International Journal of Operations Research and Information Systems
Volume 14 • Issue 1

A Computational Comparison of Three Nature-Inspired, Population-Based Metaheuristic Algorithms for Modelling-to-Generate Alternatives

Julian Scott Yeomans, York University, Canada*

ABSTRACT

In "real life" decision-making situations, inevitably, there are numerous unmodelled components, not incorporated into the underlying mathematical programming models, that hold substantial influence on the overall acceptability of the solutions calculated. Under such circumstances, it is frequently beneficial to produce a set of dissimilar–yet "good"–alternatives that contribute very different perspectives to the original problems. The approach for creating maximally different solutions is known as modelling-to-generate alternatives (MGA). Recently, a data structure that permits MGA using any population-based solution procedure has been formulated that can efficiently construct sets of maximally different solution alternatives. This new approach permits the production of an overall best solution together with n locally optimal, maximally different alternatives in a single computational run. The efficacy of this novel computational approach is tested on four benchmark optimization problems.

KEYWORDS

Bat Algorithm, Cuckoo Algorithm, Firefly Algorithm, Metaheuristic Algorithms, Modelling-to-Generate Alternatives, Population-Based Algorithms

INTRODUCTION

Multifarious real-world decision-making environments are frequently confounded by ambiguous and incompatible structural specifications that can prove difficult to incorporate into mathematical decision models (Belarbi et al., 2017; Brugnach et al., 2007; Janssen et al., 2010; Matallah et al., 2017; Matthies et al., 2007; Mowrer, 2000; Walker et al., 2003). While "optimal" solutions can normally be calculated for the mathematical formulations, these answers may not produce the best outcomes in the original real system (Acharjya & Anitha, 2017; Brugnach et al., 2007; Fahad et al., 2017; Janssen et al., 2010; Loughlin et al., 2001). To improve decision-making under such circumstances,

DOI: 10.4018/IJORIS.321119 *Corresponding Author

it is often preferable to create a limited number of dissimilar options that contribute very different perspectives (Matthies et al., 2007; Puri et al., 2020; Yeomans & Gunalay, 2011). Preferably these alternatives should all possess good (i.e. near-optimal) objective measures with respect to their modelled objective(s), but be maximally different from each other in terms of the system structures characterized by their decision variables. Several approaches collectively referred to as modelling-to-generate-alternatives (MGA) have been developed in response to this multi-solution creation requirement (Brill et al., 1982; Loughlin et al., 2001; Yeomans & Gunalay, 2011).

The primary impetus behind modelling-to-generate-alternatives (MGA) is to create a manageably small set of alternatives that are good with respect to all measured objective(s) yet are as fundamentally different as possible from each other within the prescribed decision space. By adopting a maximally different approach, the resultant alternative solution set is likely to provide very different perspectives with respect to any unmodelled issues, while simultaneously providing different choices that all perform somewhat similarly with respect to the modelled objectives (Walker et al., 2003). Decision-makers must conduct subsequent assessments of the alternatives to ascertain which specific option(s) most closely satisfies their underlying circumstances (Arrais-Castro et al., 2015). Consequently, MGA approaches are necessarily classified as decision support processes rather than as the explicit solution determination methods generally assumed for optimization (see Benatia et al., 2016; Sharma & Virmani, 2017; Strand et al., 2017).

The earliest MGA procedures employed a relatively straightforward approach in which each alternative was incrementally formulated by re-running the solution generation algorithm whenever a new option had to be produced (Baugh et al., 1997; Brill et al., 1982; Loughlin et al., 2001; Yeomans & Gunalay, 2011; Zechman & Ranjithan, 2004). These iterative procedures mimicked the seminal Hop-Skip-Jump (HSJ) MGA approach of Brill et al. (1982) in which, once an initial problem formulation has been optimized, all supplementary alternatives are produced one-by-one. Consequently, these iterative procedures all require n+1 runnings of their respective algorithms to optimize the initial problem followed by the creation of n alternatives (Imanirad & Yeomans, 2013; Imanirad et al., 2012a; Yeomans & Gunalay, 2011). These MGA approaches were subsequently extended to generate sets of maximally different solution alternatives in Yeomans (2018a, 2018b, 2018c), Imanirad and Yeomans (2013), and Imanirad et al. (2012b, 2013a, 2013b, 2013c).

Recently, Gunalay & Yeomans (2019) and Yeomans (2018d, 2019a, 2019b) introduced a data structure that permits both optimization and MGA using any population-based solution procedure. Specifically, this new data-structure-based approach to MGA enables the simultaneous generation of the overall best solution together with an additional set of m-1 locally optimal, maximally different alternatives in a single computational run. Namely, to generate the additional m-1 maximally different solution alternatives, the MGA algorithm would need to run exactly the same number of times that an optimization procedure would need to be run for function optimization purposes alone (i.e. once) irrespective of the value of m (Yeomans 2017a, 2017b). Consequently, this simultaneous procedure could be considered extremely computationally efficient for MGA purposes.

Numerous metaheuristic approaches have been developed for use in a variety of decision-making environments (for some recent examples, see: Gergin et al. 2019; Jain & Yada 2021; Murali et al. 2022; Vasant et al. 2020). For calculation and optimization purposes, Yang (2009, 2010) created three population-based metaheuristics: the Firefly Algorithm (FA), the Bat Algorithm (BA), and the Cuckoo Algorithm (CA). These three nature-inspired procedures have been shown to be more computationally efficient than the more commonly-used enhanced particle swarm, genetic algorithm, and simulated annealing metaheuristic procedures (Cagnina et al., 2008; Gandomi et al., 2011; Yang & Yeomans 2014) and have been applied to an extremely diverse spectrum of problem settings (Acharjee & Chaudhuri 2022; Aggrawal & Anuja 2022; Bangyal et al. 2021; Bharathi 2022; Chandrasekaran & Simon 2014; Garg & Kumar 2021; Gopu & Venkataraman 2021; Pandey & Bannerjee 2021; Rahman et al. 2019; Rautry et al. 2019; Wang & Ji 2021).

In this paper, for the first time, the efficacy of employing the novel population-based MGA data structure approach in conjunction with the FA, BA, and CA is computationally examined using

the highly-nonlinear, well-known benchmark optimization problems: (i) the bivariate Michalewicz function (Cagnina et al., 2008), (ii) the "spring design" problem (Aragon et al., 2010), (iii) a 100-peak multimodal optimization problem (Loughlin et al., 2001), and a constrained optimization problem (Aragon et al. 2010)..

MODELLING TO GENERATE ALTERNATIVES

Optimization techniques have traditionally focused almost exclusively on determining a unique optimal solution to problem formulations with single-objectives or, equivalently, constructing a set of noninferior solutions to problems possessing multiple-objectives (Brill et al., 1982; Janssen et al., 2010; Walker et al., 2003). While these mathematical programming methods may solve the modelled problems strictly as formulated, whether the results truly provide the "best" answer(s) in their corresponding "real world" implementation is far less obvious (Brill et al., 1982; Brugnach et al., 2007; Janssen et al., 2010; Loughlin et al., 2001). The majority of "real" decision environments generally possess several system components that are not incorporated into the underlying modelling during the problem formulation stage (Brugnach et al., 2007; Walker et al., 2003). Inevitably, numerous subjective aspects are neither quantified nor modelled in the mathematical representation. Such subjective omissions generally occur when final decisions must be based not only on modelled objectives, but also on less tangible socio- economic and political stakeholder viewpoints (Yeomans & Gunalay, 2011). Numerous illustrations of these "real world" modelling decision biases and incongruencies can be found in Baugh et al. (1997), Brill et al. (1982), Loughlin et al. (2001) and Zechman and Ranjithan (2004).

When unmodelled components or unquantified objectives are known or suspected to exist, alternate solution approaches become requisite. These methods must not only explore the solution space for noninferior solutions, but also must examine the feasible region for demonstrably inferior solutions to the problem as modelled. To explicitly reiterate this requirement, any search for good alternatives to "real world" problems possessing unmodelled objectives must necessarily explore both the problem's non-inferior solution set and also its corresponding inferior region.

