A Stochastic, Dual-Criterion, Simulation-Optimization Algorithm for Generating Alternatives

Julian Scott Yeomans (*)

OMIS Area, Schulich School of Business
York University, 4700 Keele Street
Toronto, ON, M3J 1P3 Canada

Abstract. Complex stochastic engineering problems are frequently inundated with incompatible performance requirements and inconsistent performance specifications that can be difficult to identify when supporting decision models must be constructed. Consequently, it is often advantageous to create a set of dissimilar options that afford distinctive approaches to the problem. These alternatives should satisfy the required system performance criteria and yet be maximally different from each other in their decision spaces. The approach for creating such maximally different solution sets is referred to as modelling-to-generate-alternatives (MGA)**.** This paper describes a dual-criterion stochastic MGA procedure that can generate sets of maximally different alternatives for any simulation-optimization approach that employs a population-based search algorithm. This stochastic algorithmic approach is both computationally efficient and simultaneously produces the prescribed number of maximally different solution alternatives in a single computational run of the procedure.

Keywords. Dual-criterion Objectives, Population-based algorithms, Modelling-to-generate-alternatives, Simulation-Optimization, Metaheuristics

Acknowledgement
This research was supported in part by grant OGP0155871 from the Natural Sciences and Engineering Research Council.

1. Introduction

Complex stochastic engineering problems frequently include inconsistent and incompatible design specifications that can be difficult to formulate into mathematical decision-models [1], [2], [3], [4], [5]. Although "optimal" solutions can be determined for the mathematical models, whether these can truly be considered the best solutions to the "real" underlying problems can remain somewhat debatable [1], [2], [6]. Generally, it is better to construct a small number of distinct alternatives that provide dissimilar viewpoints for the particular problem [3], [7]. These dissimilar solutions should be close-to-optimal with respect to the specified objective(s), but be maximally different from each other within the decision domain. Numerous approaches collectively referred to as *modelling-to-generate-alternatives* (MGA) have been created to address this multi-solution requirement [6], [7], [8]. The prime directive behind MGA is the production of a set of alternatives that are "good" with respect to the specified objective(s), but are fundamentally dissimilar from each other in the decision space. Decision-makers then need to perform a subsequent evaluation of this set of alternatives to determine which specific alternative(s) most closely satisfy their specific goals. Consequently, MGA approaches are classified as decision support methods rather than as solution creation processes as assumed in

explicit optimization.

Early MGA methods employed direct, iterative approaches that produced alternatives by incrementally re-running their procedures whenever new solutions needed to be generated [6], [7], [8], [9], [10]. These iterative approaches imitated the seminal MGA method of Brill *et al*. [8] where, after the initial mathematical model had been optimized, all supplementary alternatives were produced one-at-a-time. Consequently, these incremental approaches all required n+1 iterations of their respective algorithms – initially to optimize the original problem, then to produce each of the subsequent n alternatives [7], [11], [12], [13], [14], [15], [16], [17], [18]).

In this paper, it is demonstrated how a set of maximally different solution alternatives can be generated by extending several earlier MGA techniques to stochastic optimization ([12], [13], [14], [15], [16], [17], [18]). The stochastic algorithm provides an MGA process that can be accomplished by *any* population-based mechanism. This algorithm advances earlier concurrent procedures ([13], [15], [16], [17], [18]) by permitting the simultaneous generation of n distinct alternatives in a single computational run. Specifically, to generate n maximally different alternatives, the algorithm runs exactly the same number of times that a function optimization procedure needs to run (i.e. once) irrespective of the value of n [19], [20], [21], [22], [23]. In [24] a new data structure was created that permits *simultaneous* alternatives to be constructed by population-based solution methods. In this paper, dual-criterion, max-sum, max-min objective is introduced that combines the novel data structure into the simultaneous solution approach to create an effective MGA approach. The max-sum portion of the objective endeavours to produce a maximum distance between solutions by ensuring that the total deviation between all of the variables in all of the alternatives is large. It does not preclude, however, the possibility of relatively small (or zero) deviations occurring between some of the individual variables in certain solutions. In contrast, the max-min objective seeks a maximum distance between every variable over all solutions. By considering both objectives simultaneously, the alternatives created will be as far apart as possible for all variables in general and the closest distance in the worst case between any solutions will never be less than the value obtained for the max-min objective. Furthermore, the dual-objective stochastic MGA algorithm employs a data structure that permits simultaneous alternatives to be constructed in a very computationally effective way. This data structure facilitates the above-mentioned solution generalization to all population-based methods.

