Genetic algorithm for multi-objective experimental optimization

Hannes Link · Dirk Weuster-Botz

Received: 11 September 2006 / Accepted: 20 September 2006 / Published online: 18 October 2006
© Springer-Verlag 2006

Abstract A new software tool making use of a genetic algorithm for multi-objective experimental optimization (GAME.opt) was developed based on a strength Pareto evolutionary algorithm. The software deals with high dimensional variable spaces and unknown interactions of design variables. This approach was evaluated by means of multi-objective test problems replacing the experimental results. A default parameter setting is proposed enabling users without expert knowledge to minimize the experimental effort (small population sizes and few generations).

Keywords Experimental design · Genetic algorithm · Multi-objective optimization · Software tool

Introduction

In many technical problems scientists face the problem of identifying optimal process conditions like pH, temperature, concentrations or other variables. A typical example is fermentation medium development. The composition of a fermentation medium consisting of carbon sources, nitrogen sources, mineral salts, trace elements, amino acids and/or peptides, vitamins and other growth factors determines the chemical and nutritional environment of cells in a bioreactor and is thus vital for the effective manufacturing of bioproducts.

Due to the large number of process variables and the metabolic complexity of microorganisms or cells methods of experimental design need to be applied in order to identify values of the relevant variables resulting in an improved performance. Considering several objectives to be optimal there is per definition not one optimal solution but a set of efficient solutions. Mathematically such a multi-objective optimization problem (MOP) can be described in terms of Eq. 1.

$$\begin{aligned} \min/\max \mathbf{y} &= \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_n(\mathbf{x})) \\ \text{subject to } \mathbf{x} &= (x_1, x_2, \dots, x_m) \in X \\ \mathbf{y} &= (y_1, y_2, \dots, y_n) \in Y \end{aligned} \quad (1)$$

The functional relation between the m design variables and the objective functions ϕ does not necessarily need to be known, as $\mathbf{y}$ will be obtained from n experimental observations.

An important criteria for MOPs is that of Pareto optimality, shortly saying that a member of the efficient set is not dominated by any other. In Fig. 1 the principle of dominance is exemplified in the objective space, where for the case of maximizing two objectives the efficient solutions are indicated.

Classical statistical experimental design methods (Plackett–Burman Design, Response Surface Method) have drawbacks like screening for principal components and assuming an unimodal objective function, reviewed in [1].

More sophisticated stochastic search strategies like genetic algorithms (GAs) have grown in popularity since Rechenberg [2] and Holland [3] first published their work on this subject. GAs are based on evolutionary principles, encoding several sets of design

H. Link · D. Weuster-Botz (✉)
Lehrstuhl für Bioverfahrenstechnik,
Technische Universität München,
Boltzmannstr. 15, Garching 85748, Germany
e-mail: D.Weuster-Botz@lrz.tu-muenchen.de

Fig. 1 Efficient points (black) are not dominated by any other point. For a dominated point (white) there exist at least one other point that has greater values in both objectives y_1 and y_2

variables on binary strings (individuals in a population) which are processed by GA operators (crossover and mutation) throughout several generations. The principle “survival of the fittest” assures a convergence towards optimal values in the design variables with proceeding generations.

The first GA dealing with multiple objectives was the Vector Evaluated Genetic Algorithm (VEGA) proposed by Schaffer [4]. This multi-objective optimization strategy has already been applied successfully for experimental medium optimization in many cases [1]. The most recent published multi-objective GAs are the Non-Dominated Sorting Genetic Algorithm-II [5] and the Strength Pareto Evolutionary Algorithm (SPEA) [6]. The SPEAs main feature is processing two populations: besides a normal population an external population serves as a kind of archive, keeping track of the efficient set. Especially in the case of experimental design with a limited number of experiments the SPEA is supposed to have advantages compared to VEGA with respect to the experimental effort.

The new software tool with a genetic algorithm for multi-objective experimental optimization making use of SPEA will be outlined. The performance of GAME.opt will be evaluated with the help of special test-functions to achieve appropriate parameter settings for experimental design.

Methods

GAME.opt was developed with LabVIEW 7 (National Instruments). The Application Builder was used to create a stand alone application running under Windows platforms. Data handling is based on spreadsheet files, which can be accessed via a text editor or Microsoft Excel. GAME.opt will be made available on request by the Technical University of Munich.