To illustrate a solution search process for a single-objective maximization problem with unmodelled objectives, assume that the mathematically optimal objective value is $Z1^*$ for the solution $\boldsymbol{X}^*$. Further assume that a second, unquantifiable, socio-political, maximization objective $Z2$ also exists. Let the (unknown) two-objective, noninferior solution, $\boldsymbol{X}^c$, correspond to a best compromise solution if both objectives had been known and considered simultaneously. While $\boldsymbol{X}^c$ represents a best solution to the real problem, it would be considered inferior to $\boldsymbol{X}^*$ in the mathematical model because $Z1^c \leq Z1^*$ by definition. Thus, when unquantified elements are brought into a decision-making process, mathematically inferior decisions to the modelled system could, in fact, become optimal for the underlying real problem. Necessarily, if unquantified components and unmodelled objectives exist, alternative solution procedures must not only search the decision space of the modelled problem for noninferior solutions, but concurrently explore the decision region for patently inferior solutions.

Consequently, under such circumstances, the solution process requires the creation of a set of options that are quantifiably good with respect to all modelled objectives yet remain as-different-as-possible from each other within the feasible region. By achieving this maximal difference condition, the resultant set of alternatives contributes diverse perspectives that perform similarly with respect to all modelled objective(s) yet potentially dissimilarly with respect to any unmodelled components. By constructing these as-different-as-possible alternatives, the decision makers can consider desirable aspects within the options that may address the various unmodelled objectives to varying degrees of stakeholder acceptability.

It becomes necessary to formalize the mathematical definition of maximal difference in order to characterize the solution process (Loughlin et al., 2001; Yeomans & Gunalay, 2011). Let the

optimal solution to a mathematical model be **X*** with corresponding objective calculated as **Z*** = F(**X***). The ensuing maximal difference model can be evaluated to produce the alternative, **X**, that is maximally different to **X***:

$$\text{Maximize } \Delta(\boldsymbol{X}, \boldsymbol{X}^*) = \sum_i |X_i - X_i^*| \quad (1)$$

$$\text{Subject to: } \boldsymbol{X} \in D \quad (2)$$

$$|F(\boldsymbol{X}) - \boldsymbol{Z}^*| \leq T \quad (3)$$

where Δ is a difference function (an absolute difference function in this instance), D is the feasible domain of the original mathematical model, and T represents some permissible user-defined target deviation relative to the original optimal value Z^*. In effect, T determines what proportion of the inferior region will be explored in the search for acceptable alternatives. The notion of difference can be extended into a measure between any *set of alternatives* by replacing $\boldsymbol{X}^*$ in the maximal difference model's objective and evaluating the sum (or another desired function) of the pairwise differences between each pair of alternatives – subject to a condition that each alternative falls within the specified tolerance and is feasible. The strengths and drawbacks of numerous alternative difference functions have been considered extensively by Yeomans (2019c, 2019d, 2019e, 2019f, 2019g, 2020a).

The new population-based MGA data structure approach described in the next section generates a pre-determined number of maximally different alternatives by exploiting the population structure within metaheuristic algorithms. The evolution that occurs throughout the population enables the procedure to simultaneously investigate sets of diverse local optima throughout the entire solution space. The survival of each solution in the population depends upon how it performs both with respect to the original model's objective(s) as well as by the distance each alternative is from all of the other generated alternatives in the feasible region.

POPULATION-BASED SIMULTANEOUS MGA COMPUTATIONAL ALGORITHM

This section provides a synopsis of the state-of-the-art in population-based MGA research. The most logical initial MGA algorithmic approach to create each alternative iteratively increments the target, T, by some amount and then solves the resulting maximum difference model (Yeomans, 2018c, 2018d). Such a one-at-a-time method is directly analogous to the HSJ approach of Brill et al. (1982). In HSJ, once an initial optimization has been completed, additional alternatives are produced sequentially by incrementing the target constraint to force the supplementary construction of suboptimal solutions. This straightforward approach necessitates n subsequent executions of the principal optimization routine to produce the n alternatives (Imanirad & Yeomans, 2013; Imanirad et al., 2012a; Yeomans & Gunalay, 2011).

To improve upon the step-by-step requirement in HSJ, Imanirad et al. (2012a, 2012b, 2013b) used co-evolution to create a concurrent MGA solution technique. In concurrent co-evolution, the overall population of a metaheuristic is stratified into pre-specified subpopulations that permit the search procedure to evolve collectively into a number of maximally different solutions. Each alternative is represented by a specific subpopulation. Evolutionary survival within a subpopulation depends upon both solution quality with respect to the modelled objective(s) and by how far away each alternative (i.e. subpopulation) is from all of the other alternatives (i.e. the other sub-populations). Evolution into a local optimum in any one subpopulation is directly influenced by the set of existing solutions in every other subpopulation. It is this concurrent influence that forces each subpopulation to co-evolve into maximally distant regions of the feasible region (Yeomans & Gunalay, 2011). Because co-evolution concurrently generates an entire set of maximally different alternatives from the single sub-divided population, this population-based procedure is much more computationally efficient than its earlier sequential HSJ counterpart (Imanirad & Yeomans, 2013; Imanirad et al., 2013b).

Although concurrent MGA procedures exploit the structure of population-based algorithms, the actual co-evolution, itself, only occurs within each stratified subpopulation. Unfortunately, this implies that any maximal differences measured between solutions in different subpopulations can only be based upon aggregated subpopulation values.

To counteract this aggregation shortcoming, Yeomans (2018d, 2019a, 2019b) and Gunalay & Yeomans (2019) created a novel data structure that could be used by any population-based metaheuristic for multicriteria MGA. In the ensuing MGA algorithm, each solution in the population represents exactly one potential set of alternatives and the maximal difference is calculated only within that particular solution. Consequently, by the evolutionary nature of population-based searches, the maximal difference will be simultaneously calculated over the specific alternatives within each specific solution – which circumvents any need for concurrent subpopulation aggregation measures.

Suppose that decision-makers wish to generate P different solution alternatives in which each solution possesses n decision variables. Assume that a population-based metaheuristic is to be used and that the population algorithm will contain K solutions, in total. To accomplish the MGA requirement, each solution in the population is designed to contain P maximally different alternatives. Assume that $\boldsymbol{Y}_k$, $k = 1,\ldots, K$, corresponds to the k^{th} solution, so that $\boldsymbol{Y}_k$ provides one complete set of P different alternatives. If $\boldsymbol{X}_{kp}$ represents the p^{th} alternative, $p = 1,\ldots, P$, of solution k, $k = 1,\ldots, K$, then $\boldsymbol{Y}_k$ can be written as:

$$\boldsymbol{Y}_k = [\boldsymbol{X}_{k1}, \boldsymbol{X}_{k2},\ldots, \boldsymbol{X}_{kP}] . \tag{4}$$

If X_{kjq}, $q = 1, \ldots, n$ is the q^{th} variable in the j^{th} alternative of solution k, then:

$$\boldsymbol{X}_{kj} = (X_{kj1}, X_{kj2},\ldots, X_{kjn}) . \tag{5}$$

The entire population, $\boldsymbol{Y}$, in vector form consists of K different sets of P alternatives expressed as:

$$\boldsymbol{Y'} = [\boldsymbol{Y}_1, \boldsymbol{Y}_2,\ldots, \boldsymbol{Y}_K] . \tag{6}$$

Using this data structure, Yeomans (2018d, 2019a, 2019b) and Gunalay and Yeomans (2019) constructed the subsequent simultaneous MGA method that can be easily modified for solution via any population-based algorithm. The approach proceeds by creating its pre-specified number of maximally different alternatives, by adjusting the value of the bound T in the maximal difference model, and by solving the corresponding maximal difference problem. Due to the format of the data structure noted above, each and every solution within the population corresponds to exactly one set of P distinct alternatives. Based upon the evolutionary process within the population, the procedure collectively evolves each solution toward different local optima within the solution space. Solution survival is contingent upon both how the alternative performs with respect to modelled objective(s) and how far apart they are in the decision space from all other generated alternatives.

