Soft Computing Strategies in Multiobjective Optimization

ALBEANU, Grigore
Faculty of Mathematics and Informatics
Spiru Haret University
g.albeanu.mi@spiruharet.ro

BURTSCHY, Bernard
Informatique et Réseaux, Paris, France
Telecom ParisTech
bernard.burtschy@telecom-paristech.fr

POPENŢIU-VLĂDICESCU, Florin
University of Oradea, "UNESCO Chair" Department
Romanian Academy of Scientists, Bucharest, Romania
popentiu@imm.dtu.dk

Abstract

Traditional optimization methods applied to solve real life problems can experience local or unstable behaviour. Researchers have proposed recently non-traditional search and optimization methods based on natural phenomena like natural evolution, social behaviour, and annealing. This paper describes a meta-algorithm for multiobjective optimization. Finally, it is remarked that migration algorithms and bees' algorithms are good candidates for solving multiobjective optimization problems.

Keywords: *multiobjective optimization, evolution strategies, natural evolution, social behaviour*

AMS Classification: 90C99

1. Introduction

Traditional optimization methods applied to solve real life problems can experience local or unstable behavior [8]. Researchers have proposed recently non-traditional search and optimization methods based on natural phenomena [5, 13, 14, 15]: **the annealing in metallurgy** – *Simulated Annealing*; **the process of natural evolution** – *Genetic Algorithms*, *Differential Evolution*, *Tabu Search*, *Scatter Search*, and *Self-Organizing Migrating Algorithms*; **nature inspired social behaviour** – *Particle Swarm Optimization* and *Ant Colony Optimization*. These algorithms have been used successfully for solving both the single and multiobjective optimization problems. This study

considers natural evolution and social behaviour approaches in order to identify suitable procedures for multiobjective optimization.

2. **Natural evolution inspired approaches**

Evolution strategies are local search methods for continuous search spaces using simple (individual) or advanced (population) solutions [10]. When using only one parent the strategy produces one offspring by adding randomly created values with zero mean and identical standard deviation. The resulting individual is evaluated and compared to the original solution (using the values of the objective function) and the better one survives to be used for the creation of the next solution. The advanced version use m solutions (individuals) as parent population. A set of n new solutions (offspring population) is created in iteration. The next parent population is creating by choosing the m best individual from the union of the parent and offspring population, or only from the offspring population when n is greater than m. Two mechanisms are used. The *selection* mechanism has an intensifying character while *recombination* has both diversifying and intensifying character. Popular selection schemes are *proportionate selection* (the expected number of copies a solution has in the next population is proportional to its fitness) and *tournament selection* (a tournament between s randomly chosen different individuals is held and the one with the highest fitness is added to the *mating pool* $\mathbf{P}$. After K tournaments of size s the mating pool is filled. The mating pool $\mathbf{P}$ consists of all solutions which are chosen for recombination).

Genetic operators and migration loops are some examples of evolution strategies. A genetic based searching mechanism, as evolutionary procedure, starts with a set of solutions called *population*. One solution, in this set, is called a *chromosome*. The search is guided by a survival of *the fit-test principle*. The search proceeds for a number of generations. For each generation, based on the fitness function, the fit-test solution will be selected to form a new population. During the process, three main operators can be applied: *reproduction*, *crossover*, and *mutation*. Reproduction process consists in the combination of the evaluation and selection activities. It copies an individual from one generation to the next. *Crossover* takes two or more chromosomes and by swapping information between them will produce one or more chromosomes (the so called *children*). *Mutation* is the process that randomly modifies a part of chromosome's information. The whole cycle can be repeated along a number of generations until certain termination criteria are met (the number of generations is maximal admitted, a high fitness value is obtained or, the simulation time hits its upper boundary). This chain has to be adapted depending on the particular problem to be solved [15].

A special case of evolution strategy is based on *migration* [16] and is incorporated in *self-organizing migrating algorithms* (SOMAs) which work on a population of candidate solutions in loops called *migration loops*. The initial population is a random sampling over the search space. In each loop, the population is evaluated and the solution with the highest fitness becomes the

leader L. In one migration loop, all individuals, apart from the leader, traverse the input space in the direction of the leader. For mutation, SOMA uses a parameter to achieve perturbation, this parameter being created before an individual starts its migration over the search space, and is used to define the final movement. SOMA approaches proved a good behaviour when used to solve real problems [2, 3, 4].

