

PARAMETER OPTIMIZATION OF FERMENTATION PROCESSES MODELS

Olympia N. Roeva

¹Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Sciences,
105 Acad. G. Bonchev Str., Sofia 1113, Bulgaria

Abstract

The paper presents the problems of real parameter optimization of fermentation process mathematical models. General and global optimisation methods are classified here and some short notes on the comparison of different methods are shown. A typical test function, as well as two mathematical models of E. coli MC4110 fed-batch fermentation processes are considered as case studies. Considering used test function the results show that the all optimization methods, except Genetic algorithms, get into the local minimum. Considering fermentation process models identification simulation results reveal that accurate and consistent results can be obtained only using genetic algorithms. The best result is obtained using the Multipopulation genetic algorithm.

Key words: fermentation process, optimization, genetic algorithms, mathematical models

1. INTRODUCTION

Within the broad field of nonlinear fermentation processes systems modelling and management studies, the objective of global optimization is the analysis and application of [35]:

- nonlinear decision models that (may) possess multiple optimal solutions;
- suitable solution algorithms that are able to find the absolutely best (global) solution(s).

The dynamic modelling of the nonlinear fermentation processes systems is formulated as a reverse problem, which requires a well-suited mathematical model and a very efficient computational method to achieve the model structure and parameters. Numerical integration for differential equations and finding global parameter values are still two major challenges in this field of the parameter estimation of nonlinear dynamic fermentation processes systems.

The usage of optimization in engineering is getting larger every day as the computational capabilities of the computers are increasing. Today calculations could be performed in a fraction of the time it took just a couple of years ago. Therefore, the applications for numerical optimization have increased dramatically. A great part of the design process is and will always be intuitive; however analytical techniques as well as numerical optimization could be of great value and can permit vast improvement in designs. Real engineering design problems are generally characterized by the presence of many often conflicting and incommensurable objectives. This raises the issue about how different objectives should be combined to yield a final solution. There is also the question on how to search for an optimal solution to the design problem. This chapter presents a survey of some methods and techniques to conduct numerical optimization in a parameter estimation context and different ways of developing hybrids among them.

2. METHODOLOGY

2.1. Problem formulation

2.1.1. Basic ingredients

Optimization problems are made up of three basic ingredients, namely an objective function, a set of unknowns or variables and a set of constraints [49].

1. An **objective function** which should be minimized or maximized. For instance, in fitting fermentation experimental data to a defined model, it might be to minimize the total deviation of observed data from predictions based on the model. Almost all optimization problems have a single objective function. The two interesting exceptions are [49]: **No objective function** and **Multiple objective functions**.

2. A set of **unknowns** or **variables** which affect the value of the objective function. In the manufacturing problem, the variables might include the amounts of different resources used or the time spent on each activity. In fitting-the-data problem, the unknowns are the *parameters* that define the fermentation process model.

3. A set of **constraints** that allows the unknowns to take on certain values but exclude others. For the problem of fitting experimental data, it does not make sense to have a negative rate of growth or negative production rate, so all the "time" variables have to be non-negative.

The optimization problem is then [49]: **Find values of the variables that minimize or maximize the objective function while satisfying the constraints.**

2.1.2. Complexity of the problem

The complexity in solving fermentation models parameter estimation problems is dependent on the following features [45, 46]: (i) The size p^* of region of attraction of the global minimum in relation to research region A ; (ii) The affordable number of function evaluations N_f ; (iii) Embedded or isolated global minimisers; (iv) The number of local minimisers. The discussion how these features influence on the complexity to obtain a solution is presented in [45, 46].

2.2. General optimization methods

As parameter estimation problems often consist of a mixture of numerical simulations, analytical calculations and catalogue selections, there is no easy way of calculating derivatives of the objectives function. Therefore, non-gradient optimization methods are better suited for these types of problems. The classical numerical approaches, due to their local search scope, are not directly applicable to solve general optimization problems. Therefore under such circumstances, it is essential to use a proper global search strategy. Note also that, instead of 'exact' solutions, most typically one has to accept diverse numerical approximations to the globally optimal solution (set) and optimum value [35].