The steps in this simultaneous MGA algorithm are as follows (Gunalay & Yeomans, 2019; Yeomans, 2018d, 2019a, 2019b):

Preliminary Step. Determine $\boldsymbol{X}^*$, the best solution to the original problem formulation. P corresponds to the required number of maximally different alternatives to be generated within a prescribed target deviation from $\boldsymbol{X}^*$. Note that the value of P is fixed by the decision maker, *a priori*. Create P target values based on the calculated objective value $F(\boldsymbol{X}^*)$. Without loss of generality, this preliminary step can be ignored. The algorithm could solve for $\boldsymbol{X}^*$ in conjunction with the subsequent steps. If this were the case, the initial algorithmic stages would be devoted to the search for $\boldsymbol{X}^*$. This would render the remaining components within each solution as "computational overhead." Consequently, the number of computational iterations would have to increase.

Step 1. Generate the initial population of size K with each solution partitioned into P equally sized divisions. Each partition size represents the number of decision variables in the original problem formulation. $\boldsymbol{X}_{kp}$ corresponds to the p^{th} alternative, $p = 1, \ldots, P$ in solution $\boldsymbol{Y}_k$, $k = 1, \ldots, K$.

Step 2. In each of the K solutions, evaluate each $\boldsymbol{X}_{kp}$, $p = 1, \ldots, P$, using the modelled objective. Alternatives that satisfy both their target constraint and all the other problem constraints are deemed to be *feasible*, whereas all remaining alternatives are designated *infeasible*. Each individual solution in the population would only be designated feasible if every alternative contained within it was feasible.

Note: If the best-solution-found-so-far is always retained over each iteration, at least one feasible solution will always exist. At the very least, one feasible solution can always be constructed by using P repetitions of the solution $\boldsymbol{X}^*$ found in the initialization step.

Step 3. Implement an appropriate elitism operator to rank order the best solutions within the population. The best solution will be the feasible solution containing the most distant set of alternatives within the feasible region (distance measures will be declared in Step 5).

Step 4. If the designated termination criterion has been met (e.g., maximum number of iterations, solution convergence, etc.), stop the algorithm. Otherwise, proceed to Step 5.

Step 5. For each solution $\boldsymbol{Y}_k$, $k = 1, \ldots, K$, calculate the distance measure, $D^I{}_k$, between all of the alternatives contained within it.

One example of an appropriate distance measure is calculated as:

$$D^I{}_k = \Delta^1(\boldsymbol{X}_{ka}, \boldsymbol{X}_{kb}) = \sum_{a=1 to P} \quad \sum_{b=1 to P} \quad \sum_{q=1 \ldots n} \quad | X_{kaq} - X_{kbq} | . \tag{7}$$

This measure represents the total absolute pairwise distance between every alternative contained within solution k. Alternative distance measures could be determined by some other appropriately defined function. Any measure could be appropriately modified to incorporate violation penalties corresponding to infeasibility within the solutions.

Step 6. Rank order all solutions based upon $D^I{}_k$. The goal of maximal difference is to force the alternatives within each solution as far apart as possible. This step orders the specific solutions based upon those solutions which contain alternatives that are most distant from each other in the decision space.

Step 7. Implement the required evolutionary "change operations" to each solution in the population and return to Step 2.

FIREFLY ALGORITHM

Although this section provides a brief overview of the FA, more extensive descriptions can be found in Yang (2009, 2010) and Yang and Yeomans (2014). The FA is a population-based metaheuristic that possesses three ideals inspired by the natural world: (1) All fireflies within a population are considered unisex, implying that any one firefly could be attracted to any other firefly regardless of their sex; (2) Relative attractiveness between any two fireflies is directly proportional to their brightness, meaning that a less brightly flashing firefly will always move toward a brighter one. Furthermore, relative brightness and attractiveness will both decrease as the distance between them increases. If there is no brighter firefly visible within its vicinity, then that particular firefly moves about randomly; and (3) A firefly's brightness is based upon the value of the objective function. Specifically, for any maximization problem, the brightness is considered to be directly proportional to the objective function value. Based upon these ideals, Yang (2010) used the based following pseudo-code to summarize the FA's basic operational steps.

Objective Function $F(\boldsymbol{X})$, $\boldsymbol{X} = (x_1, x_2, \ldots x_d)$
Generate the initial population of n fireflies, $\boldsymbol{X}_i$, $i = 1, 2, \ldots, n$
Light intensity I_i at $\boldsymbol{X}_i$ is determined by $F(\boldsymbol{X}_i)$

```
Define the light absorption coefficient γ
while (t < Max number of generations) or (Stop criterion)
  for i = 1: n, all n fireflies
          for j = 1: n, all n fireflies (inner loop)
                  if (I_i < I_j), Move firefly i towards j; end if
                  Vary attractiveness with distance r via e^(-γr)
          end for j
  end for i
  Rank the fireflies and find the current global best solution G*
end while
  Postprocess the results
```

There are two important issues to resolve when implementing the FA: (i) the variation of light intensity and (ii) the formulation of attractiveness. For simplicity, it is assumed that any firefly's attractiveness depends upon its brightness which, in turn, depends upon the calculated objective function value. For the basic situation, firefly brightness at location $\boldsymbol{X}$ is calculated as its objective value $F(\boldsymbol{X})$. However, the attractiveness, β, between fireflies is relative and varies based upon the distance r_{ij} between firefly i and firefly j. Additionally, because the light intensity decreases with the distance from its source and the light can also be absorbed by the surrounding media, the attractiveness is allowed to vary based upon the degree of absorption. Consequently, firefly attractiveness is calculated as:

$$\beta = \beta_0 \exp(-\gamma r^2) \tag{8}$$

where β_0 represents attractiveness at distance $r = 0$, while γ represents the light absorption coefficient in a particular medium. If the distance r_{ij} between fireflies i and j located at $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, respectively, is calculated using a Euclidean norm, then the movement of the firefly i that is attracted to the more attractive (i.e. brighter) firefly j is calculated as:

$$\boldsymbol{X}_i = \boldsymbol{X}_i + \beta_0 \exp(-\gamma (r_{ij})^2)(\boldsymbol{X}_i - \boldsymbol{X}_j) + \alpha \boldsymbol{\varepsilon}_i . \tag{9}$$

In this movement expression, the second term represents relative attraction and the third component introduces an element of randomness. Yang (2009, 2010) asserts that α is a randomization parameter generally selected within the [0,1] range and $\boldsymbol{\varepsilon}_i$ represents a vector of random numbers drawn from either the uniform (generally [-0.5,0.5]) or Gaussian distributions. This expression represents a random walk biased towards brighter fireflies, becoming a simple random walk when $\beta_0 = 0$. The parameter γ embodies variation in attractiveness and controls the overall speed of convergence for the FA. In general, γ is fixed at a value between 0.1 to 10 (Gandomi et al., 2011; Yang, 2010). In an optimization problem containing a very large number of fireflies $n >> k$, where k is the number of local optima, the initial locations for the n fireflies ought to be distributed uniformly throughout the search space. As the FA proceeds, the fireflies tend to converge into all k local optima (Yang, 2009, 2010). Under such circumstances, the global optima can be easily determined by identifying the best solution(s) among all these optima. Yang (2010) demonstrated that the FA will approach the global optima whenever $n \to \infty$ and the number of iterations t, is set so that $t >> 1$. In practice, the FA has been found to converge extremely quickly (Gandomi et al., 2011; Yang, 2009, 2010).

Two important asymptotic cases occur for $\gamma \to 0$ and $\gamma \to \infty$. When $\gamma \to 0$, the attractiveness remains as the constant $\beta = \beta_0$, which is equivalent to light intensity that does not decrease. This implies that a firefly always remains visible from anywhere in the solution domain. Hence, a single (usually global) optima can easily be reached. If $\boldsymbol{X}_j$ is replaced by the current global best solution, $\boldsymbol{G}^*$, then the FA becomes a special instance of the accelerated particle swarm optimization (PSO) algorithm. The computational efficiency of this special case is equivalent to enhanced PSO. Conversely,

whenever $\gamma \to \infty$, relative attractiveness becomes zero with respect to the view between any firefly pair. This corresponds to the situation in which fireflies simply roam through an impenetrably thick region of fog. Furthermore, each firefly must roam in a completely random fashion because no other fireflies are ever visible and corresponds to a random search method. Because the FA generally operates between these asymptotic extremes, it becomes possible to adjust α and γ so that the FA outperforms both enhanced PSO algorithms and random searches (Gandomi et al., 2011).

