Society for Science and Education
United Kingdom
TMLAI Transactions on Machine Learning and Artificial Intelligence
Volume 6 No. 5
ISSN 2054-7390

An Algorithm for Generating Sets of Maximally Different Alternatives Using Population-Based Metaheuristic Procedures

[1] **Julian Scott Yeomans**
[1] *OMIS Area, Schulich School of Business, York University, Toronto, ON, M3J 1P3 Canada;*
syeomans@schulich.yorku.ca

ABSTRACT

"Real world" problems typically possess complex performance conditions peppered with inconsistent performance requirements. This situation occurs because multifaceted problems are often riddled with incompatible performance objectives and contradictory design requirements which can be difficult – if not impossible – to specify when the requisite decision models are formulated. Thus, it is often desirable to generate a set of disparate alternatives that provide diverse approaches to the problem. These dissimilar options should be close-to-optimal with respect to any specified objective(s), but remain maximally different from all other solutions in the decision space. The approach for creating maximally different sets of solutions is referred to as modelling-to-generate-alternatives (MGA). This paper outlines an MGA algorithmic approach that can simultaneously generate a set of maximally different alternatives using any population-based metaheuristic.

Keywords: Modelling-to-generate-alternatives, Metaheuristics, Population-based algorithms.

1 Introduction

"Real world" decision-making environments involve complex problems containing design specifications that are frequently difficult to incorporate into underlying mathematical programming formulations and are often overwhelmed with numerous unquantifiable components [1]-[5]. Whereas "optimal" solutions can be calculated for the modelled representations, whether these are truly the best solutions to the real problems can be questionable, as there are always unmodeled components when mathematical models are constructed [1][2][6]. Generally, it is more desirable to create a discrete number of dissimilar alternatives that afford contrasting perspectives to the particular problem [3][7]. All of these options should be close-to-optimal with respect to any specified objective(s), but should be maximally different from each other within the decision space. Numerous procedures referred to as *modelling-to-generate-alternatives* (MGA) have been created to address this multi-solution approach [6]-[8].

The primary impetus behind MGA methods is to produce a set of alternatives that can be considered good when measured by the specified objective(s), but which are inherently distinct from one another within the decision domain. The resulting solution set should deliver alternative perspectives that perform similarly with respect to all modelled objectives, yet very differently with respect to any unmodelled aspects [5]. Decision-makers must conduct a subsequent assessment of the set of alternatives to determine which alternative(s) would most nearly achieve their specific requirements. Consequently,

DOI: 10.14738/tmlai.65.5184
Publication Date: 23rd September, 2018
URL: http://dx.doi.org/10.14738/tmlai.65.5184

MGA methods are classified as decision support procedures rather than as solution determination processes as assumed for explicit optimization.

Earlier MGA methods have employed direct, iterative approaches for alternative generation by incrementally re-running their algorithms whenever new solutions need be constructed [6]-[10]. These iterative methods imitate the seminal MGA approach of Brill *et al*. [8] where, once the initial mathematical formulation has been optimized, all supplementary alternatives are produced one-at-a-time. Consequently, these incremental approaches all require *n*+1 iterations of their respective algorithms – initially to optimize the original problem, then to produce each of the subsequent *n* alternatives [7][11]-[13].

In this study, it is demonstrated how a set of maximally different solution alternatives can be *simultaneously* generated using any population-based metaheuristic algorithm by extending several earlier MGA approaches [12]-[18]. All of the earlier MGA procedures employed the Firefly Algorithm (FA) for their solution procedure. The FA is a very specific instance of one population-based metaheuristic procedure that can be used for solving optimization problems. In this paper, a new algorithm is provided that has been updated and generalized so that now a simultaneous MGA solution process can be achieved using *any* population-based mechanism. This new MGA algorithmic approach advances the earlier concurrent procedures of Imanirad *et al*. [13][15]-[18] by permitting the simultaneous generation of the overall best solution together with *n* distinct alternatives in a single computational run. Stated explicitly, to generate the *n* maximally different solution alternatives, the new MGA algorithm would run exactly the same number of times that a procedure would need to be run for function optimization alone (i.e. once) irrespective of the value of *n* [19]-[23]. Furthermore, a new data structure is created that permits simultaneous alternatives to be constructed in a very novel, highly computationally efficient way. It is the implementation of this data structure which facilitates the above-mentioned generalization to solution by all population-based methods. Consequently, this simultaneous MGA algorithmic approach proves to be extremely computationally efficient.