2. Modelling to Generate Alternatives

Mathematical optimization has focused almost entirely on generating single optimal solutions to single-objective formulations or, equivalently, producing a set of noninferior solutions for multi-objective problems [2], [5], [8]. While such approaches may create solutions to the mathematical models, whether these outputs actually are the "best" solutions to the underlying real problems remains less certain [1], [2], [6], [8]. Within most "real world" decision-making environments, there are countless system specifications that can never be incorporated into the problem formulation [1], [5]. Furthermore, most subjective aspects remain unavoidably unmodelled and unquantified in the constructed decision models. Thus, most subjective features unavoidably remain unmodelled in the mathematical decision models. This frequently occurs when final decisions are made based not only upon modelled objectives, but also upon more subjective stakeholder preferences and socio-political-economic goals [7]. Several "real life" examples of such incongruent modelling characteristics are illustrated in [6], [8], [9], and [10].

When unmodelled objectives and unquantified issues exist, non-traditional methods are required for searching the decision region not only for noninferior sets of solutions, but also

for alternatives that are evidently *sub-optimal* to the modelled problem. Namely, any search for alternatives to problems known or suspected to contain unmodelled components must concentrate not only on a non-inferior set of solutions, but also necessarily on an explicit exploration of the problem's inferior solution space.

To demonstrate the implications of unmodelled objectives in a decision search, assume that an optimal solution for a maximization problem is $\boldsymbol{X}^*$ with objective value $Z1^*$ [24]. Suppose a second, unquantified, maximization objective $Z2$ exists that represents some "politically acceptable" factor. Assume that the solution, $\boldsymbol{X}^a$, belonging to the 2-objective noninferior set, exists that corresponds to a best compromise solution if both objectives could have been simultaneously considered. Although $\boldsymbol{X}^a$ would be considered as the best solution to the real problem, in the actual mathematical model it would appear inferior to solution $\boldsymbol{X}^*$, since $Z1^a \leq Z1^*$. Therefore, when unquantified components are included in the decision-making process, inferior decisions to the mathematically modelled problem could be optimal to the underlying "real" problem. Thus, when unquantified issues and unmodelled objectives could exist, alternative solution procedures are required to not only explore the decision domain for noninferior solutions to the modelled problem, but also to concurrently search the decision domain for inferior solutions. Population-based algorithms prove to be proficient solution methods for concurrent searches throughout a decision space.

The objective of MGA is to construct a workable set of alternatives that are quantifiably good with respect to the modelled objectives, yet are as different as possible from each other within the solution space. By accomplishing this requirement, the resulting set of alternatives is able to provide truly different perspectives that perform similarly with respect to the known modelled objective(s) yet very differently with respect to various potentially unmodelled aspects. By creating these good-but-different solutions, the decision-makers are able to explore potentially desirable qualities within the alternatives that might be able to satisfy the unmodelled objectives to varying degrees of stakeholder acceptability.