What differentiates the FA from other population-based metaheuristics, is that it converges simultaneously into a specified number of local optima (including the global ones) in highly non-linear optimization problems (see, also: Arun et al., 2017; Dekhici et al., 2015, Yeomans 2020a, 2020b). As noted, within the two asymptotic extremes, the population in the FA can concurrently determine both global optima as well as local optima. With a judicious selection of parameter settings, the FA can simultaneously converge extremely quickly into both local and global optima (Gandomi et al., 2011; Yang, 2009, 2010; Yang & Yeomans 2014). Imanirad & Yeomans (2013) have illustrated how the FA's abilities to find multiple local optima can be modified to produce n maximally different alternatives as required for MGA. Concurrent population-based procedures have been shown to possess significant computational efficiency for MGA purposes (Yeomans & Gunalay, 2011). Consequently, an additional advantage in using an FA for MGA is that, because the different fireflies will independently tend to aggregate more closely around each local optimum, the FA procedures prove far better than genetic algorithms and PSO for MGA (Gandomi et al., 2011; Yang, 2010).

BAT ALGORITHM

This section provides a concise synopsis of the population-based BA metaheuristic that is covered in significantly greater detail in Yang (2009, 2010) and Yeomans (2021). Each bat in the population corresponds to one potential solution to a problem and the initial population of fireflies is distributed randomly and uniformly throughout the decision space. The BA operates under the following three ideals: (i) All bats use echolocation to sense distance and can distinguish between food/prey and background barriers; (ii) Bats fly randomly with velocity v_i at position x_i with a fixed frequency f_{min} (or wavelength λ), varying wavelength λ (or frequency f) and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r within the range [0,1] depending upon the proximity to their target; and, (iii) Although the loudness can vary in many ways, it can be assumed that loudness actually varies from a large (positive) A_0 down to some minimum value A_{min}. The operational steps of the BA are summarized in the following pseudo-code (Yang 2010).

```
Objective Function F(X), X = (x1, x2,… xd)
Initialize population of n bats, Xi, i = 1, 2,…, n and vi
Initialize pulse rates ri and the loudness Ai.
Define the pulse frequency fi at Xi.
while (t < Max number of generations) or (Stop criterion)
Generate new solutions by adjusting frequency, and updating
velocities and locations/solutions
  if (rand > ri)
          Select a solution among the best solutions
          Generate a local solution around the selected best
solution
  end if
  Generate a new solution by flying randomly
  if (rand < Ai) & (F(Xi) < F(X*))
          Accept the new solutions
          Increase ri and reduce Ai
```

```
  end if
Rank the bats and find the current best solution x*
end while
  Postprocess the results and generate any visualizations
```

In the BA, the virtual bats are simulated. Certain rules are adopted to define how their positions x_i and velocities v_i in the d-dimensional search space are updated. The solutions/positions x_{it} and velocities v_{it} at time step t are determined by:

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta \tag{10}$$
$$v_{it} = v_{it-1} + (x_{iy} - x_*)\,f_i \tag{11}$$
$$x_{it} = x_{it-1} + v_{it} \tag{12}$$

where β is a random vector with each element generated from a uniform distribution over the range [0,1]. The value x_* is the current global best solution which is determined by comparing all the solutions among the n bats in the n-dimensional solution vector $\boldsymbol{x}$. Initially each bat is assigned a random frequency drawn uniformly from the interval $[f_{min}, f_{max}]$. For the local search portion, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using a random walk:

$$\boldsymbol{x}_{new} = \boldsymbol{x}_{old} + \boldsymbol{\varepsilon} A_t \tag{13}$$

where $\boldsymbol{\varepsilon}$ is a random vector in the range [-1,1] and A_t is the average loudness of all the bats at this time step, t.

The loudness A_i and the rate of pulse emissions, r_i, have to be updated accordingly as the iterations proceed. As the loudness usually decreases once the bat has found its prey while the rate of pulse emissions increases, the loudness can be chosen as any value of convenience. For simplicity, one can use $A_0 = 1$ and $A_{min} = 0$, assuming that $A_{min} = 0$ implies that a bat has just found the prey and temporarily stops emitting any sound. Thus:

$$A_{it+1} = \alpha\, A_{it},\ r_{it+1} = r_{i0}\,[1 - \exp(\gamma\, t)] \tag{14}$$

where α and γ are constants. The choice of parameters requires some experimenting, but in the simplest case $\alpha = \gamma$. Initially, each bat should have different values for their loudness and emissions rate, and this can be achieved via randomization. The loudness and emissions rates will only be updated by the algorithm if the new solutions provide an improvement, which means that the bats are actively moving towards the optimal solution. The algorithm proceeds either until some convergence condition has been achieved or for a maximum number of iterations (Gandomi *et al*., 2011; Yang, 2009, 2010).

CUCKOO ALGORITHM

This section provides a very brief outline of the CA procedure that is covered in significantly greater detail in Yang (2009, 2010) and Yang and Deb (2010). The CA is a nature-inspired, population-based metaheuristic in which each cuckoo egg represents one possible solution. The CA operates under the following three idealized rules: (i) Each cuckoo lays a single egg at a time and deposits it in a randomly chosen nest; (ii) The best nests with highest quality of eggs (solutions) will carry over to the next generations; and, (iii) The number of available host nests is fixed, and a host bird can discover an alien egg with probability P_a within the range [0,1]. In the case of detection, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.

For simplicity, this last assumption can be approximated by a fraction P_a of the n nests being replaced by new nests (with new random solutions provided at each new location). For a maximization problem, the quality or fitness of a solution is simply proportional to the objective function value. Alternative forms of fitness could also be defined in analogous fashion.

The operational steps of the BA can be summarized in the following pseudo-code (Yang 2010):

```
Objective Function F(X), X = (x_1, x_2,... x_d)
Initialize population of n host nests, X_i, i = 1, 2,..., n
while (t < Max number of generations) or (Stop criterion)
  Randomly select a cuckoo (say i) and generate a new solution by
Levy flights
  Evaluate its quality/fitness, F_i
  Randomly choose a nest among n (say j)
  if (F_i > F_j)
          Replace j by the new solution
  end if
  Abandon a fraction (P_a) of worse nests and build new ones at new
locations via Levy flights
  Build new nests at new locations
  Keep the best solutions (or nests with quality solutions)
  Rank the solutions and find the current best solutions
end while
  Postprocess the results and generate any visualizations
When generating new solutions X_i^(t+1) for cuckoo i, a Levy flight is
performed:
```

$$\boldsymbol{X}_i^{(t+1)} = \boldsymbol{X}_i^{(t)} + \alpha \oplus \text{Levy}(\lambda) \tag{15}$$

where $\alpha > 0$ is a step size related to the scale of the problem. In most cases, $\alpha = O(1)$ can be employed and the product $\oplus$ implies entry-wise multiplications. Levy flights essentially provide a random walk, while their random steps are drawn from a Levy distribution for large steps:

$$\text{Levy} \sim u = t^{\lambda},\ 1 < \lambda \leq 3 \tag{16}$$

which has an infinite variance with an infinite mean. Here, the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail (see Yang, 2010).

COMPUTATIONAL TESTING OF THE MGA ALGORITHM

As outlined earlier, "real world" decision-makers often prefer to be able to choose from a set of close-to-optimal alternatives that significantly differ from each other in terms of the system structures characterized by their decision variables. The effectiveness of the novel MGA data structure procedure introduced in the previous section to simultaneously produce maximally different alternatives will be evaluated against four frequently-tested, well-known benchmark problems. In each test, the target constraint is set so that all of the alternatives produced can fall within, at worst, 10% of the best objective value determined.