As the core algorithm is based on the SPEA, fitness assignment, selection and clustering is described in detail by Zitzler and Thiele [6]. To abstract the algorithm: fitness assignment is either based on the strength of an individuals dominance or on the degree an individual is dominated by others. A mating pool equal to the population size is filled by selecting individuals using binary tournament without replacement [7]. Crossover is performed between two parental binary strings from the mating pool at the specified number of crossover points in order to create two off-springs. Each bit of these new strings is inverted with a probability which is defined by the mutation rate. Clustering is used in order to bound the external population to a maximal value N_{extern} . Each time the external population exceeds this value it is pruned to a size of N_{cluster} by means of average clustering.

The first step when using GAME.opt is to create a new project where all information on the experimental design is defined. For this purpose the user is led through the steps explained together with some recommendations in the following.

Design variables

The design variables considered in the problem ($\mathbf{x}$ in Eq. 1) need to be defined initially. Alternatively to choosing the amount of bits encoding a variable GAME.opt gives the opportunity to choose the amount of levels, resulting from the variable's bounds and its increment. The choice of levels can be necessary for practical purposes but causes problems in the coding of a decision variable as follows. A general decoding function linearly transforms an n -bit-long binary string ($a_1 \dots a_n$) into a bounded real number $x \in [x^L, x^U]$:

$$x = \left(\frac{x^U - x^L}{2^n - 1} \right) \sum_{i=0}^{n-1} a_{(n-i)} 2^i + x^L \quad (2)$$

where a_1 is the most significant bit and a_n is the least significant bit [8]. This kind of transformation is just unique if the amount of levels equals $(2^n - 1)$. In other words, like its natural paradigm, the binary code is degenerated if there are more bit combinations than levels (the opposite case will not occur as the program computes the minimal amount of required bits).

Considering the above mentioned the following points are important when defining the design variables:

1. If the experimental setup prevents an adequate choice of levels the coding can be made visible in GAME.opt. In order to assure equal processing of

all variables the amount of levels among the variables should not vary too much.

2. From the amount of levels results the length of the binary string that is needed to encode the design variables, which is shown in GAME.opt in terms of bits for each variable. From a set of M binary strings with the length l each point in the search space (X in Eq. 1) can be reached with a probability p . The relation is given in Eq. 3 and GAME.opt will display p when selecting the number of individuals (M) [9].

$$p = (1 - 0.5^{M-1})^l \quad (3)$$

The value of p should be higher than 0.99.

3. As each individual corresponds to one experiment, the experimental effort will be correlated to the chosen amount of levels.
4. The increment should be higher than the possible precision in the experiment.

Experimental results

The experimental observations y have to be specified in GAME.opt just in terms of their names and units. The following items should be considered when planning the experiments:

1. All parameters different from the design variables have to remain fixed in all experiments throughout the optimization procedure.
2. The experimental error should be lower than the differences that are expected to result from different levels in the design variables.

Objective functions

The objective functions in GAME.opt can be any linear combination of the experimental results and the decision variables. GAME.opt will maximize all objective values. If for example one objective is minimizing a certain experimental result, the objective function is -1 times the according result. When defining the objective functions it is important to make sure that they are independent from each other. For example it is no use maximizing simultaneously the amount of a reaction's product and minimizing the amount of non-converted reactants at the end of a batch process, because a high product concentration is correlated to low resting concentrations of reactants. A reasonable combination is for example maximization of an experimental result while minimizing the

sum of decision variables, possibly reducing the costs of a process.

Optimization procedure

Once the MOP is defined by the above mentioned steps a random generated initial population is available. This set of design variables can be exported from GAME.opt and the according experiments are performed. After the experimental results are entered in GAME.opt the next population is generated and checked for individuals that have already appeared in order to prevent repeated experiments. This iterative process is performed until satisfying results are obtained. The external population contains the efficient set recovered so far and thus constitutes the result of the optimization process. The experimental results in the external population are already assigned because their members originate from former populations.