To motivate the MGA process, it is necessary to more formally characterize the mathematical definition of its goals [6], [7]. Assume that the optimal solution to an original mathematical model is $\boldsymbol{X}^*$ with corresponding objective value $\boldsymbol{Z}^* = F(\boldsymbol{X}^*)$. The resultant difference model can then be solved to produce an alternative solution, $\boldsymbol{X}$, that is maximally different from $\boldsymbol{X}^*$:

$$\text{Maximize} \qquad \Delta(\boldsymbol{X}, \boldsymbol{X}^*) = \min_i |X_i - X_i^*| \tag{1}$$

$$\text{Subject to:} \qquad \boldsymbol{X} \in D \tag{2}$$

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

where Δ represents an appropriate difference function (shown in (1) as an absolute difference) and T is a tolerance target relative to the original optimal objective value $\boldsymbol{Z}^*$. T is a user-specified limit that determines what proportion of the inferior region needs to be explored for acceptable alternatives. This difference function concept can be extended into a difference measure between any *set of alternatives* by replacing $\boldsymbol{X}^*$ in the objective of the maximal difference model and calculating the overall minimum absolute difference (or some other function) of the pairwise comparisons between corresponding variables in each pair of alternatives – subject to the condition that each alternative is feasible and falls within the specified tolerance constraint.

The population-based MGA procedure to be introduced is designed to generate a pre-determined small number of close-to-optimal, but maximally different alternatives, by adjusting the value of T and solving the corresponding maximal difference problem instance by exploiting the population structure of the algorithm. The survival of solutions depends upon

how well the solutions perform with respect to the problem's originally modelled objective(s) and simultaneously by how far away they are from all of the other alternatives generated in the decision space.

3. Simulation-Optimization for Stochastic Optimization

Finding optimal solutions to large stochastic problems proves complicated when numerous system uncertainties must be directly incorporated into the solution procedures ([25], [26], [27], [28]). *Simulation-Optimization* (SO) is a broadly defined family of stochastic solution approaches that combines simulation with an underlying optimization component for optimization [25]. In SO, all unknown objective functions, constraints, and parameters are replaced by simulation models in which the decision variables provide the settings under which simulation is performed.

The general steps of SO can be summarized in the following fashion ([26], [29]). Suppose the mathematical model of the optimization problem contains n decision variables, X_i, represented in the vector $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]$. If the objective function is expressed by F and the feasible region is designated by D, then the mathematical programming problem is to optimize $F(\boldsymbol{X})$ subject to $\boldsymbol{X} \in D$. When stochastic conditions exist, values for the objective and constraints can be determined by simulation. Any solution comparison between two different solutions ***X1*** and ***X2*** requires the evaluation of some statistic of F modelled with ***X1*** compared to the same statistic modelled with ***X2*** ([25], [30]). These statistics are calculated by simulation, in which each $\boldsymbol{X}$ provides the decision variable settings employed in the simulation. While simulation provides a means for comparing results, it does not provide the mechanism for determining optimal solutions to problems. Hence, simulation cannot be used independently for stochastic optimization.

Since all measures of system performance in SO are stochastic, every potential solution, $\boldsymbol{X}$, must be calculated through simulation. Because simulation is computationally intensive, an optimization algorithm is employed to guide the search for solutions through the problem's feasible domain in as few simulation runs as possible ([27], [30]). As stochastic system problems frequently contain numerous potential solutions, the quality of the final solution could be highly variable unless an extensive search has been performed throughout the entire feasible region. A stochastic SO approach contains two alternating computational phases; (i) an "evolutionary" module directed by some optimization (frequently a metaheuristic) method and (ii) a simulation module ([31]). Because of the stochastic components, all performance measures are necessarily statistics calculated from the responses generated in the simulation module. The quality of each solution is found by having its performance criterion, F, evaluated in the simulation module. After simulating each candidate solution, their respective objective values are returned to the evolutionary module to be utilized in the creation of ensuing candidate solutions. Thus, the evolutionary module aims to advance the system toward improved solutions in subsequent generations and ensures that the solution search does not become trapped in some local optima. After generating new candidate solutions in the evolutionary module, the new solution set is returned to the simulation module for comparative evaluation. This alternating, two-phase search process terminates when an appropriately stable system state (i.e. an optimal solution) has been attained. The optimal solution produced by the procedure is the single best solution found throughout the course of the entire search process ([31]).

Population-based algorithms are conducive to SO searches because the complete set of candidate solutions maintained in their populations permit searches to be undertaken throughout multiple sections of the feasible region, concurrently. For population-based

optimization methods, the evolutionary phase evaluates the entire current population of solutions during each generation of the search and evolves from a current population to a subsequent one. A primary characteristic of population-based procedures is that better solutions in a current population possess a greater likelihood for survival and progression into the subsequent population.