The first computational test uses the multimodal bivariate Michalewicz function problem taken from Yang and Deb (2010). The mathematical formulation for the tested instance of the bivariate Michalewicz function is:

Table 1.
Five maximally different alternatives for the Michalewicz function generated by each algorithm

FIREFLY ALGORITHM	*F*(*x*,*y*)	*x*	*y*
Optimal found in Preliminary Step	-1.80	2.20	1.57
Alternative 1	-1.78	2.23	1.58
Alternative 2	-1.73	2.26	1.56
Alternative 3	-1.65	2.30	1.56
Alternative 4	-1.58	2.08	1.56
BAT ALGORITHM	***F*(*x*,*y*)**	***x***	***y***
Optimal found in Preliminary Step	-1.80	2.20	1.57
Alternative 1	-1.76	2.16	1.55
Alternative 2	-1.72	2.26	1.59
Alternative 3	-1.63	2.31	1.56
Alternative 4	-1.62	2.31	1.58
CUCKOO ALGORITHM	***F*(*x*,*y*)**	***x***	***y***
Optimal found in Preliminary Step	-1.80	2.20	1.57
Alternative 1	-1.76	2.15	1.56
Alternative 2	-1.65	2.10	1.56
Alternative 3	-1.63	2.31	1.56
Alternative 4	-1.58	2.08	1.56

$$\text{Minimize } F(x, y) = -\sin(x)\sin^{2m}\left(\frac{x^2}{\pi}\right) - \sin(y)\sin^{2m}\left(\frac{2y^2}{\pi}\right) \tag{17}$$

$$0.0 \leq x \leq 5.0\ 0.0 \leq y \leq 5.0\ m = 10 \tag{18}$$

The highly non-linear feasible region of this problem contains a considerable number of peaks (local optima) separated by a proportionately large number of valleys. For the design parameters employed in this specific instance of the formulation, the best solution of $F(x, y) =$ -1.8013 occurs at the point $(x, y) = (2.20319, 1.57049)$ (Yang & Deb, 2010). The MGA approach outlined in the previous section was run to produce five maximally different solutions shown in Table 1. As can be observed, the best solutions determined by each algorithm, respectively, match the true optimal solution for the Michalewicz formulation. Furthermore, the remaining four alternatives created by each algorithm all composed of a similar range of solution instances.

The second application of the MGA procedure is illustrated using the spring design problem taken from Cagnina et al. (2008). The design of a tension and compression spring has frequently been employed as a benchmark problem for constrained engineering optimization problems (Aragon et al., 2010; Cagnina et al., 2008). The spring problem involves three design variables: (1) x_1, the wire diameter; (2) x_2, the coil diameter; and (3) x_3, the length of the coil. The aim is to essentially minimize the weight subject to constraints on deflection, stress, surge frequency, and geometry. The mathematical formulation for this test problem can be summarized as:

$$\text{Minimize } \mathrm{F}(\boldsymbol{X}) = x_1^2 x_2 \left(2 + x_3\right) \tag{19}$$

$$\text{Subject to: } g_1(\boldsymbol{X}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \tag{20}$$

$$g_2(\boldsymbol{X}) = \frac{4x_2^2 - x_1 x_2}{12566\left(x_1^3 x_2 - x_1^4\right)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \tag{21}$$

$$g_3(\boldsymbol{X}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \tag{22}$$

$$g_4(\boldsymbol{X}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \tag{23}$$

$$0.05 \leq x_1 \leq 2.0 \; 0.25 \leq x_2 \leq 1.3 \; 2.0 \leq x_3 \leq 15.0 \tag{24}$$

The optimal solution for the specific design parameters employed within this formulation is $F(\boldsymbol{X}^*) = 0.0127$ with decision variable values of $\boldsymbol{X}^* = (0.051690, 0.356750, 11.287126)$ (Cagnina et al., 2008). The MGA procedure was used to create the five maximally different solutions shown for each algorithm in Table 2. As with the Michalewicz problem, the best spring design solutions found for each algorithm are all identical to the actual optimal solution. As with the previous test, all four of the remaining alternatives are comprised of a very similar set of solution instances.

The third MGA solution creation application will be tested on the 100-peak multimodal optimization problem taken from Loughlin et al. (2001). The mathematical formulation for this multimodal test problem is:

$$\text{Maximize } F(x, y) = \text{Sin}(19\pi x) + \frac{x}{1.7} + \text{Sin}(19\pi y) + \frac{y}{1.7} + 2 \tag{25}$$

$$0.0 \leq x \leq 1.0 \tag{26}$$

$$0.0 \leq y \leq 1.0 \tag{27}$$

The corresponding feasible region for this highly-nonlinear problem contains 100 peaks separated by valleys with the amplitudes of both the peaks and valleys increasing as the values of the decision variables increase from their lower bounds of (0,0) toward their upper limits at (1,1). For the design parameters employed in this specific problem formulation, the mathematically optimal solution of $F(x, y) = 5.146$ occurs at point $(x, y) = (0.974, 0.974)$ (Loughlin et al. 2001). The MGA difference model was used to generate the 5 maximally different solutions shown in Table 3. Similar to the two previous tests, the best solutions found are all identical to the true optimal solution for each algorithm. Similarly, the remaining four solutions in each of the sets of alternatives all appear to be entirely congruent with one another.

The fourth testing of the MGA procedure will be on the commonly evaluated constrained optimization problem from Aragon et al. (2010). The mathematical formulation for this test problem can be summarized as:

$$\text{Min } F(\boldsymbol{X}) = \left(x_1 - 10\right)^2 + 5\left(x_2 - 12\right)^2 + x_3^4 + 3\left(x_4 - 11\right)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6 x_7 - 10x_6 - 8x_7 \tag{28}$$

Subject to:

$$g_1(\boldsymbol{X}) = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \leq 0 \tag{29}$$

$$g_2(\boldsymbol{X}) = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \leq 0 \tag{30}$$

$$g_3(\boldsymbol{X}) = 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \leq 0 \tag{31}$$

Table 2.
Five maximally different alternatives for the spring design problem generated by each algorithm

FIREFLY ALGORITHM	F(*X*)	x_1	x_2	x_3
Optimal found in Prelim. Step	0.0127	0.0517	0.3567	11.2871
Alternative 1	0.0128	0.0500	0.3164	14.1754
Alternative 2	0.0131	0.0500	0.3129	14.777
Alternative 3	0.0138	0.0523	0.348	13.3247
Alternative 4	0.0140	0.0535	0.3857	14.162
BAT ALGORITHM	**F(*X*)**	x_1	x_2	x_3
Optimal found in Prelim. Step	0.0127	0.0517	0. 3567	11.2871
Alternative 1	0.0128	0.0500	0.3165	14.1598
Alternative 2	0.0130	0.0521	0.3656	11.0667
Alternative 3	0.0132	0.0500	0.3167	14.6402
Alternative 4	0.0138	0.0523	0.348	13.3247
CUCKOO ALGORITHM	**F(*X*)**	x_1	x_2	x_3
Optimal found in Prelim. Step	0.0127	0.0517	0.3567	11.2871
Alternative 1	0.0128	0.0514	0.3472	12.0089
Alternative 2	0.0129	0.0529	0.3862	9.9684
Alternative 3	0.0137	0.0520	0.3629	12.1615
Alternative 4	0.0140	0.0557	0.4307	9.5783

$$g_4(\boldsymbol{X}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 12x_7 \leq 0 \quad (32)$$

$$-10 \leq x_i \leq 10,\ i = 1, 2, 3, 4, 5, 6, 7 \quad (33)$$

The optimal solution for the specific design parameters employed within this formulation is F(***X****) = 680.6300573 with decision variable values of ***X**** = (2.330499, 1.951372, -0.4775414, 4.365726, 0.6244870, 1.038131, 1.594227) (Aragon et al., 2010). The MGA procedure was run to create the five maximally different solutions shown in Table 4. In line with the observations in the preceding experimentation, the best solutions determined by each metaheuristic are identical to the true optimal solution. Likewise, the sets of alternatives are all correspondingly very similar in structure to one another.