Local search. Local search methods use local information, such as gradients and Hessian matrices, to generate iterative points and attempt to locate constrained local minima quickly [48]. Local search methods may not guarantee to find constrained local minima, and their solution quality is heavily dependent on starting points. These constrained local minima are constrained global minima [34] only if the objective function $f(x)$ is convex, every inequality constraint $g_i(x)$ is convex, and every equality constraint $h_i(x)$ is linear.

Global search. Global search methods employ local search methods to find constrained local minima and, as they get stuck at local minima, utilize some mechanisms, such as multistart, to escape from these local minima. Hence, one can seek as many local minima as possible and pick the best one as the

result. These mechanisms can be either deterministic or probabilistic and do not guarantee to find constrained global minima [48].

Global optimization. Global optimization methods are methods that are able to find constrained global minima of constrained nonlinear problems. They can either hit constrained global minima during their search or converge to constrained global minima when they stop. Global optimization is the task of finding the absolutely best set of parameters to optimize an objective function. In general, there can solutions that are locally optimal but not globally optimal [23, 33, 48].

Several classification schemes of the optimization methods could be used depending on which features that are used in the classification. For example the optimization methods could be classified as follows [14]: *Unconstrained* versus *Constrained methods*; *Derivative* versus *Non-derivative methods*; *Deterministic* versus *Stochastic methods*; *Local* versus *Global methods*; *Continuous* versus *Discrete methods*.

In the *Section 2.2.1* and *2.2.2* a short review of *Unconstrained* and *Constrained local search methods* are presented [3, 10, 14, 49]. A tentative classification of *Global optimization methods* are provided in the *Section 2.3* [14, 23, 32].

2.2.1. Unconstrained optimization

In the *unconstrained optimization problem*, a local minimiser of a real-valued function, $f(x)$, where x is a vector of n real variables is sought. In other words, a vector, x^* , such that $f(x^*) \leq f(x)$ for all x close to x^* is sought [10, 14, 18, 49].

Global optimization algorithms try to find an x^* that minimizes f over all possible vectors x . This is a much harder problem to solve. For many applications, local minima are good enough, particularly when the user can draw based on experience and provide a good starting point for the algorithm.

Newton method gives rise to a wide and important class of algorithms that require computation of the

gradient vector $\nabla f(x) = \begin{pmatrix} \partial_1 f(x) \\ \vdots \\ \partial_n f(x) \end{pmatrix}$ and the Hessian matrix, $\nabla^2 f(x) = (\partial_j \partial_i f(x))$ [24].

Newton method forms a quadratic model of the objective function around the current iterate x_k . The model function is defined by [18, 49]:

$$q_k(s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s \quad (1)$$

Convergence is guaranteed if the starting point is sufficiently close to a local minimiser x^* at which the Hessian is positive definite. Moreover, the rate of convergence is quadratic, that is, $\|x_{k+1} - x^*\| \leq \beta \|x_k - x^*\|^2$, for some positive constant β . In most circumstances, however, the basic *Newton method* has to be modified to achieve convergence. Some of these modified methods are [10, 18, 49]: *Line-search variant*, *Trust-region variant* and *Truncated Newton methods*.

So far, it is assumed that the Hessian matrix is available, but the algorithms are unchanged if the Hessian matrix is replaced by a reasonable approximation. Two kinds of methods use approximate Hessians in place of the real thing [18, 49]: *Difference approximations* and *Quasi-Newton methods*. Finally, two other approaches for unconstrained problems that are not so closely related to *Newton method* are mentioned [18, 49]: *Nonlinear conjugate gradient methods* and *the nonlinear Simplex method*.

Nonlinear least squares and *systems of nonlinear equations* are the two basic unconstrained algorithms [49].

Nonlinear Least Squares: The nonlinear least squares problem has the general form $\min \{r(x) : x \in \mathbb{R}^n\}$, where r is the function defined by $r(x) = \frac{1}{2} \|f(x)\|_2^2$ for some vector-valued function f that maps $\mathbb{R}^n$ to $\mathbb{R}^m$ [3, 28, 49]. Least squares problems often arise in data-fitting applications. Suppose that some physical or economic process is modelled by a nonlinear function φ that depends on a parameter vector x and time t . If b_i is the actual output of the system at time t_i , then the residual $\varphi(x, t_i) - b_i$ measures the discrepancy between the predicted and observed outputs of the system at time t_i . A reasonable estimate for the parameter x may be obtained by defining the i -th component of f by $f_i(x) = \varphi(x, t_i) - b_i$, and solving the least squares problem with this definition of f .