To solve $optf(x)$ where x belongs to D (the search space) by SOMA, a procedure consisting of five steps is required [10]:

1. Set up *the control parameters* of the algorithm: PS - the size of population, N - the dimension of the search space (the number of components of the solution), ST - the number of steps for one migration, PL - the path length (the distance toward leading item), the step size (the granularity of sampling the search space) $= PL/ST$, PP - the perturbation parameter, and ML - the number of migration loops;

2. Generate (using a uniform distribution over the search space) *the initial population*, and for each individual the objective functions are evaluated. The individuals are generated in a bounded domain according to

$$\lambda_{i,j}^0 = \lambda_j^{\min} + \alpha_{i,j}\left(\lambda_j^{\max} - \lambda_j^{\min}\right),$$

   where the j-th component belongs to $\left[\lambda_j^{\min}, \lambda_j^{\max}\right]$, $i = 1, 2, \ldots, PS$, and $\alpha_{i,j}$ is a uniform distributed number in the unit interval $[0, 1]$.

3. According to the values of the objective functions the solution with the highest fitness becomes *the leader* L, and the individuals (others than leader) are migrated to new positions, traversing the input space in the direction of the leader; any individual will travel, in a number of steps, a certain distance (path length) towards the leader (*AllToOne* strategy) or towards all individuals (*AllToAll* strategy). The operators used during migration are *perturbation* (a special kind of mutation), and *movement* (not a genetic crossover but in place of). Perturbation depends on a vector β generated according to the following rule: if $\gamma_j < PP$ then $\beta_j := 1$ else $\beta_j := 0$ $(j = 1, \ldots, N)$. The perturbation vector β is generated before the individual starts to migrate, with γ_j uniform distributed in $[0, 1]$. The movement operator (of the individual λ_p towards the individual λ_q), during the ith migration loop, produces an individual denoted by $\lambda_{\tau(s)}$ and defined, for every $(j = 1, 2, \ldots, N)$, as:

$$\lambda_{\tau(s),j}(i) = \lambda_{p,j}(i-1) + \beta_j(\lambda_{q,j}(i-1) - \lambda_{p,j}(i-1))\frac{s}{ST}PL,$$

   where s is the index of the movement step.

4. Test for the termination condition (the maximum number of migration loops ML is reached; no improvements during last migration step, or no significant improvement comparing against the leader), and if necessary continues with step 3;

5. Output the set of solutions.

A characteristic of the algorithm consists in *the memory property of individuals*: during a movement, each individual remembers the best found position. The advantage of using operators like perturbation and movement is to traverse the input space in the direction of the leader(s) to avoid the deadlock of the algorithm in the local optimum, and reach the global optimum by faster convergence [17, 18].

3. **Nature inspired approaches**

Social optimization strategies are inspired by the observation of the behaviour of swarms. Simple individuals cooperate through self-organization, without any form of central control. The *Bees Algorithm* (BA) approach is used to illustrate the social optimization paradigm [9, 13]. The algorithm performs a kind of *neighbourhood search* combined with *random search*, according to the following steps:

0. Let t = 0

1. Initialize population with random solutions: The algorithm starts with the NSB scout bees being placed randomly in the search space:

$$\{X_i(t)|i = 1, 2, \ldots, NSB\}\,.$$

2. Evaluate the fitness of the population and chose the best individual $X_{\text{best}}(t)$.

3. Repeat the following steps:

   3.1. Select the m best sites for neighbourhood search.

   3.2. Select e sites from m having the best quality.

   3.3. Each site of the e sites conducts neighbourhood searching for nep times assisted by nep bees.

   3.4. Each site of the $m - e$ sites conducts neighbourhood searching for nsp times assisted by nsp bees.

   3.5. Each site of the $NSB - m$ conducts search only one time.

   3.6. Recombine a new population.

   3.7. Evaluate the fitness of the population and chose the best individual $X_{\text{best}}(t)$.

   3.8. Let $t := t + 1$ until the stopping criterion is fulfilled.

4. Output $X_{\text{best}}(t)$, and the fitness of $X_{\text{best}}(t)$.

with the following parameters: number of scout bees (NSB), number of sites selected out of NSB visited sites (m), number of best sites out of m selected sites (e), number of bees recruited for best e sites (nep), number of bees recruited for the other $(m-e)$ selected sites (nsp), initial size of patches (ngh) which includes site and its neighbourhood. A stopping criterion (the maximum number of iterations) is necessary.

