0.001(x_1^2 + x_2^2)^2}$$
$$\text{s.t.} \quad -100 \leq x_1, x_2 \leq 100.$$

Fn4 contains more than 50 local minima in the region of interest. The global maximum solution is $F(x) = 1$ at (0, 0).

A.1.5 Fn5: DeJong's F_5 function (inverted Shekel's foxholes)

$$F(x) = F(x) = \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}$$
$$\text{s.t.} \quad -66 \le x_1, x_2 \le 66,$$

where a_{ij} is the ith coordinate of the jth ordered pair generated by $\{-32, -16, 0, 16, 32\}$ (the ith coordinate of the jth inverted foxhole).

This two-dimensional function contains 25 foxholes of various depths surrounded by a relatively flat surface. Many algorithms become stuck in the first foxhole that they fall into. The global minimum solution is $F(x) = 0.998$ at $(-32, -32)$.

A.2 *General test functions*

A.2.1 Gear problem (Deb and Goyal 1997). A compound gear train is to be designed to achieve a specific gear ratio between the driver and the driven shafts. The objective of the gear problem is to find the number of teeth in each of the four gears so as to minimize the error between the obtained gear ratio and a required gear ratio of 1/6.931. The variables are coded in the range (12, 60). If there are four integer variables, say x_1, x_2, x_3 and x_4, therefore, the formulation can be written as follows:

$$\text{minimize } F(x) = \left(\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4}\right)$$
$$\text{s.t.} \quad -12 \le x_1, x_2, x_3, x_4 \le 60, \quad x_i \text{ integers.}$$

The optimal solution is $F(x) = 10^{-12}$ at $x_1 = 19, x_2 = 16, x_3 = 49$ and $x_4 = 43$.

A.2.2 Cutting stock (Speilberg *et al.* 2000). A lumberyard has a supply of 10 ft boards that are cut into 3 ft, 4 ft and 5 ft boards according to customer demand. The 10 ft boards can be cut in six different sensible patterns as shown in table A1. If a customer orders 50 3 ft boards, 65 4 ft boards and 40 5 ft boards, the question is: how many 10 ft boards need to be cut to achieve this?

Table A1. Patterns for cutting 10 ft boards.

	Number required of the following			
Pattern	3 ft boards	4 ft boards	5 ft boards	Waste
1	3	0	0	1
2	2	1	0	0
3	1	0	1	2
4	0	1	1	1
5	0	2	0	2
6	0	0	2	0

If y^j is the number of 10 ft boards cut according to pattern j, $j = 1, \ldots, 6$, the formulation can be written as follows:

$$\begin{aligned}
&\text{minimize } y_1 + y_2 + y_3 + y_4 + y_5 + y_6\\
&\text{s.t.}\\
&3y_1 + 2y_2 + y_3 \geq 50,\\
&y_2 + y_4 + 2y_5 \geq 65,\\
&y_3 + y_4 + 2y_6 \geq 40,\\
&y_j \in N_0, j = 1, \ldots, 6.
\end{aligned}$$

The optimal solution for cutting a total of 65 10 ft boards is $(y_1, y_2, y_3, y_4, y_5, y_6) = (0, 25, 0, 34, 3, 3)$.

A.2.3 Trim loss (Harjunkoski *et al.* 1998). A design problem similar to the cutting stock problem to minimize losses from a layout problem over eight discrete decisions of which two are binary and six are integer. Denoting the binary variables as b_1 and b_2, and the integers as i_3, i_4, i_5, i_6, i_7 and i_8, the formulation can be written as follows:

$$\begin{aligned}
&\text{minimize } 0.1b_1 + 0.2b_2 + i_3 + i_4\\
&\text{s.t.}\\
&460i_5 + 570i_7 = L = 1900,\\
&460i_6 + 570i_8 = L = 1900,\\
&-460i_5 - 570i_7 = L = -1700,\\
&-460i_6 - 570i_8 = L = -1700,\\
&i_5 + i_7 = L = 5,\\
&i_6 + i_8 = L = 5,\\
&b_1 - i_3 = L = 0,\\
&b_2 - i_4 = L = 0,\\
&-15b_1 + i_3 = L = 0,\\
&15b_2 + i_4 = L = 0,\\
&-(i_3i_5 + i_4i_6) = L = -8,\\
&-(i_3i_7 + i_4i_8) = L = -7,\\
&0 \leq i_3 \leq 15,\\
&0 \leq i_4 \leq 15,\\
&0 \leq i_5 \leq 5,\\
&0 \leq i_6 \leq 5,\\
&0 \leq i_7 \leq 5,\\
&0 \leq i_8 \leq 5,\\
&b_1, b_2 \text{ binary}.
\end{aligned}$$

Table A2. Distances between cities.

	City 1	City 2	City 3	City 4	City 5	City 6
City 1		44	35	18	28	23
City 2	44		38	28	27	42
City 3	35	38		26	14	14
City 4	18	28	26		14	20
City 5	28	27	14	14		15
City 6	23	42	14	20	15	

The optimal solution is $F(x) = 5.30$ at $b_1 = 1$, $b_i = 1$, $i_3 = 3$, $i_4 = 2$, $i_5 = 0$, $i_6 = 4$, $i_7 = 3$ and $i_8 = 0$.

A.2.4 Traveling-salesman problem. (http://www.canb.auug.org.au/~dakin/tspimp.htm) A standard traveling-salesman problem with six cities is considered. The distances between the cities are given in table A2.

As formulated in the optimization routine, the decision variables x_i correspond to the ending journey city from the initiation city i. For example (5, 2, 4, 6, 3, 1) corresponds to the loop shown in figure A1 beginning at city 1.

The minimum traveling distance is 124 units with the city sequence of (city 1–city 6–city 3–city 5–city 2–city 4–city 1) or $x = (6, 3, 5, 2, 4, 1)$.

A.2.5 Simpleton25 and Simpleton50. (http://solon.cma.univie.ac.at/~vpk/math/funcs.html). This simple summation problem is formulated as

$$\text{maximize } f(x) = \sum_{i=1}^{i=n} x_i$$

$$\text{s.t.}$$

$$l \geq x_i \geq u,$$

$$x_i \geq 0,$$

$$x_i \text{ integers.}$$

The lower and upper variable bounds are 0 and 10 respectively. The problem was solved with n equal to 25 and 50.

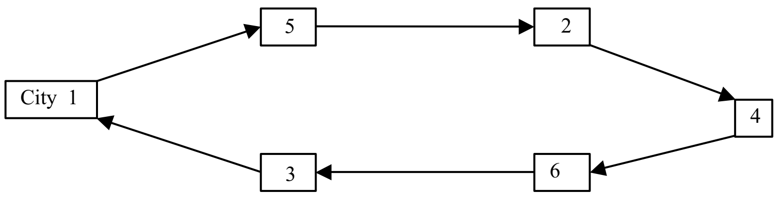


Figure A1. Example loop for the traveling-salesman problem.