Some of the methods that used *nonlinear least squares* are [28, 49]: Gauss-Newton method, Levenberg-Marquardt method and Hybrid methods.

An algorithm that is particularly suited to the small-residual case is the *Gauss-Newton algorithm*, in which the Hessian is approximated by its first term. The *Levenberg-Marquardt algorithm* can be thought of as a trust-region modification of the *Gauss-Newton algorithm*. The *Levenberg-Marquardt algorithm* has proved to be an effective and popular way to solve nonlinear least squares problems. The *Gauss-Newton* and *Levenberg-Marquardt algorithms* usually exhibit quadratic convergence for zero-residual ($r(x^*) = 0$) problems. Otherwise, the convergence is only linear. It would seem that something could be gained by treating a nonlinear least squares problem as a general unconstrained minimization problem and applying *Quasi-Newton algorithms* to it, since *Quasi-Newton algorithms* are superlinearly convergent. For example, a simple hybrid strategy combines the *Gauss-Newton* and *BFGS Quasi-Newton algorithms* [18, 49].

Systems of nonlinear equations: *Systems of nonlinear equations* arise as constraints in optimization problems, but also arise, for example, when differential and integral equations are discretized. In solving a system of nonlinear equations, a vector such that $f(x) = 0$ where x is an n -dimensional – of n variables is sought. Most algorithms in this section are closely related to algorithms for unconstrained optimization and nonlinear least squares. Indeed, algorithms for systems of nonlinear equations usually proceed by seeking a local minimiser to the problem $\min \{\|f(x)\| : x \in \mathbb{R}^n\}$ for some norm $\|\cdot\|$, usually the 2-norm. This strategy is reasonable, since any solution of the nonlinear equations is a global solution of the minimization problem.

Some of the methods that solved nonlinear equations are [49]: *Trust Region* and *Line-search methods*, *Truncated Newton method*, *Broyden method*, *Tensor methods*, *Homotopy* and *Continuation methods*.

2.2.2. Constrained optimization

The general constrained optimization problem is to minimize a nonlinear function subject to nonlinear constraints [13, 14]. Two equivalent formulations of this problem are useful for describing algorithms [49]. They are:

$$\min \{f(x) : c_i(x) \leq 0, i \in \mathfrak{I}, c_i(x) = 0, i \in \mathfrak{N}\}, \quad (2)$$

where each c_i is a mapping from $\mathbb{R}^n$ to $\mathbb{R}$, and $\mathfrak{I}$ and $\mathfrak{N}$ are index sets for inequality and equality constraints, respectively;

and

$$\min \{f(x) : c(x) = 0, l \leq x \leq u\}, \quad (3)$$

where c maps $\mathbb{R}^n$ to $\mathbb{R}^m$, and the lower- and upper-bound vectors, l and u , may contain some infinite components.

Nonlinearly constrained optimization

The main techniques that have been proposed for solving nonlinear constrained optimization problems are [13, 14, 49]: *Reduced-gradient methods*, *Sequential linear* and *Quadratic programming methods*, *methods based on augmented Lagrangians* and *Exact penalty functions*.

The *Sequential quadratic programming (SQP) algorithm* is a generalization of *Newton method* for unconstrained optimization in that it finds a step away from the current point by minimizing a quadratic model of the problem. The convergence properties of the basic *SQP algorithm* can be improved by using a line search [13, 49]. *Augmented Lagrangian algorithms* are based on successive minimization of the augmented Lagrangian with respect to x . This approach is relatively easy to implement because the main computational operation at each iteration is minimization of the smooth Lagrangian function with respect to x , subject only to bound constraints [4, 49]. *Reduced-gradient algorithms* avoid the use of penalty parameters by searching along curves that stay near the feasible set. Essentially, these methods take the formulation (5.3) and use the equality constraints to eliminate a subset of the variables, thereby reducing the original problem to a bound-constrained problem in the space of the remaining variables [20, 49]. Finally, *Feasible sequential quadratic programming (FSQP) algorithms* is mentioned, which, as their name suggests, constrain all iterates to be feasible. They are more expensive than standard *SQP algorithms*, but they are useful when the objective function is difficult or impossible to calculate outside the feasible set, or when termination of the algorithm at an infeasible point is undesirable [49].