If the size of the solution vector is D then

1. *The population is initialized by the following rule:* For $i := 1$ to NSB do for $j := 1$ to D do

$$X_{i,j}(t) = \left({}^{U}X_{i,j}(t) - {}^{L}X_{i,j}(t)\right) * \mathrm{RAND} + {}^{L}X_{i,j}(t),$$

where ${}^{L}X_{i,j}$ and ${}^{U}X_{i,j}$ are the lower, respective the upper limits for the jth component of the solution vector, and RAND is generated as uniformly distributed in $[0,1]$.

2. A new individual is generated according to the following steps (*AllToOne* strategy):

   a) $V_{i,j}(t) = X_{i,j}(t) + \alpha_{i,j}(X_{i,j}(t) - X_{k,j}(t))$, where $\alpha_{i,j}$ is random and uniformly distributed in $[-1,1]$, and k is the index of best individual, $k \neq i$, identified by computing the probabilities

$$\mathrm{prob}_l = \mathrm{Fit}_l / \sum_{j=1}^{NSB} \mathrm{Fit}_j,$$

   and choosing k as

$$\mathrm{prob}_k = \max \mathrm{prob}_l | l = 1, 2, \ldots, NSB.$$

   b) If the fitness of V_i is higher than the fitness of X_i then X_i is replaced by V_i, otherwise remains unchanged.

4. Multiobjective evolutionary optimization

The multi-objective optimization is the process of simultaneously optimization of two or more conflicting objectives subject to certain constraints. A collection of case studies is given in [1]. Constructing a single aggregate objective function is the basic approach. If the objective functions have different weights then the decision is essentially subjective. Classical and non-traditional optimizers were proposed for multiobjective optimization. Currently most evolutionary optimizers apply Pareto-based ranking schemes. The concept of Pareto dominance is of extreme importance in multiobjective optimization, especially where some or all of the objectives are mutually conflicting: maximize the system reliability under minimum costs. The Pareto

optimum, in general, gives not a single solution, but a set of solutions. Such set is maintained using a database as described below. In the following, a meta-algorithm for evolutionary multiobjective optimization is described and experimental results are reported.

4.1. *A meta-algorithm*

The multi-objective version of evolutionary algorithms uses the concept of the Pareto domination [9]: *a solution* A *is said to dominate another solution* B *if* A *is not worse than* B *in all objectives, and* A *is strictly better than* B *in one objective at least.* In order to manage the set of the non-dominated solutions, a *database* **DB** should be used [6, 10]. The multiobjective procedure is based on the following steps:

1) Setup the parameters of the algorithm (the size of **DB**, the maximum number of evolution loops).

2) Generate the initial population and evaluate the objective functions for each individual.

3) Search the current population for non-dominated solutions and register them in **DB**.

4) Apply the *AllToMany* evolutionary procedure for every individual from the current generation, evaluate the new solutions and update **DB**.

5) Test for the termination condition, and if necessary continues with step 4.

6) Select the non-dominated optimal solution from **DB**.

The new individuals are generated according to the following rule (*AllToMany* strategy):

$$V_{i,j}(t) = X_{i,j}(t) + \alpha_{i,j}(X_{i,j}(t) - X_{k,j}(t)),$$

where $\alpha_{i,j}$ is random and uniformly distributed in $[-1, 1]$, and k is the index of individuals stored in **DB**. If the fitness of V_i is higher than the fitness of X_i then X_i is replaced by V_i, otherwise remains unchanged.

The above algorithm can use both **SOMA** and **BA** approaches. However, other recombination strategies can be considered when inspiring according to [7].

4.2. *Experiments*

In order to compare **SOMA** and **BA** algorithms the intuitionistic fuzzy optimization problem described in [10, 11] was considered firstly in the software reliability optimization framework. Starting from the architectural graph,

and using the intuitionistic fuzzy computation of the system reliability [12] and its optimization, the results demonstrate the efficiency of both approaches, the convergence being achieved after a small number of steps. However, the tuning of the **BA** parameters requested more time than tuning **SOMA** parameters.