Results

A previous version of GAME.opt was used for experimental design for media optimization (13 medium components, 20 individuals/generation, 91 bits/binary string, eight generations) and gave comparable results to another GA making use of VEGA with half the number of experiments [10]. Further process optimizations considering media composition together with process parameters are currently performed in our laboratory. In order to evaluate the software for a proper processing and its general ability to solve MOPs it was tested by means of several multi-objective test problems in such a way that the experimental results are replaced by evaluation of mathematical functions. An exemplary MOP with three objectives and six design variables is given in Eq. 4. This MOP is based on Deb et al. [11], where the development of test problems is described.

$$\begin{aligned} \text{Minimize } f_1 &= (1 + g(\mathbf{x})) \cdot \cos(x_5\pi/2) \cdot \cos(x_6\pi/2) \\ \text{Minimize } f_2 &= (1 + g(\mathbf{x})) \cdot \cos(x_5\pi/2) \cdot \sin(x_6\pi/2) \\ \text{Minimize } f_3 &= (1 + g(\mathbf{x})) \cdot \sin(x_5\pi/2) \end{aligned} \quad (4)$$

$$\text{where } g(\mathbf{x}) = \sum_{i=1}^4 (x_i - 0.5)^2$$

subject to: $0 \leq x_i \leq 1$ for $i = 1, 2, \dots, 6$

The MOP in Eq. 4 was solved with GAME.opt, therefore the six design variables have been defined

in the program together with the bounds, being 0 and 1 for each variable. As mentioned above the choice of the increment and thus the amount of levels is crucial for a proper processing of the variables. An increment of 0.1 will give 11 Levels, namely [0, 0.1, 0.2, ..., 1] encoded with 4 bits giving 16 levels. According to Eq. 2 an adequate increment for a given number of bits n results from:

$$\text{incr} = \left(\frac{x^U - x^L}{2^n - 1} \right) \quad (5)$$

Table 1 confronts a proper coding with 4 bits with a degenerated coding for a variable range from 0 to 1.

Using the proper coding shown in Table 1 the GA was run eight generations with 20 individuals in the normal population and 10 individuals in the external population. Two crossover points and a mutation probability of 1% was chosen what has been identified as a reasonable default setting in other problems. As a rule of thumb the size of the external population should be half of the normal population, for which a minimal number can be estimated by Eq. 3. Table 2 summarizes the default parameter setting.

The results are shown in Fig. 2, by means of the objective values f_1 , f_2 , f_3 of the ten individuals in the external population after eight generations.

The two spherical surfaces in Fig. 2 constitute the objective space of Eq. 4.

For the minimization problem the first quadrant of the unit sphere constitutes the Pareto optimal frontier where $g(\mathbf{x}) = 0$ and it is already approximated well by

Table 1 Four bit binary coding of the design variables in Eq. 4 for different increments (incr)

Binary code	Degenerated coding Decoded value (incr = 0.1)	Proper coding Decoded value (incr = 1/15)
0 0 0 0	0	0
0 0 0 1	0.1	1/15
0 0 1 0	0.1	2/15
0 0 1 1	0.2	3/15
0 1 0 0	0.3	4/15
0 1 0 1	0.3	5/15
0 1 1 0	0.4	6/15
0 1 1 1	0.5	7/15
1 0 0 0	0.5	8/15
1 0 0 1	0.6	9/15
1 0 1 0	0.7	10/15
1 0 1 1	0.7	11/15
1 1 0 0	0.8	12/15
1 1 0 1	0.9	13/15
1 1 1 0	0.9	14/15
1 1 1 1	1	1

Table 2 Default parameter setting for the genetic algorithm

Parameter	Value
Crossover points	2
Mutation rate	1%
Individuals in normal population (N)	Eq. 3
Individuals in external population (N_{extern})	$N/2$
Individuals in external population after cluster analysis (N_{cluster})	$N/2$

the ten members of an external population after eight generations (Fig. 2a). Beside the closeness to the Pareto frontier the distribution of the solutions is an important performance criteria. Figure 2b reveals the equal distribution of the ten solutions resulting amongst other things from the cluster analysis. Optimal

Fig. 2 The objective space of the test problem lies between the surfaces. Optimal solutions are located on the unit sphere in the first quadrant. The black dots indicate the efficient set after eight generations of GAME.opt (external population). The two views of the plot point out the closeness of the efficient set to the Pareto front (a) and its equal distribution (b)