Bound-constrained optimization

Bound-constrained optimization problems play an important role in the development of software for the general constrained problem because many constrained codes reduce the solution of the general problem to the solution of a sequence of bound-constrained problems [49].

Algorithms for the solution of bound-constrained problems seek a local minimiser x^* of f . The standard first-order necessary condition for a local minimiser x^* can be expressed in terms of the *binding set* $B(x^*) = \{i : x_i^* = l_i, \partial_i f(x^*) \geq 0\} \cup \{i : x_i^* = u_i, \partial_i f(x^*) \leq 0\}$ at x^* by requiring that $\partial_i f(x^*) = 0, i \notin B(x^*)$.

Some of the algorithms for the solution of bound-constrained problems are [49]: *Newton methods* and *Gradient-projection methods*.

Quadratic programming

The quadratic programming problem involves minimization of a quadratic function subject to linear constraints [49]. Most codes use the formulation

$$\min \left\{ \frac{1}{2} x^T Q x + c^T x : a_i^T x \leq b_i, i \in \mathfrak{I}, a_i^T x = b_i, i \in \mathfrak{N} \right\}, \quad (4)$$

where $Q \in \mathbb{R}^{n \times n}$ is symmetric, and the index sets $\mathfrak{I}$ and $\mathfrak{N}$ specify the inequality and equality constraints, respectively.

The difficulty of solving the quadratic programming problem depends largely on the nature of the matrix Q . In *convex* quadratic programs, which are relatively easy to solve, the matrix Q is positive semidefinite. If Q has negative eigenvalues – *nonconvex* quadratic programming – then the objective function may have more than one local minimiser.

Linear programming

The basic problem of linear programming is to minimize a linear objective function of continuous real variables, subject to linear constraints [14, 49]. For purposes of describing and analyzing algorithms, the problem is often stated in the standard form $\min \{c^T x : Ax = b, x \geq 0\}$, where $x \in \mathbb{R}^n$ is the vector of unknowns, $c \in \mathbb{R}^n$ is the cost vector and $A \in \mathbb{R}^{m \times n}$ is the constraint matrix. The feasible region described by the constraints is a polytope, or *simplex*, and at least one member of the solution set lies at a vertex of this polytope [49]. The *Simplex algorithm*, so named because of the geometry of the feasible set, underlies the vast majority of available software packages for linear programming. However, this situation may change in the future, as more software for *Interior-point algorithms* becomes available [49].

The *Simplex method* generates a sequence of feasible iterates by repeatedly moving from one vertex of the feasible set to an adjacent vertex with a lower value of the objective function $c^T x$. When it is not possible to find an adjoining vertex with a lower value of $c^T x$, the current vertex must be optimal, and termination occurs. Algebraically speaking, the simplex method is based on the observation that at least $(n - m)$ of the components of x are zero if x is a vertex of the feasible set. Accordingly, the components of x can be partitioned at each vertex into a set of m basic variables – all nonnegative – and a set of $(n - m)$ nonbasic variables – all zero. If the basic variables are gathered into a subvector, $x_B \in \mathbb{R}^m$, and the nonbasics into another subvector, $x_N \in \mathbb{R}^{n-m}$, the columns of A could be partitioned as $[B|N]$, where B contains the m columns that correspond to x_B [49]. Interior-point products have emerged and have proven to be competitive with, and often superior to, the best simplex packages, especially on large problems. Recent research has shown the *Primal-dual interior-point algorithms* to be the most promising from a practical point of view, as well as the most amenable to theoretical analysis [49].

Semidefinite programming

The (linear) semidefinite programming problem (SDP) is essentially an ordinary linear program where the nonnegativity constraint is replaced by a semidefinite constraint on matrix variables [49]. The standard form for the primal problem is $\min C \sqcap X$ subject to $A_k \sqcap X = b_k, (k = 1, \dots, m), X \geq 0$, where C, A_k and X are all symmetric $n \times n$ matrices, b_k is a scalar, and the constraint $X \geq 0$ means that X , the unknown matrix, must lie in the closed, convex cone of positive semidefinite. Here, $\sqcap$ refers to the standard inner product on the space of symmetric matrices, i.e. for symmetric matrices A and B ,

$A \sqcap B = \text{trace}(AB)$. The dual version of the problem is $\max b^T y$ subject to $\sum_{k=1}^m y_k A_k + Z = C, Z \geq 0$,

where the dual variable y is an m -vector of Lagrange multipliers corresponding to the equality constraints in the primal, and Z is a symmetric $n \times n$ dual slack matrix.