2 Modelling to Generate Alternatives

Mathematical programming methods appearing in the optimization literature have focused almost exclusively upon 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 methods may establish solutions to the derived complex mathematical models, whether their outputs actually generate “best” solutions to the real, underlying problems is somewhat less certain [1][2][6][8]. Within most “real world” decision circumstances, there are countless system requirements and objectives that will never be explicitly apparent or included in the problem formulation stage [1][5]. Furthermore, it may not be possible to explicitly convey all of the subjective requirements as there are frequently numerous incompatible, design components and adversarial stakeholders involved. Therefore, most subjective aspects remain unavoidably unmodelled and unquantified in the constructed decision models. This commonly occurs where final decisions are constructed based not only upon modelled objectives, but also upon more subjective stakeholder preferences and socio-political-economic goals [7]. Numerous “real world” illustrations of such incongruent modelling dualities are discussed in [6][8]-[10].

When unmodelled issues and unquantified objectives exist, non-conventional methods are needed to not only search the decision region for noninferior sets of solutions, but to also explore the decision region for alternatives that are obviously *sub-optimal* for the problem modelled. Specifically, any search for alternatives to problems suspected or known to contain unmodelled components needs to focus not only on a non-inferior set of solutions, but necessarily also on an unambiguous exploration of the problem's inferior solution space.

To demonstrate the consequences of an unmodelled objective in a decision search, assume that the quantifiably optimal solution for a single-objective, maximization problem is $\boldsymbol{X}^*$ with a corresponding objective value $Z1^*$. Now suppose that a second, unmodelled, maximization objective $Z2$ exists that subjectively incorporates some unquantifiable "politically acceptable" component. Now assume that some solution, $\boldsymbol{X}^a$, belonging to the 2-objective noninferior set, exists that represents a potentially best compromise solution for the decision-maker if both objectives had somehow been simultaneously evaluated. While $\boldsymbol{X}^a$ could reasonably be considered as the best compromise solution for the real problem, in the quantified mathematical model it would appear inferior to solution $\boldsymbol{X}^*$, since it must be the case that $Z1^a \leq Z1^*$. Therefore, when unmodelled components are incorporated into a decision-making process, mathematically inferior options to the modelled problem could actually be optimal for the real underlying problem. Consequently, when unmodelled issues and unquantified objectives potentially exist, alternative solution procedures are needed to not only explore the decision region for noninferior sets of solutions, but also to concurrently search the decision region for inferior solutions to the problem modelled. Population-based algorithms permit concurrent searches throughout a decision space and prove to be particularly proficient solution methods.

The principal drive for MGA is to create a manageably small set of alternatives that are as different from each other as possible within the solution space, yet are quantifiably good with respect to all modelled objectives. By achieving this, the resultant set of solution alternatives is able to supply truly different choices that perform similarly with respect to the known modelled objective(s) yet very differently with respect to any unmodelled issues. By generating such good-but-different solutions, the decision-makers are able to examine potentially desirable qualities within the alternatives that might satisfactorily be able to address the unmodelled objectives to varying degrees of stakeholder acceptability.

In order to motivate the MGA search process, it becomes necessary to apply a more formal mathematical definition to the goals of MGA [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 ensuing difference model can then be solved to produce an alternative solution, $\boldsymbol{X}$, that is maximally different from $\boldsymbol{X}^*$:

$$\text{Maximize} \qquad \Delta(\boldsymbol{X}, \boldsymbol{X^*}) = \sum_i \left(X_i - X_i^* \right)^2 \tag{1}$$

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

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

where Δ represents an appropriate difference function (for clarity, shown as a quadratic difference in this instance) and T is a specified targeted tolerance limit relative to the original optimal objective value $\boldsymbol{Z^*}$. T is user-supplied and quantifies what proportion of the inferior region must be explored in the solution search 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 sum (or some other function) of the differences of the pairwise comparisons between 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 metaheuristic. 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 Population-based Simultaneous MGA Computational Algorithm