Fig. 3 Box plot of the initial population (a), the first external generation (b), the eighth generation (c) and the eighth external generation (d). The boxes contain the middle of 50% of the data. The line in the box indicates the median. The bold dashed line separates variables with an optimal solution ($x_1, x_2, x_3, x_4 = 0.5$) from variables without optimal value (x_5, x_6)

solutions are characterized by $x_i = 0.5$ for $i = 1, 2, 3, 4$, whereas x_5 and x_6 can take any value in the given range. Splitting the design variables in such a way allows to test the algorithm for a crucial feature of GAs, the populations diversity. On the one hand the algorithm is expected to converge fast to the optimal solutions but on the other the populations diversity must be high enough to ensure exploration of the whole Pareto optimal frontier. This fact is reflected in the box plots of the initial and the eighth generation together with their corresponding external populations in Fig. 3.

Initially the individuals are randomly distributed, what is reflected by the box plots of the six design variables in Fig. 3a. A trend towards 0.5 is already visible in Fig. 3b where box plots are shown for the first external population. In Fig. 3c, d, depicting the box plots of the corresponding populations after eight generations, a clear distinction between the variables with optimal value and those without can be made. After eight generations the diversity of x_5 and x_6 is still high in both populations. The remaining variables show the expected divergence towards 0.5 in the external population but remain in parts distributed over a wider range in the normal population. From this result it gets obvious that GAME.opt performs well with a small amount of individuals and a short evolu-

tion on a MOP considered a benchmark of evolutionary computing.

Conclusions

In order to keep the experimental effort low, GAME.opt will use small population sizes and fewer generations compared to other GA applications, where the amount of objective function evaluations is mostly limited by the computational costs. For this reason the external population is expected to be only a rough approximation of the true Pareto optimal frontier. Nevertheless in all test problems a satisfying performance of GAME.opt was achieved in spite of the relatively low number of experiments (160 compared to 16^6 possible design variable combinations with the test problem presented).

In the near future, efficient parallel search strategies for experimental design are becoming more and more important in bioprocess optimization due to the availability of new parallel stirred-tank bioreactor technologies [12].

Acknowledgment The many helpful discussions with Ezequiel Franco-Lara, Technical University of Braunschweig, Germany are gratefully acknowledged.

References

1. Weuster-Botz D (2000) Experimental design for fermentation media development: statistical design or global random search? *J Biosci Bioeng* 90:473–483
2. Rechenberg I (1973) *Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Friedrich Frommann Verlag, Stuttgart
3. Holland HJ (1975) *Adaption in natural and artificial systems, an introductory analysis with application to biology, control and artificial intelligence*. The University of Michigan Press, Ann Arbor
4. Schaffer JD (1985) Multi-objective optimization with vector evaluated genetic algorithms. In: *Proceedings of an international conference on genetic algorithms and their applications*, pp 93–100
5. Deb K, Agrawal S, Pratap A, Meyarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: *Proceedings of the Parallel Problem Solving from Nature*, vol VI, pp 849–858
6. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans Evol Comput* 3:257–271
7. Chakraborty M, Chakraborty UK (1997) An analysis of linear ranking and binary tournament selection in genetic algorithms. *Proc Int Conf Inf Commun Signal Process* 1:407–411
8. Kim JW, Kim SW, Park P, Park TJ (2002) On the similarities between binary-coded GA and real-coded GA in wide search space. In: *Proceedings of the 2002 Congress on evolutionary computation*, vol 1, pp 681–686
9. Reeves CR (1993) Using genetic algorithms with small populations. In: Forrest S (ed) *Proceedings of the 5th international conference on genetic algorithms*. Morgan Kaufmann Publisher, San Mateo, pp 92–99
10. Havel J, Link H, Hofinger M, Franco-Lara E, Weuster-Botz D (2006) Comparison of genetic algorithms for experimental multi-objective optimization on the example of medium design for cyanobacteria. *Biotechnol J* 1:549–555
11. Deb K, Thiele L, Laumanns M, Zitzler E (2002) Scalable multi objective optimization test problems. In: *Proceedings of the 2002 Congress on evolutionary computation*, vol 1, pp 825–830
12. Weuster-Botz D, Puskeiler R, Kusterer A, Kaufmann K, John GT, Arnold M (2005) Methods and milliliter scale devices for high-throughput bioprocess design. *Bioprocess Biosyst Eng* 28:109–119