2.3. Global optimization methods

To solve the global optimization (GO) problem in a strict mathematical sense means to find the complete set of globally optimal solutions x^* , and the associated global optimum value $f^* = f(x^*)$. In most cases this ‘ambitious’ objective is needed to be replaced by finding a verified estimate of f^* , and corresponding approximation(s) of points from the set x^* . Note that in many practical contexts the set of global optimizers consists (only) of a single point, or of a finite collection of points. On one hand, a very general search strategy should work for all cases covered – albeit its efficiency might be low for specialized problems. On the other hand, strictly specialized solvers will not work (as a rule) for problem-classes outside of their scope.

At present, there exist several main classes of algorithmic global optimization approaches that possess strong theoretical convergence properties. However, all such rigorous approaches have an inherent computational demand that increases exponentially, as a function of problem-size, even in case of the simplest *GO* problem instances. A brief review of elements of global optimization methods and a tentative classification of *GO* methods are provided below [23, 32]. Note that the list is not complete and its items are not necessarily mutually exclusive.

2.3.1. Elements of global optimization methods

All methods will evaluate the function $f(x)$ in some points $x_1, \dots, x_n$ in A , and they differ only in their choice of these points.

Strategies in choosing points. Due to there is no information about where in A to find the global minimum, one strategy that must be used is to spread out some of the points to cover A [45]. Any realization of this strategy is called a **global technique**. A possible global technique is uniform sampling. Any serious *GO* method must use some global technique. Given a point x it is normally possible to find a nearby point with a smaller function value. Any technique realizing this is called a **local technique**. A possible local technique is local optimization. Any serious *GO* method will use local optimization, at least to improve upon the estimates of the global minimum found.

Stopping conditions and solvability. In any computer algorithm there must be some stopping condition, which after some finite number of computing steps stops the computation. The condition should of course relate to the quality of the solution achieved. This is a very crucial point in global optimization. Without some additional information or assumptions about the problem there is no way to decide on the quality of the solution after a given number of steps (e.g. number of function evaluations made) [45].

Convergence with probability one. Because convergence is not in general possible, the ambition might be get lower and only to require convergence with probability 1. This is of course possible only for algorithms that can be made to run forever [45].

2.3.2. Global optimization strategies

Branch and bound

Branch and bound is a general search method [23, 25, 32, 35, 45, 46]. To apply branch and bound, one must have a means of computing a lower bound on an instance of the optimization problem and a means of dividing the feasible region of a problem to create smaller subproblems. There must also be a way to compute an upper bound (feasible solution) for at least some instances. For practical purposes, it should be possible to compute upper bounds for some set of nontrivial feasible regions.

Multistart and clustering methods

These methods are estimating the global minimum by finding local minima. Clustering global optimization methods can be viewed as a modified form of the standard multistart procedure, which performs a local search from several points distributed over the entire search domain. A drawback of multistart is that when many starting points are used the same local minimum may be identified several times, thereby leading to an inefficient global search. Clustering methods attempt to avoid this inefficiency by carefully selecting points at which the local search is initiated. Clustering methods are more efficient and robust than multistart methods because they try to identify the vicinity of local optima, thus increasing efficiency by avoiding the repeated determination of the same local solutions [23, 31, 45, 46, 47]

Evolutionary algorithms

Evolutionary algorithms (EA) are also known as biologically inspired methods, or population-based stochastic methods [9, 14, 23, 31, 35]. This is a very popular class of methods based on the ideas of

biological evolution [15], which is driven by the mechanisms of reproduction, mutation, and the principle of survival of the fittest. Similarly to biological evolution, evolutionary computing methods generate better and better solutions by iteratively creating new “generations” by means of those mechanisms in numerical form. Several different types of evolutionary search methods were developed independently. These include:

- *Genetic programming*, which evolve programs;
- *Evolutionary programming* [16], which focuses on optimizing continuous functions without recombination;
- *Evolutionary strategies* [5, 23, 35, 43], which focuses on optimizing continuous functions with recombination;
- *Genetic algorithms (GAs)* [22, 23, 26, 29, 35, 45, 46], which focuses on optimizing general combinatorial problems.