Taken together, the computational examples have illustrated how the new MGA data structure modelling can be used to simultaneously create multiple different alternatives by employing the various computationally efficient, population-based metaheuristics FA, BA, and CA. All options generated by the data structure MGA algorithm satisfy the requisite system criteria to within the pre-specified, 10% bound while remaining maximally different from each other within the decision space. In addition to their alternative creation proficiencies, the population-based aspect of each of

Table 3.
Five maximally different alternatives for the 100-peak multimodal optimization problem generated by each algorithm

FIREFLY ALGORITHM	$F(x,y)$	x	y
Optimal found in Prelim. Step	5.14	0.97	0.97
Alternative 1	5.01	0.87	0.87
Alternative 2	5.00	0.76	0.98
Alternative 3	4.89	0.55	0.97
Alternative 4	4.65	0.33	0.97
BAT ALGORITHM	$F(x,y)$	x	y
Optimal found in Prelim. Step	5.14	0.97	0.97
Alternative 1	5.05	0.87	0.98
Alternative 2	4.99	0.98	0.87
Alternative 3	4.74	0.34	0.98
Alternative 4	4.69	0.98	0.24
CUCKOO ALGORITHM	$F(x,y)$	x	y
Optimal found in Prelim. Step	5.14	0.97	0.97
Alternative 1	5.10	0.98	0.97
Alternative 2	4.89	0.66	0.87
Alternative 3	4.77	0.87	0.45
Alternative 4	4.64	0.13	0.98

the metaheuristics simultaneously enables the MGA data structure algorithm to perform extremely well in terms of function optimization. It has been explicitly noted in Tables 1-4 that the best solutions determined by the MGA algorithm in each test instance, using each of the FA, BA, or CA, is always completely identical to the true optimal solutions found in Yang and Deb (2010), Cagnina et al. (2008), Loughlin et al. (2001), and Aragon et al. (2010), respectively.

In summary, this section has demonstrated the efficacy of employing the population-based MGA data structure procedure in conjunction with the FA, BA, and CA by testing it on four well-known benchmark problems. There are several key findings. First, the FA, BA, and CA are all excellent optimization methods. They always enabled the MGA procedure to determine the optimal solution to each of the benchmark problems in every instance. Second, in general, the evolutionary features operating within each of the FA, BA, and CA actually create more good alternatives than planners would be able to produce using other MGA techniques. This is due to the underlying evolving characteristics of population-based searches. Third, due to the inherent design of the MGA algorithm, the alternatives generated would be intrinsically suitable for planning purposes. Their decision-variable structures are simultaneously maximally different *from each other* while falling within the user-specified bound of overall optimality (not just different from the optimal solution as in HSJ). Fourth, for *each* solution set for each test problem, *irrespective* of the actual population-based procedure employed, the alternatives within these sets are essentially identical in terms of range and distribution of decision-variable structures together with the similarity of their corresponding objectives. Fifth, the population-based MGA data structure algorithm is computationally efficient because it requires a single run of whatever population-based metaheuristic procedure is employed to generate its entire set of alternatives (not via multiple runs as under an HSJ-styled approach). Specifically, in order to

Table 4.
Five maximally different alternatives for the constrained optimization problem generated by each algorithm

FIREFLY ALGORITHM	**F(*X*)**	x_1	x_2	x_3	x_4	x_5	x_6	x_7
Optimal from Prelim. Step	680.630	2.3304	1.9513	-0.4775	4.3657	0.6244	1.0381	1.5942
Alternative 1	687.580	2.2892	1.8985	-0.4605	4.3364	-0.5962	1.0208	1.5782
Alternative 2	706.837	2.2913	1.9003	-0.3965	4.3548	-0.6388	1.0796	1.6023
Alternative 3	718.478	2.2904	1.9037	-0.427	4.3637	-0.5871	0.9955	1.6230
Alternative 4	744.901	2.3468	1.9118	-0.4087	4.3557	-0.6283	0.9899	1.6024
BAT ALGORITHM	**F(*X*)**	x_1	x_2	x_3	x_4	x_5	x_6	x_7
Optimal from Prelim. Step	680.630	2.3304	1.9513	-0.4775	4.3657	0.6244	1.0381	1.5942
Alternative 1	683.917	2.3025	1.9353	-0.4881	4.3333	-0.6169	1.0355	1.5889
Alternative 2	696.899	2.2934	1.9096	-0.4397	4.3369	-0.6616	1.0331	1.6176
Alternative 3	718.478	2.2904	1.9037	-0.427	4.3637	-0.5871	0.9955	1.6230
Alternative 4	741.897	2.2904	1.9037	-0.427	4.3637	-0.5871	0.9955	1.6230
CUCKOO ALGORITHM	**F(*X*)**	x_1	x_2	x_3	x_4	x_5	x_6	x_7
Optimal from Prelim. Step	680.630	2.3304	1.9513	-0.4775	4.3657	0.6244	1.0381	1.5942
Alternative 1	687.022	2.3056	1.9076	-0.4245	4.3256	-0.6184	1.0388	1.6067
Alternative 2	705.926	2.3080	1.9171	-0.4724	4.3343	-0.6578	1.053	1.6078
Alternative 3	711.793	2.3174	1.9111	-0.4084	4.3668	-0.6166	1.0759	1.6116
Alternative 4	730.091	2.2892	1.8985	-0.4605	4.3364	-0.5962	1.0208	1.5782

generate *P* different solution alternatives, the population-based MGA data structure algorithm needs to run a single time, irrespective of the value of *P*.

As motivated in the earlier sections, many "real world" optimization applications are plagued by incongruent performance requirements that are extraordinarily difficult to quantify. Thus, it is never a disadvantage to construct a set of quantifiably good alternatives that provide distinct perspectives to any potentially unmodelled design issues introduced during the solution formulation stage. The unique performance features incorporated within these dissimilar alternatives may be able to provide very different system performance with respect to the unmodelled issues, possibly incorporating some of the unmodelled issues into the actual solution process. The evaluation testing on each of the benchmark problems clearly demonstrates all of these requisite, desirable characteristics.

CONCLUSION

Complex "real world" decision-making can prove very difficult and is frequently influenced by numerous unquantifiable issues, unmodelled objectives and uncertain factors. These decision environments frequently contain incompatible design specifications that are problematic – if not impossible – to incorporate when ancillary decision support models are constructed. Invariably, there are unmodelled elements, not apparent during model formulation, that can significantly affect solution adequacy. With so much uncertainty, it is doubtful that any single solution could ever be constructed that concurrently fulfills all of the incongruent system requirements. Therefore, any

decision support approach must somehow address these complicating features in some way, while simultaneously being flexible enough to condense the potential effects within the intrinsic planning incongruities. Hence the need for MGA approaches.

In this study, an MGA data structure procedure was tested to demonstrate how three widely-employed, population-based metaheuristics could be employed to simultaneously produce numerous maximally different alternatives. This population-based MGA data structure approach is designed to always generate a set of dissimilar solution options, such that each generated alternative contributes an entirely different perspective to the problem. The set of maximally different alternatives produced by the computationally efficient MGA data structure algorithm supplies contrasting solutions that can introduce very different perspectives to the problem-solving process while still performing robustly with respect to the requisite performance measures. Consequently, these alternatives could potentially capture possibilities completely overlooked under more traditional modelling approaches that can now be incorporated into the decision-making.

The computational effectiveness from employing the MGA algorithm in conjunction with the FA, BA, and CA metaheuristics was demonstrated on four well-known, widely-tested, highly-nonlinear benchmark problems. The MGA computational procedure under each metaheuristic created not only a very similar set of near-best, maximally different alternatives for each tested problem, but also determined the optimal solution in every instance. Hence, the testing on the FA, BA, and CA methods clearly demonstrated that each of these population-based procedures is highly effective for MGA purposes and that the algorithms are, effectively, mutually interchangeable from a computational performance perspective. Furthermore, since population-based metaheuristics can be adapted to a wide spectrum of problem types, the adaptability of this population-based MGA approach can be extended into numerous different problem domains. Extensions to the procedure using other population-based metaheuristic algorithms will be examined in forthcoming studies.