It has been shown that SO can be used as a very computationally intensive, stochastic MGA technique ([30], [32]). However, because of the very long computational runs, several approaches to accelerate the search times and solution quality of SO have been examined subsequently [29]. The next section provides an MGA algorithm that incorporates stochastic uncertainty using SO to much more efficiently generate sets of maximally different solution alternatives.

4. Population-based, Dual-Criterion Simulation-Optimization MGA Algorithm

In this section, a data structure is employed that enables a dual-criterion MGA solution approach via any population-based algorithm [24], [33], [34]. Suppose that it is desired to produce P alternatives that each possess n decision variables and that the population algorithm is to possess K solutions in total. That is, each solution contains one possible set of P maximally different alternatives. Let $\boldsymbol{Y}_k$, $k = 1,\ldots, K$, represent the k^{th} solution which consists of one complete set of P different alternatives. Specifically, if $\boldsymbol{X}_{kp}$ corresponds to the p^{th} alternative, $p = 1,\ldots, P$, of solution k, $k = 1,\ldots, K$, then $\boldsymbol{Y}_k$ can be represented 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

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

Consequently, the entire population, $\boldsymbol{Y}$, consisting of K different sets of P alternatives can be expressed in vectorized form as,

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

The following population-based MGA method produces a pre-determined number of close-to-optimal, but maximally different alternatives, by modifying the value of the bound T in the maximal difference model and using any population-based method to solve the corresponding, maximal difference problem. The dual-criterion MGA algorithm that follows constructs a pre-determined number of maximally different, near-optimal alternatives, by modifying the bound value T in the maximal difference model and using any population-based technique to solve the corresponding maximal difference problem. Each solution in the population comprises one set of p different alternatives. By exploiting the co-evolutionary aspects of the algorithm, the algorithm evolves each solution toward sets of dissimilar local optima within the solution domain. In this processing, each solution alternative mutually experiences the search steps of the algorithm. Solution survival depends upon both how well the solutions perform with respect to the modelled objective(s) and by how far apart they are from every other alternative in the decision space.

A straightforward process for generating alternatives solves the maximum difference model iteratively by incrementally updating the target T whenever a new alternative needs to be produced and then re-solving the resulting model [24]. This iterative approach parallels the original Hop, Skip, and Jump (HSJ) MGA algorithm [8] in which the alternatives are created one-by-one through an incremental adjustment of the target constraint. While this approach is straightforward, it entails a repetitive execution of the optimization algorithm [7], [12], [13]. To improve upon the stepwise HSJ approach, a concurrent MGA technique was subsequently

designed based upon co-evolution ([13], [15], [17]). In a co-evolutionary approach, pre-specified stratified subpopulation ranges within an algorithm's overall population are established that collectively evolve the search toward the specified number of maximally different alternatives. Each desired solution alternative is represented by each respective subpopulation and each subpopulation undergoes the common processing operations of the procedure. The survival of solutions in each subpopulation depends simultaneously upon how well the solutions perform with respect to the modelled objective(s) and by how far away they are from all of the other alternatives. Consequently, the evolution of solutions in each subpopulation toward local optima is directly influenced by those solutions contained in all of the other subpopulations, which forces the concurrent co-evolution of each subpopulation towards good but maximally distant regions within the decision space according to the maximal difference model [7]. Co-evolution is also much more efficient than a sequential HSJ-style approach in that it exploits the inherent population-based searches to concurrently generate the entire set of maximally different solutions using only a single population [12], [17].

While concurrent approaches can exploit population-based algorithms, co-evolution can only occur within each of the stratified subpopulations. Consequently, the maximal differences between solutions in different subpopulations can only be based upon aggregate subpopulation measures. Conversely, in the following simultaneous MGA algorithm, each solution in the population contains exactly one entire set of alternatives and the maximal difference is calculated only for that particular solution (i.e. the specific alternative set contained within that solution in the population). Hence, by the evolutionary nature of the population-based search procedure, in the subsequent approach, the maximal difference is simultaneously calculated for the specific set of alternatives considered within each specific solution – and the need for concurrent subpopulation aggregation measures is avoided.