Evolutionary algorithms differ from more traditional optimization techniques in that they involve a search from a "population" of solutions, not from a single point. Each iteration of an *EA* involves a competitive selection that weeds out poor solutions. The solutions with high "fitness" are "recombined" with other solutions by swapping parts of a solution with another. Solutions are also "mutated" by making a small change to a single element of the solution [23].

Genetic algorithms

GAs and the closely related evolutionary strategies are a class of non-gradient methods. The basic idea of *GAs* is the mechanics of natural selection. Each optimization parameter, (x_n), is coded into a gene as for example a real number or string of bits. The corresponding genes for all parameters, $x_1, \dots, x_n$, form a chromosome, which describes each individual. A chromosome could be an array of real numbers, a binary string, a list of components in a database, all depending on the specific problem. Each individual represents a possible solution, and a set of individuals form a population. In a population, the fittest are selected for mating. Mating is performed by combining genes from different parents to produce a child, called a crossover. Finally the children are inserted into the population and the procedure starts over again, thus representing an artificial Darwinian environment, depicted in Fig. 1 below. The optimization continues until the population has converged or the maximum number of generations has been reached.

The popularity of *GAs* has grown tremendously under recent years and they have been applied to a wide range of engineering problems, see for example [1, 6, 7, 17, 32, 40, 41]. There is also an abundance of different type of *GAs* such as *simple GAs*, *steady state GAs*, *GAs with multiple populations* [36, 42], *GAs with crowding and sharing techniques*, and many, many more [2]. The different *GAs* have different features in order to solve different type of problems. *GAs* are very robust and can handle all type of fitness landscapes and mixture of real and discrete parameters [44].

Simulated annealing

Simulated annealing is another extremely popular class of methods [23, 31, 35, 45, 46]. These methods were created by simulating certain natural phenomena taking place in, for example, the cooling of metals, where atoms adopt the most stable configuration as slow cooling of a metal takes place [27]. The concept of the method is based on the manner in which liquids freeze or metals recrystallize in the process of annealing. In an annealing process a melt, initially at high temperature and disordered, is slowly cooled so that the system at any time is approximately in thermodynamic equilibrium. As cooling proceeds, the system becomes more ordered and approaches a "frozen" ground state at $T = 0$. Hence the process can be thought of as an adiabatic approach to the lowest energy state. If the initial temperature of the system is too low or cooling is done insufficiently slowly the system may become quenched forming defects or freezing out in metastable states (i.e. trapped in a local minimum energy state).

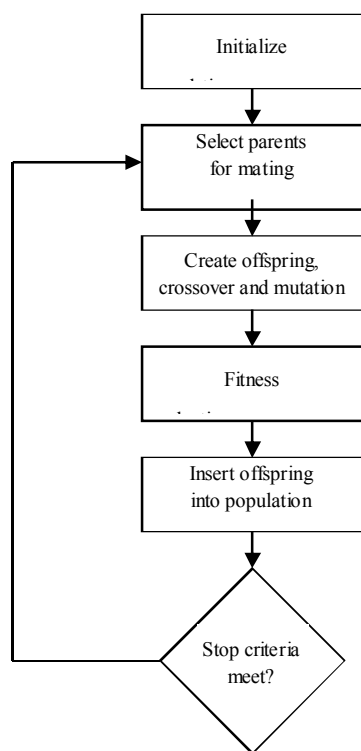


Fig. 1. An algorithm for a simple *GA*.

Other meta-heuristics

Several stochastic methods have been presented during recent years that are mostly based on other biological or physical phenomena, and with combinatorial optimization as their original domain of application. Examples of these more recent methods are *Tabu search (TS)*, *Ant colony optimization (ACO)*, and *Particle swarm methods* [31]. A thorough review of these and other recent techniques can be found in [8, 14].

Tabu search. The essential idea is to “forbid” search moves to points already visited in the (usually discrete) search space, at least for the upcoming few steps [23, 35]. In *TS* philosophy, one can temporarily accept new inferior solutions, to avoid paths already investigated. This approach can lead

to exploring new regions, with the goal of avoiding local minima and ultimately finding a solution by “globalized” search [21, 35].