In this section, a novel data structure is introduced that permits alternatives to be simultaneously constructed in a computationally efficient way that also enables an algorithmic generalization to solution by any population-based procedure. Suppose that it is desired to be able 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 is to contain one possible set of *P* maximally different alternatives. In this representation, let $\boldsymbol{Y}_k$, $k = 1,\dots, K$, represent the kth solution which is made up of one complete set of *P* different alternatives. Namely, if $\boldsymbol{X}_{kp}$ is the pth alternative, $p = 1,\dots, P$, of solution *k*, $k = 1,\dots, K$, then $\boldsymbol{Y}_k$ can be represented as

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

If X_{kjq}, $q = 1,\dots, n$, is the qth variable in the jth alternative of solution k, then

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

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

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

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 metaheuristic to solve the corresponding, maximal difference problem. Each solution within the population contains one potential set of *p* different alternatives. By exploiting the co-evolutionary solution structure within the metaheuristic, the algorithm collectively evolves each solution toward sets of different local optima within the solution space. In this process, each desired solution alternative undergoes the common search procedure of the metaheuristic. However, the survival of solutions depends both 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 generated in the decision space.

A straightforward process for generating alternatives would be to iteratively solve the maximum difference model by incrementally updating the target *T* whenever a new alternative needs to be produced and then re-running the algorithm. This iterative approach would parallel the original Hop, Skip, and Jump (HSJ) MGA algorithm of Brill *et al*. [8] in which, once an initial problem formulation has been optimized, supplementary alternatives are systematically created one-by-one through an incremental adjustment of the target constraint to force the sequential generation of the suboptimal solutions. While this approach is straightforward, it requires a repeated execution of the optimization algorithm [7][12][13].

To improve upon the stepwise alternative approach of the HSJ algorithm, a concurrent MGA technique was subsequently designed based upon the concept of co-evolution Imanirad *et al*. [13][15][17]. In a co-evolutionary approach, pre-specified stratified subpopulation ranges within an algorithm's overall population were established that collectively evolved the search toward the creation of 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].

By employing this co-evolutionary concept, it becomes possible to implement an MGA procedure that concurrently produces alternatives which possess objective function bounds that are somewhat analogous to those created by the sequential, iterative HSJ-styled solution generation approach. While each alternative produced by an HSJ procedure is maximally different only from the overall optimal solution (together with its bound on the objective value which is at least x% different from the best objective (i.e. x = 1%, 2%, etc.)), a concurrent procedure is able to generate alternatives that are no more than x% different from the overall optimal solution but with each one of these solutions being as maximally different as possible from every other generated alternative that was produced. 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 a concurrent approach can exploit population-based solution approaches, the co-evolution process 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 circumvented.

Using the terminology introduced above, the steps in the co-evolutionary population-based MGA procedure are as follows ([14][19]-[23]:

Preliminary Step. In this initialization step, solve the original optimization problem to determine the optimal solution, ***X****. As with prior solution approaches Imanirad *et al*. [13][15]-[18]) and without loss of generality, it is entirely possible to forego this step and construct the algorithm to find ***X**** as part of its solution processing. However, such a requirement increases the number of computational iterations of the overall procedure and the initial stages of the processing focus upon finding $\boldsymbol{X^*}$ while the other elements of each population solution remain essentially "computational overhead". 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.

Step 1. Create the initial population of size *K* in which each solution is divided into *P* equally-sized partitions. The size of each partition corresponds to the number of variables for the original optimization problem. $\boldsymbol{X}_{kp}$ represents the p^{th} alternative, $p = 1,\ldots,P$, in solution $\boldsymbol{Y}_{\boldsymbol{k}}$, $k = 1,\ldots,K$.

Step 2. In each of the *K* solutions, evaluate each $\boldsymbol{X}_{\boldsymbol{kp}}$, $p = 1,\ldots,P$, 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*. A solution can only be designated as feasible 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 measure is 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_k, a distance measure between all of the alternatives contained within the solution.