Using the data structure terminology, the steps for the dual-criterion MGA algorithm are as follows ([14], [19], [20], [21], [22], [23], [24], [33], [34]). It should be readily apparent that the stratification approach employed by this method can be easily modified for any population-based algorithm.

Initialization Step. Solve the original optimization problem to find its optimal solution, $\boldsymbol{X}^*$. Based upon the objective value $F(\boldsymbol{X}^*)$, establish P target values. P represents the desired number of maximally different alternatives to be generated within prescribed target deviations from the $\boldsymbol{X}^*$. Note: The value for P has to have been set *a priori* by the decision-maker.

Without loss of generality, it is possible to forego this step and to use the algorithm to find $\boldsymbol{X}^*$ as part of its solution processing in the subsequent steps. However, this significantly increases the number of iterations of the computational procedure and the initial stages of the processing become devoted to finding $\boldsymbol{X}^*$ while the other elements of each population solution are retained as essentially "computational overhead".

Step 1. Create an initial population of size K where each solution contains P equally-sized partitions. The partition size corresponds to the number of decision variables in the original optimization problem. $\boldsymbol{X}_{kp}$ represents 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 simulation module with respect to the modelled objective. Alternatives meeting their target constraint and all other problem constraints are designated as *feasible*, while all other alternatives are designated as *infeasible*.

Note: A solution can be designated as feasible only if all of the alternatives contained within it are feasible.

Step 3. Apply an appropriate elitism operator to each solution to rank order the best individuals in the population. The best solution is the feasible solution containing the most

distant set of alternatives in the decision space (the distance measures are defined in Step 5).

Note: Because the best solution to date is always retained in the population throughout each iteration, at least one solution will always be feasible. A feasible solution for the first step can always consists of P repetitions of $\boldsymbol{X}^*$.

Step 4. Stop the algorithm if the termination criteria (such as maximum number of iterations or some measure of solution convergence) are met. Otherwise, proceed to Step 5.

Step 5. For each solution $\boldsymbol{Y}_k$, $k = 1,\ldots, K$, calculate D^1_k and D^2_k, which are the dual-criterion Max-Min and Max-Sum distance measures determined, respectively, between all of the alternatives contained within the solution.

As an illustrative example for calculating the dual-criterion distance measures, compute

$$D^1_k = \Delta^1(\boldsymbol{X}_{ka}, \boldsymbol{X}_{kb}) = \underset{a,b,q}{\text{Min}} \, | X_{kaq} - X_{kbq} | , \qquad a = 1,\ldots,P,\ b = 1,\ldots,P,\ q = 1,\ldots,n, \quad (7)$$

and

$$D^2_k = \Delta^2(\boldsymbol{X}_{ka}, \boldsymbol{X}_{kb}) = \sum_{a=1 \text{to} P} \sum_{b=1 \text{to} P} \sum_{q=1\ldots n} (X_{kaq} - X_{kbq})^2. \quad (8)$$

D^1_k denotes the minimum absolute distance and D^2_k represents the overall quadratic deviation between all of the alternatives contained within solution k. Alternatively, the distance functions could be calculated using some other appropriately defined measures.

Step 6. This step orders the specific solutions by those solutions which contain the set of alternatives which are most distant from each other. The goal of maximal difference is to force alternatives to be as far apart as possible in the decision space from the alternatives of each of the partitions within each solution.

Let $D_k = G(D^1_k, D^2_k)$ represent the dual-criterion objective for solution k. Rank the solutions according to the distance measure D_k objective – appropriately adjusted to incorporate any constraint violation penalties for infeasible solutions.

Step 7. Apply applicable algorithmic "change operations" to each solution within the population and return to Step 2.