Ant colony optimization. ACO studies artificial systems that take inspiration from the behaviour of real ant colonies and which are used to solve discrete optimization problems [11, 12]. Real ants are capable of finding shortest path from a food source to the nest without using visual cues. Also, they are capable of adapting to changes in the environment, for example finding a new shortest path once the old one is no longer feasible due to a new obstacle.

Adaptive stochastic search algorithms

This is another comprehensive class of approaches, based upon random sampling in the feasible set [14, 31, 37, 45, 46]. *Adaptive stochastic methods* (or *Adaptive random search*) were originally developed in the domains of electrical and control engineering and applied mathematics. In its basic form, it includes various random search strategies that are convergent (with probability one). Parameter adjustment, clustering and deterministic solution refinement options, statistical stopping rules, etc. can be added as enhancements.

Statistical global optimization algorithms

Statistical global optimization algorithms employ a statistical model of the objective function to bias the selection of new sample points. These methods are justified with Bayesian arguments that suppose that the particular objective function that is being optimized comes from a class of functions that is modelled by a particular stochastic function. Information from previous samples of the objective function can be used to estimate parameters of the stochastic function, and this refined model can subsequently be used to bias the selection of points in the search domain [23, 45, 46]. *Bayesian methods (P-algorithms)* [30] are based upon some chosen a priori information, to enable a stochastic description of the function-class modelled. During optimization, the problem-instance characteristics are adaptively estimated and updated. Note that, typically only the one-dimensional model development is exact; further, that in most practical cases “myopic” approximate decisions govern the search procedure [14, 23, 35, 45, 46].

Hybrid methods

There are a wide range of *hybrid global optimization algorithms* that have been developed. Some of *hybrid algorithms* that can be useful are [23, 50]:

- *Mixed Integer Nonlinear Programming (MINLP).* MINLP refers to mathematical programming algorithms that can optimize both continuous and integer variables, in a context of nonlinearities in the objective function and/or constraints [14, 23].
- *Tree Annealing. Simulated annealing* was designed for combinatorial optimization, usually implying that the decision variables are discrete [23]. A variant of SA called *Tree annealing* was developed to globally minimize continuous functions.
- *Pipelining hybrids.* The simplest and most straightforward way of implementing hybrids is to do in sequentially. First, one starts with exploring the whole search space with a method that is like to identify global optima but perhaps with slow convergence. After identifying promising regions, one could switch to a method with higher convergence rate in order to speed up the search.
- *Asynchronous hybrids.* In *Asynchronous hybrids* different methods work on different subsets of the solution space. The different methods might work on a subset of a shared population for some iterations. With this approach, a method that converges slowly could be combined with one that converges faster, or a method that performs well on one subset of the search space could be combined with method that performs well on another subset.

- *Hierarchical hybrids.* In *Hierarchical hybrids*, different optimization methods are applied at different levels of the optimization. On an overall level, you can apply a robust optimization strategy to find an optimal layout. At lower levels, where the system might be less complex and sensitive it might more appropriate to employ for instance pure analytical methods.
- *Additional operators.* In the literature they are many examples of hybrids between different optimization methods where operators from one method are added to or even replacing the standard operators of the other method. For instance, Yen et al. [50] have introduced a *Simplex method* as a new way of generating children in a *GA*. At each iteration a certain percentage of the population is generated by employing a simplex method on group of promising individuals. The rest of the population is generated using the usual genetic operators. Many variations of the *Simple GA*, as presented by Goldberg [22], can be found in the literature. All these modifications and hybridizations have been motivated by a desire to improve the performance of the *GAs*, and to adapt it to particular problem domains. It is known that in the *Simple GA* the best chromosome does not always keep on improving in each generation. A *Modified GA* that prevents this disadvantage is proposed in [38, 39].

3. RESULTS AND DISCUSSION

Parameter estimation problem is clearly about making many decisions often under uncertainty and with multiple conflicting criteria. In order to solve such a problem a variety of different optimization methods are available. As each optimization method has different properties suited for different type of problems, there is no simple answer to which method to use for a particular problem. It is all a matter of opinion; very much depending on the nature of the problem and the availability of different optimization software that fits the problem statement. However, if the problem is very complex with a mixture of continues and discrete variables and simultaneously there is no knowledge about the objective space, a robust method such as *GA* might be the best choice. As the design problem is multiobjective these objectives somehow has to be combined in order to yield one final solution.