As an illustrative example for determining a distance measure, calculate

$$D_k = \Delta (\boldsymbol{X}_{ka}, \boldsymbol{X}_{kb}) = \sum_{a=1 to P} \sum_{b=1 to P} \sum_{q=1\ldots n} (X_{kaq} - X_{kbq})^2. \qquad (7)$$

This represents the total quadratic distance between all of the alternatives contained within solution *k*. Alternatively, the distance measure could be calculated by some other appropriately defined function

Step 6. Rank the solutions according to the distance measure D_k objective – appropriately adjusted to incorporate any constraint violation penalties for infeasible solutions. 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. This step orders the specific solutions by those solutions which contain the set of alternatives which are most distant from each other.

Step 7. Apply appropriate metaheuristic "change operations" to the each of the solutions within the population and return to Step 2.

It should be apparent that the stratification approach outlined in this algorithm can be easily modified to accommodate any population-based solution procedure.

4 Conclusion

“Real world” decision-making situations inherently involve complicated performance components that are further confounded by incongruent requirements and unquantifiable performance objectives. 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 the adequacy of its solutions. These confounding features require the decision-makers to integrate numerous uncertainties into their solution process before an ultimate solution can be determined. Faced with such incongruencies, it is unlikely that any single solution can simultaneously satisfy all ambiguous system requirements without significant compromises. Therefore, any decision support approach must somehow address these complicating facets in some way, while simultaneously being flexible enough to condense the potential effects within the intrinsic planning incongruities.

This study has provided an updated computational procedure, a new data structure, and a significant solution-approach generalization to what has appeared previously in the literature. This new computationally efficient MGA procedure demonstrates how population-based metaheuristics can simultaneously construct entire sets of close-to-optimal, maximally different alternatives by exploiting the evolutionary characteristics of any population-based solution approach. In this MGA role, the simultaneous algorithm efficiently generates the requisite set of disparate alternatives, with each solution generated outlining a completely different perspective to the problem. Since population-based methods can be applied to a diverse spectrum of problem types, the efficacy of this new simultaneous MGA algorithm can be extended to wide range of “real world” applications. These extensions will become the topic of future studies.

[1]. **REFERENCES**

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

[2] Janssen, J.A.E.B., 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, 2010. 13(3): p. 229-238.

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

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

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

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

[7] Yeomans, J.S., and Y Gunalay, *Simulation-Optimization Techniques for Modelling to Generate Alternatives in Waste Management Planning*. Journal of Applied Operational Research, 2011. 3(1): p. 23-35.

[8] Brill, E.D., 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. 1982. 28(3): p. 221-235.

[9] Baugh, J.W., S.C. Caldwell, and E.D Brill, *A Mathematical Programming Approach for Generating Alternatives in Discrete Structural Optimization*. Engineering Optimization. 1997, 28(1): p. 1-31.

[10] Zechman, E.M., and S.R. Ranjithan, *An Evolutionary Algorithm to Generate Alternatives (EAGA) for Engineering Optimization Problems*. Engineering Optimization. 2004, 36(5): p. 539-553.

[11] Gunalay, Y., 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, 2012. *19(2)*: p. 58-69.

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

[13] Imanirad, R., X.S. Yang, and J.S. Yeomans, *A Computationally Efficient, Biologically-Inspired Modelling-to-Generate-Alternatives Method*. Journal on Computing. 2012, 2(2): p. 43-47.

[14] Yeomans, J.S., *An Efficient Computational Procedure for Simultaneously Generating Alternatives to an Optimal Solution Using the Firefly Algorithm*, in *Nature-Inspired Algorithms and Applied Optimization*, Yang, X.S. Editor 2018. Heidelberg (Springer), Germany. p. 261-273.

[15] Imanirad, R., X.S. Yang, and J.S. Yeomans, *A Co-evolutionary, Nature-Inspired Algorithm for the Concurrent Generation of Alternatives*. Journal on Computing. 2012, 2(3): p. 101-106.

[16] Imanirad, R., X.S. Yang, and J.S. Yeomans, *Modelling-to-Generate-Alternatives Via the Firefly Algorithm*. Journal of Applied Operational Research. 2013. 5(1): p. 14-21.

[17] Imanirad, R., 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. 2013, 5(2): p. 33-45.

[18] Imanirad, R., X.S. Yang, and J.S. Yeomans, *A Biologically-Inspired Metaheuristic Procedure for Modelling-to-Generate-Alternatives*. International Journal of Engineering Research and Applications. 2013, 3(2): p. 1677-1686.

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

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

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

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

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