5. Conclusions

Complex problem solving inherently involves unquantifiable aspects and incongruent performance specifications. These decision settings frequently possess inconsistent structural elements that are difficult – if not impossible – to integrate into ancillary decision support models. There are always unmodelled features, not apparent during model formulation, that can significantly impact the adequacy of its solutions. These confounding components require decision-makers to incorporate numerous uncertainties into their solution process prior to the eventual problem resolution. When challenged by these incongruencies, it is unlikely that any single solution can concurrently satisfy all of the ambiguous system requirements. Consequently, MGA decision support approaches have been created to address these confounding features in some way, while simultaneously retaining enough flexibility to incorporate the inherent planning incongruities.

This paper has applied a population-based, dual-criterion MGA procedure to WRM. This computationally efficient MGA approach establishes how population-based algorithms can simultaneously construct entire sets of close-to-optimal, maximally different alternatives by exploiting the evolutionary characteristics of population-based solution algorithms. In this MGA role, the dual-criterion objective can efficiently generate the requisite set of dissimilar alternatives, with each generated solution providing an entirely different perspective to the

problem. The max-sum objective criteria ensure that the distances between the alternatives created by this approach are good in general, while the max-min criteria ensure that the distances between the alternatives are good in the worst case. The absolute function has been considered, since its value provides a meaningful, physical interpretation to its measure of distance. Since population-based procedures can be applied to a wide range of problem types, the practicality of this dual-criterion MGA approach can be extended to wide array of "real world" environmental applications. Such extensions will be explored in future computational studies.

References

[1] M. Brugnach, A. Tagg, F. Keil, and W.J. De Lange, Uncertainty matters: computer models at the science-policy interface, *Water Resources Management*, *21*, 2007, 1075-1090.

[2] J.A.E.B. Janssen, M.S. Krol, R.M.J. Schielen, and A.Y. Hoekstra, The effect of modelling quantified expert knowledge and uncertainty information on model-based decision making, *Environmental Science and Policy*, *13(3)*, 2010, 229-238.

[3] M. Matthies, C. Giupponi, and B. Ostendorf, Environmental decision support systems: Current issues, methods and tools, *Environmental Modelling and Software*, *22(2)*, 2007, 123-127.

[4] H.T. Mowrer, Uncertainty in natural resource decision support systems: Sources, interpretation, and importance, *Computers and Electronics in Agriculture*, *27(1-3)*, 2000, 139-154.

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

[6] D.H. Loughlin, S.R. Ranjithan, E.D. Brill, and J.W. Baugh, Genetic algorithm approaches for addressing unmodelled objectives in optimization problems, *Engineering Optimization*, *33(5)*, 2001, 549-569.

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

[8] E.D. Brill, S.Y. Chang, and L.D. Hopkins, Modelling to generate alternatives: the HSJ approach and an illustration using a problem in land use planning, *Management Science*, *28(3)*, 1982, 221-235.

[9] J.W. Baugh, S.C. Caldwell, and E.D. Brill, A mathematical programming approach for generating alternatives in discrete structural optimization, *Engineering Optimization*, *28(1)*, 1997, 1-31.

[10] E.M. Zechman, and S.R. Ranjithan, Generating alternatives using evolutionary algorithms for water resources and environmental management problems, *Journal of Water Resources Planning and Management*, *133(2)*, 2007, 156-165.

[11] Y. Gunalay, J.S. Yeomans, and G.H. Huang, Modelling to generate alternative policies in highly uncertain environments: An application to municipal solid waste management planning, *Journal of Environmental Informatics*, *19(2)*, 2012, 58-69.

[12] R. Imanirad, and J.S. Yeomans, Modelling to Generate Alternatives Using Biologically Inspired Algorithms, in X.S. Yang (Ed.), *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, (Amsterdam: Elsevier, 2013), 313-333.

[13] R. Imanirad, X.S. Yang, and J.S. Yeomans, A computationally efficient, biologically-inspired modelling-to-generate-alternatives method, *Journal on Computing*, *2(2)*, 2012, 43-47.