REFERENCES

Acharjee, S., & Chaudhuri, S. S. (2022). Test zone search optimization using cuckoo search algorithm for VVC. *International Journal of Multimedia Data Engineering and Management*, *13*(1), 1–16. doi:10.4018/IJMDEM.314574

Acharjya, D., & Anitha, A. (2017). A comparative study of statistical and rough computing models in predictive data analysis. *International Journal of Ambient Computing and Intelligence*, *8*(2), 32–51. doi:10.4018/IJACI.2017040103

Aggrawal, K., & Anuja, A. (2022). Detecting community structure in financial markets using the bat optimization algorithm. *International Journal of Information Technology Project Management*, *13*(3), 1–21. doi:10.4018/IJITPM.313421

Aragon, V. S., Esquivel, S. C., & Coello, C. C. A. (2010). A modified version of a t-cell algorithm for constrained optimization problems. *International Journal for Numerical Methods in Engineering*, *84*, 351–378. doi:10.1002/nme.2904

Arrais-Castro, A., Varela, M. L. R., Putnik, G. D., Ribeiro, R., & Dargram, F. C. C. (2015). Collaborative negotiation platform using a dynamic multi-criteria decision model. *International Journal of Decision Support System Technology*, *7*(1), 1–14. doi:10.4018/ijdsst.2015010101

Arun, B., & Kumar, V. (2017). Materialized view selection using bumble bee mating optimization. *International Journal of Decision Support System Technology*, *9*(3), 1–27. doi:10.4018/IJDSST.2017070101

Bangyal, W. H., Ahmad, J., & Rauf, H. T. (2021). Comparison of different bat initialization techniques for global optimization problems. *International Journal of Applied Metaheuristic Computing*, *12*(1), 157–184. doi:10.4018/IJAMC.2021010109

Baugh, J. W. Jr, Caldwell, S. C., & Brill, E. D. Jr. (1997). A mathematical programming approach for generating alternative in discrete structural optimization. *Engineering Optimization*, *28*(1), 1–31. doi:10.1080/03052159708941125

Belarbi, M. A., Mahmoudi, S., & Belalem, G. (2017). PCA as dimensionality reduction for large-scale image retrieval systems. *International Journal of Ambient Computing and Intelligence*, *8*(4), 45–58. doi:10.4018/IJACI.2017100104

Benatia, I., Laouar, M. R., Bendjenna, H., & Eom, S. (2016). Implementing a cloud-based decision support system in a private cloud: The infrastructure and the deployment process. *International Journal of Decision Support System Technology*, *8*(1), 25–42. doi:10.4018/IJDSST.2016010102

Bharathi, M. (2022). Hybrid particle swarm and ranked firefly metaheuristic optimization-based software test case minimization. *International Journal of Applied Metaheuristic Computing*, *13*(1), 1–20. doi:10.4018/IJAMC.2022010106

Brill, E. D. Jr, Chang, S. Y., & Hopkins, L. D. (1982). Modelling to generate alternatives: The HSJ approach and an illustration using a problem in land use planning. *Management Science*, *28*(3), 221–235. doi:10.1287/mnsc.28.3.221

Brugnach, M., Tagg, A., Keil, F., & De Lange, W. J. (2007). Uncertainty matters: Computer models at the science-policy interface. *Water Resources Management*, *21*(7), 1075–1090. doi:10.1007/s11269-006-9099-y

Cagnina, L. C., Esquivel, C. A., & Coello, C. A. (2008). Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica (Vilnius)*, *32*, 319–326.

Chandrasekaran, K., & Simon, S. P. (2014). Wind-thermal integrated power system scheduling problem using cuckoo search algorithm. *International Journal of Operations Research and Information Systems*, *5*(3), 1–16. doi:10.4018/ijoris.2014070104

Dekhici, L., & Belkai, K. (2015). A bat algorithm with generalized walk for the two-stage hybrid flow shop problem. *International Journal of Decision Support System Technology*, *7*(3), 1–16. doi:10.4018/IJDSST.2015070101

Fahad, M., Ullah, R., & Abdullah, F. (2017). Big data analysis and implementation in different areas using IoT. *International Journal of Hyperconnectivity and the Internet of Things*, *1*(2), 12–25. doi:10.4018/IJHIoT.2017070102

Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2011). Mixed variable structural optimization using firefly algorithm. *Computers & Structures*, *89*(23-24), 2325–2336. doi:10.1016/j.compstruc.2011.08.002

Garg, D., & Kumar, P. (2021). Accelerated cuckoo search with extended diversification and intensification. *International Journal of Swarm Intelligence Research*, *12*(3), 125–148. doi:10.4018/IJSIR.2021070106

Gergin, Z., Tuncbilek, N., & Esnaf, S. (2019). Clustering approach using artificial bee colony algorithm for healthcare waste disposal facility location problem. *International Journal of Operations Research and Information Systems*, *10*(1), 56–75. doi:10.4018/IJORIS.2019010104

Gopu, A., & Venkataraman, N. (2021). Virtual machine placement using multi-objective bat algorithm with decomposition in distributed cloud: MOBA/D for VMP. *International Journal of Applied Metaheuristic Computing*, *12*(4), 62–77. doi:10.4018/IJAMC.2021100104

Gunalay, G., & Yeomans, J. S. (2019). Multicriteria generation of alternatives for engineering optimization problems using population-based metaheuristics: A computational test. *WSEAS Transactions on Computers*, *18*(31), 239–247.

Imanirad, R., Yang, X. S., & Yeomans, J. S. (2012a). A computationally efficient, biologically-inspired modelling-to-generate-alternatives method. *Journal of Computing*, *2*(2), 43–47.

Imanirad, R., Yang, X. S., & Yeomans, J. S. (2012b). A co-evolutionary, nature-inspired algorithm for the concurrent generation of alternatives. *Journal of Computing*, *2*(3), 101–106.

Imanirad, R., Yang, X. S., & Yeomans, J. S. (2013a). Modelling-to-generate-alternatives via the firefly algorithm. *Journal of Applied Operational Research*, *5*(1), 14–21.

Imanirad, R., Yang, X. S., & Yeomans, J. S. (2013b). A concurrent modelling to generate alternatives approach using firefly algorithm. *International Journal of Decision Support System Technology*, *5*(2), 33–45. doi:10.4018/jdsst.2013040103

Imanirad, R., Yang, X. S., & Yeomans, J. S. (2013c). A biologically-inspired metaheuristic procedure for modelling-to-generate-alternatives. *International Journal of Engineering Research and Applications*, *3*(2), 1677–1686.

Imanirad, R., & Yeomans, J. S. (2013). Modelling to generate alternatives using biologically inspired algorithms. In X. S. Yang (Ed.), *Swarm intelligence and bio-inspired computation: Theory and applications* (pp. 313–333). Elsevier. doi:10.1016/B978-0-12-405163-8.00014-4

Jain, A., & Yada, D. (2021). Particle swarm optimization for Punjabi text summarization. *International Journal of Operations Research and Information Systems*, *12*(3), 1–17. doi:10.4018/IJORIS.20210701.oa1

Janssen, J. A. E. B., Krol, M. S., Schielen, R. M. J., & Hoekstra, A. Y. (2010). The effect of modelling quantified expert knowledge and uncertainty information on model-based decision making. *Environmental Science & Policy*, *13*(3), 229–238. doi:10.1016/j.envsci.2010.03.003

Loughlin, D. H., Ranjithan, S. R., Brill, E. D. Jr, & Baugh, J. W. Jr. (2001). Genetic algorithm approaches for addressing unmodeled objectives in optimization problems. *Engineering Optimization*, *33*(5), 549–569. doi:10.1080/03052150108940933

Matallah, H., Belalem, G., & Bouamrane, K. (2017). A thorough insight into theoretical and practical developments in multi-agent systems. *International Journal of Ambient Computing and Intelligence*, *8*(1), 23–49. doi:10.4018/IJACI.2017010102

Matthies, M., Giupponi, C., & Ostendorf, B. (2007). Environmental decision support systems: Current issues, methods and tools. *Environmental Modelling & Software*, *22*(2), 123–127. doi:10.1016/j.envsoft.2005.09.005

Mowrer, H. T. (2000). Uncertainty in natural resource decision support systems: Sources, interpretation and importance. *Computers and Electronics in Agriculture*, *27*(1-3), 139–154. doi:10.1016/S0168-1699(00)00113-7

Murali, K., & Doma, S. (2022). Artificial intelligence-based breast cancer detection using WPSO. *International Journal of Operations Research and Information Systems*, *13*(2), 1–16. doi:10.4018/IJORIS.306195

Pandey, A., & Bannerjee, S. (2021). Test data generation and selection using Levy flight-based firefly algorithm. *International Journal of Operations Research and Information Systems*, *9*(2), 18–34.