5. Conclusion

Recently, an increased interest in evolutionary optimization inspired by nature was identified. Both migrating strategy algorithms and bees' algorithms are *controlled random search* approaches, the control being realized over generations of individuals obtained by recombination rules. This paper has described these approaches and presented a *meta-algorithm* for multiobjective optimization. Even difficult to calibrate the initial parameters, the algorithm proved a good behaviour when applied for real industrial problems concerning the complex systems reliability optimization.

References

1. Binh, T.T., *A Multiobjective Evolutionary Algorithm. The Study Cases*, "Technical Report, Institute for Automation and Communication", Berleben, Germany, 1999.
2. Coelho, L.S., *Self-Organizing Migrating Strategies Applied to Reliability-Redundancy Optimization of Systems*, "IEEE Transactions on Reliability", 58(3), pp. 501–510, 2009.
3. Coelho, L.S., *Self-Organizing Migration Algorithm Applied to Machining Allocation of Clutch Assembly*, "Mathematics and Computers in Simulation", 80, pp. 427–435, 2009.
4. Coelho, L.S. and Alotto, P., *Electromagnetic Optimization Using a Cultural Self-Organizing Migrating Algorithm Approach Based on Normative Knowledge*, "IEEE Transactions on Magnetics", 45(3), pp. 1446–1449, 2009.
5. Gonzalez, T.F., *Handbook of Approximation Algorithms and Metaheuristics*, "Chapman & Hall/CRC", 2007.
6. Kadlec, P. and Raida, Z., *A Novel Multi-Objective Self-Organizing Migrating Algorithm*, "Radioengineering" 20(4), 804–816, 2011.
7. Konak, A., Coit, D.W. and Smith, A.E., *Multiobjective Optimization Using Genetic Algorithms. A Tutorial*, "Reliability Engineering and System Safety", 91, pp. 992–1007, 2006.
8. Liu, G.P., Yang, J.B. and Whidborne, J.F., *Multiobjective Optimisation and Control*, "Research Studies Press LTD", 2003.
9. Li, H., Liu, K. and Li, X., *A Comparative Study of Artificial Bee Colony Bees Algorithms and Differential Evolution on Numerical Benchmark Problems*, In: "Cai, Z., Tong, H., Kang, Z., Liu, Y. (eds), Computational Intelligence and Intelligent Systems: Proceedings of ISICA 2010", Springer, 2010.

10. Madsen, H., Albeanu, G., Popentiu-Vladicescu, F. and Albu, R.-D., *Optimal Reliability Allocation for Large Software Projects through Soft Computing Techniques*, "Proceedings of PSAM 11 & ESREL 2012", 25-29 June 2012, Helsinki, Finland. 2012.
11. Madsen, H., Albeanu, G. and Popentiu-Vladicescu, F., *An Intuitionistic Fuzzy Methodology for Component-Based Software Reliability Optimization*, "International Journal of Performability Engineering", 8(1), pp. 67–76, 2012.
12. Mahapatra, G.S., *Reliability Optimization in Fuzzy and Intuitionistic Fuzzy Environment*, "Bengal Engineering and Science University, PhD Thesis", 2009.
13. Pham, D.T., Ghanbarzadeh, A., Koş, E., Otri, S., Rahim, S. and Zaidi, M., *The Bees Algorithm – A Novel Tool for Complex Optimisation Problems*, In: "Proceedings of IPROMS 2006", pp. 454-461, 2006.
14. Rothlauf, F., *Design of Modern Heuristics: Principles and Application*, "Springer", Berlin, Heidelberg, 2011.
15. Spears, W.M., *Evolutionary Algorithms. The Role of Mutation and Recombination*, "Springer", 2000.
16. Zelinka, I. and Lampinen, J., *SOMA – Self Organizing Migrating Algorithm*, "Proceedings of the 6th International Conference on Soft Computing (Mendel 2000)", Brno, Czech Republic, 2000, pp. 177-187.
17. Zelinka, I., Lampinen, J. and Noulle, L., *On the Theoretical Proof of Convergence for a Class of SOMA Search Algorithms*, "Proceedings of the 7th International Conference on Soft Computing (Mendel 2001)", Brno, Czech Republic, 2001, pp. 103-110.
18. Zelinka, I., *SOMA – Self-Organizing Migrating Algorithm*, In New optimization techniques in engineering, G.C. Onwubolu and B.V. Babu (Eds), "Springer", Ch7, 2004.