[14] J.S. Yeomans, 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*. (New York: Springer, 2018) 261-273.

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

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

[17] R. Imanirad, X.S. Yang, and J.S. Yeomans, A Concurrent Modelling to Generate Alternatives Approach Using the Firefly Algorithm, *International Journal of Decision Support System Technology*, *5(2)*, 2013, 33-45.

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

[19] J.S. Yeomans, Simultaneous Computing of Sets of Maximally Different Alternatives to Optimal Solutions, *International Journal of Engineering Research and Applications*, *7(9)*, 2017, 21-28.

[20] J.S. Yeomans, An Optimization Algorithm that Simultaneously Calculates Maximally Different Alternatives, *International Journal of Computational Engineering Research*, *7(10)*, 2017, 45-50.

[21] J.S. Yeomans, Computationally Testing the Efficacy of a Modelling-to-Generate-Alternatives Procedure for Simultaneously Creating Solutions, *Journal of Computer Engineering*, *20(1)*, 2018, 38-45.

[22] J.S. Yeomans, A Computational Algorithm for Creating Alternatives to Optimal Solutions, *Transactions on Machine Learning and Artificial Intelligence*, 5(5), 2017, 58-68.

[23] J.S. Yeomans, A Simultaneous Modelling-to-Generate-Alternatives Procedure Employing the Firefly Algorithm, in N. Dey (Ed.), *Technological Innovations in Knowledge Management and Decision Support*. (Hershey, Pennsylvania: IGI Global, 2019) 19-33.

[24] J.S. Yeomans, An Algorithm for Generating Sets of Maximally Different Alternatives Using Population-Based Metaheuristic Procedures, *Transactions on Machine Learning and Artificial Intelligence*, 6(5), 2018, 1-9.

[25] M.C. Fu, Optimization for Simulation: Theory vs. Practice, *INFORMS Journal on Computing*, 14(3), 2002, 192-215.

[26] P. Kelly, Simulation Optimization is Evolving, *INFORMS Journal on Computing*, 14(3),

2002, 223-225.

[27] R. Zou, Y. Liu, J. Riverson, A. Parker and S. Carter, A Nonlinearity Interval Mapping Scheme for Efficient Waste Allocation Simulation-Optimization Analysis, *Water Resources Research*, 46(8), 2010, 1-14.

[28] R. Imanirad, X.S. Yang and J.S. Yeomans, Stochastic Decision-Making in Waste Management Using a Firefly Algorithm-Driven Simulation-Optimization Approach for Generating Alternatives, in S. Koziel, L. Leifsson, and X.S. Yang (Eds.), Recent *Advances in Simulation-Driven Modeling and Optimization*, (Heidelberg, Germany: Springer, 2016), 299-323.

[29] J.S. Yeomans, Waste Management Facility Expansion Planning Using Simulation-Optimization with Grey Programming and Penalty Functions, *International Journal of Environmental and Waste Management*, 10(2/3), 2012, 269-283.

[30] J.S. Yeomans, Applications of Simulation-Optimization Methods in Environmental Policy Planning Under Uncertainty, *Journal of Environmental Informatics*, 12(2), 2008, 174-186.

[31] J.S. Yeomans, and X.S. Yang, Municipal Waste Management Optimization Using a Firefly Algorithm-Driven Simulation-Optimization Approach, *International Journal of Process Management and Benchmarking*, 4(4), 2014, 363-375.

[32] J.D. Linton, J.S. Yeomans and R. Yoogalingam, Policy Planning Using Genetic Algorithms Combined with Simulation: The Case of Municipal Solid Waste, *Environment and Planning B: Planning and Design*, 29(5), 2002, 757-778.

[33] J.S. Yeomans, A Bicriterion Approach for Generating Alternatives Using Population-Based Algorithms, *WSEAS Transactions on Systems*, 18(4), 2019, 29-34.

[34] J.S. Yeomans, A Simulation-Optimization Algorithm for Generating Sets of Alternatives Using Population-Based Metaheuristic Procedures, *Journal of Software Engineering and Simulation*, Forthcoming.