Puri, V., Le, C. V., Kumar, R., & Jagdev, S. S. (2020). Fruitful synergy model of artificial intelligence and Internet of thing for smart transportation system. *International Journal of Hyperconnectivity and the Internet of Things*, *4*(1), 43–57. doi:10.4018/IJHIoT.2020010104

Rahman, R. M., Hasan, S. S., Rahman, R., Jahan, K. A., Islam, S., & Shadman, A. I. (2019). A novel fuzzy inspired bat algorithm for multidimensional function optimization problem. *International Journal of Fuzzy System Applications*, *8*(1), 83–100. doi:10.4018/IJFSA.2019010105

Rautry, R., Balabantaray, R. C., Dash, R. S., & Dash, R. (2019). CSMDSE-Cuckoo search based multi document summary extractor: Cuckoo search based summary extractor. *International Journal of Cognitive Informatics and Natural Intelligence*, *13*(4), 56–70. doi:10.4018/IJCINI.2019100103

Sharma, K., & Virmani, J. (2017). A decision support system for classification of normal and medical renal disease using ultrasound images: A decision support system for medical renal diseases. *International Journal of Ambient Computing and Intelligence*, *8*(2), 52–69. doi:10.4018/IJACI.2017040104

Strand, M., Syberfeldt, A., & Geertsen, A. (2017). A decision support system for sustainable waste collection. *International Journal of Decision Support System Technology*, *9*(4), 49–65. doi:10.4018/IJDSST.2017100104

Vasant, P., Le, L. D., Vo, D. N., Huynh, S. T., & Nguyen-Hoang, T. M. (2020). A hybrid differential evolution and harmony search for optimal power flow with FACTS devices. *International Journal of Operations Research and Information Systems*, *11*(3), 39–65. doi:10.4018/IJORIS.2020070103

Walker, W. E., Harremoes, P., Rotmans, J., Van der Sluis, J. P., Van Asselt, M. B. A., Janssen, P., & Krayer von Krauss, M. P. (2003). Defining uncertainty – a conceptual basis for uncertainty management in model-based decision support. *Integrated Assessment*, *4*(1), 5–17. doi:10.1076/iaij.4.1.5.16466

Wang, J., & Ji, Y. (2021). Firefly algorithm based on Euclidean metric and dimensional mutation. *International Journal of Cognitive Informatics and Natural Intelligence*, *15*(4), 1–19. doi:10.4018/IJCINI.286769

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. *Lecture Notes in Computer Science*, *5792*, 169–178. doi:10.1007/978-3-642-04944-6_14

Yang, X. S. (2010). *Nature-inspired metaheuristic algorithms* (2nd ed.). Luniver Press.

Yang, X. S., & Deb, S. (2010). Engineering optimization by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, *1*(4), 330–343. doi:10.1504/IJMMNO.2010.035430

Yang, X. S., & Yeomans, J. S. (2014). Municipal waste management optimization using a firefly algorithm-driven simulation-optimization approach. *International Journal of Process Management and Benchmarking*, *4*(4), 363–375. doi:10.1504/IJPMB.2014.065518

Yeomans, J. S. (2017a). Simultaneous computing of sets of maximally different alternatives to optimal solutions. *International Journal of Engineering Research and Applications*, *7*(9), 21–28.

Yeomans, J. S. (2017b). An optimization algorithm that simultaneously calculates maximally different alternatives. *International Journal of Computational Engineering Research*, *7*(10), 45–50.

Yeomans, J. S. (2018a). A nature-inspired metaheuristic for generating alternatives. In M. Khosrow-Pour (Ed.), *Encyclopedia of information science and technology* (4th ed., pp. 2178–2187). IGI Global.

Yeomans, J. S. (2018b). An efficient computational procedure for simultaneously generating alternatives to an optimal solution using the firefly algorithm. In X. S. Yang (Ed.), *Nature-inspired algorithms and applied optimization* (pp. 261–273). Springer. doi:10.1007/978-3-319-67669-2_12

Yeomans, J. S. (2018c). A biologically-inspired metaheuristic approach for the simultaneous generation of alternatives. *International Journal of Computers in Clinical Practice*, *3*(2), 1–12. doi:10.4018/IJCCP.2018070101

Yeomans, J. S. (2018d). An algorithm for generating sets of maximally different alternatives using population-based metaheuristic procedures. *Transactions on Machine Learning and Artificial Intelligence*, *6*(5), 1–9. doi:10.14738/tmlai.65.5184

Yeomans, J. S. (2019a). A bicriterion approach for generating alternatives using population-based algorithms. *WSEAS Transactions on Systems*, *18*(4), 29–34.

Yeomans, J. S. (2019b). A simulation-optimization algorithm for generating sets of alternatives using population-based metaheuristic procedures. *Journal of Software Engineering and Simulation*, *5*(2), 1–6. doi:10.35629/9795-05020101

Yeomans, J. S. (2019c). Water resource management using stochastic multicriteria population-based algorithms for producing alternatives. *Journal of Earth and Environmental Sciences*, *3*(2), 1–10.

Yeomans, J. S. (2019d). A stochastic bicriteria procedure for creating system options. *Algorithms Research*, *5*(1), 11–18.

Yeomans, J. S. (2019e). A population-based multicriteria algorithm for alternative generation. *Transactions on Machine Learning and Artificial Intelligence*, *7*(4), 1–8.

Yeomans, J. S. (2019f). A stochastic, dual-criterion, simulation-optimization algorithm for generating alternatives. *Journal of Computing Science and Engineering : JCSE*, *5*(6), 1–10.

Yeomans, J. S. (2019g). Waste management using multicriteria population-based simulation-optimization algorithms. *Journal of Waste Management and Disposal*, *2*(1), 1–8.

Yeomans, J. S. (2020a). A stochastic, multicriteria, firefly algorithm-based approach for waste management decision-making. *International Journal of Mathematics, Game Theory, and Algebra*, *29*(4), 251–275.

Yeomans, J. S. (2020b). Alternative generation in complex decision modelling using a firefly algorithm metaheuristic approach. *International Journal of Hyperconnectivity and the Internet of Things*, *4*(2), 68–79. doi:10.4018/IJHIoT.2020070105

Yeomans, J. S. (2021). A multicriteria, bat algorithm approach for computing the range limited routing problem for electric trucks. *WSEAS Transactions on Circuits and Systems*, *20*(13), 96–106. doi:10.37394/23201.2021.20.13

Yeomans, J. S., & Gunalay, Y. (2011). Simulation-optimization techniques for modelling to generate alternatives in waste management planning. *Journal of Applied Operational Research*, *3*(1), 23–35.

Zechman, E. M., & Ranjithan, S. R. (2004). An evolutionary algorithm to generate alternatives (EAGA) for engineering optimization problems. *Engineering Optimization*, *36*(5), 539–553. doi:10.1080/03052150410001704863

Julian Scott Yeomans has been a Professor of Operations Management and Information Systems at the Schulich School of Business, York University since 1993. He is the Director for both the Master of Business Analytics (MBAN) program and the Master of Artificial Intelligence (MMAI) program at Schulich. He holds degrees in management science/information systems, environmental engineering, and statistics. He generally teaches courses on VBA programming and spreadsheet-based decision support systems. He has published 6 books and over 125 peer-reviewed, academic journal articles on a wide range of topics. His current research focuses upon simulation-decomposition, simulation-optimization, machine learning, visual analytics, population-based metaheuristics, and modelling-to-generate-alternatives. Recent application areas have included environmental informatics, solid/hazardous waste management, empirical finance, and the optimal osmotic dehydration of fruits, vegetables, and fungi.