In most comparison studies different methods come out on top depending on the problem and how well the different methods have been tuned to fit that particular problem. One should rightfully suspect that the optimization methods performance on one test problem might not be representative of their performance on other problems. It will therefore prove useful to have available not just one, but a set of test problems. The three test problems described below are very hard global optimization problems, on which most conventional optimization algorithms would fail.

The following optimization methods are compared: *Simple GA*; *Multipopulation GA* and *Modified GA*; as well as the methods available in *Matlab*: *Sequential quadratic programming method*; *Nelder-Mead Simplex method*; *BFGS Quasi-Newton method*; *Steepest descent* and *Minimax method*.

The *Matlab* implementation is considered of optimization procedures for all considered tests. The computations are performed using a PC/Pentium IV (3 GHz) platform running *Windows XP*. All methods are compared based on following criteria: number of iterations, CPU time, parameters value and function value.

3.1. Maximization a function of two variables

Test function, shown on Fig. 2, is a 2-D landscape to be maximized. It is defined by [7]:

$$f(x_1, x_2) = 0.8 \exp\left(-\frac{r_1^2}{(0.3)^2}\right) + 0.879008 \exp\left(-\frac{r_2^2}{(0.03)^2}\right), \text{ where} \quad (5)$$

$$r_1^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2, \quad r_2^2 = (x_1 - 0.6)^2 + (x_2 - 0.1)^2. \quad (6)$$

The maximum $f(x_1, x_2) = 1$ is at $(x_1, x_2) = (0.6, 0.1)$ and corresponds to the peak of the second, narrowed Gaussian. The test function is a hard global optimization problem. There are only two local maxima, with the global maximum covering about 1% of parameter space. For this function moving to the secondary maximum pulls solutions away from the global maximum.

Detailed results of function optimization are given in Table 1. Time variations of $f(x_1, x_2)$, as well as parameters variation are presented in Fig. 3 – Fig. 5.

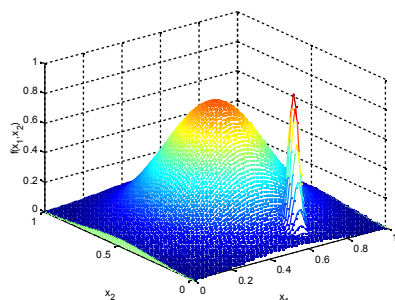
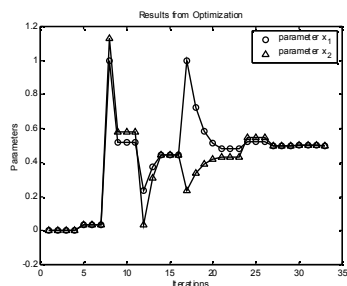


Fig. 2. Test function.

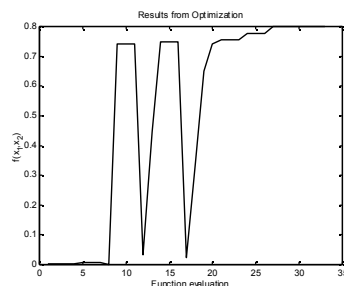
Table 1. Optimization results

Method	Number of iterations	CPU time, s	Starting point	Value		
				x_1	x_2	$f(x_1, x_2)$
Sequential quadratic programming	33	0.237	[0 0]	0.5	0.5	0.8
	33	0.219	[1 1]	0.5	0.5	0.8
Simplex method	68	0.547	[0 0]	0.5	0.5	0.8
	74	0.593	[1 1]	0.5	0.5	0.8
Steepest descent	27	0.441	[0 0]	0.5	0.5	0.8
	27	0.456	[1 1]	0.5	0.5	0.8
Quasi-Newton method	27	0.203	[0 0]	0.5	0.5	0.8
	27	0.250	[1 1]	0.5	0.5	0.8
Minimax method	55	0.437	[0 0]	0.5	0.5	0.8
	55	0.178	[1 1]	0.5	0.5	0.8
Simple GA	30	0.687	—	0.6	0.1	1.0
Multipopulation GA	30	1.594	—	0.6	0.1	1.0
Modified GA	30	0.609	—	0.6	0.1	1.0

As it can be seen from the results (Table 1) only *GAs* are able to find the global extremum. *Modified GA* finds the decision after total computation time of 0.609 s. The rest of methods get into the local extremum.

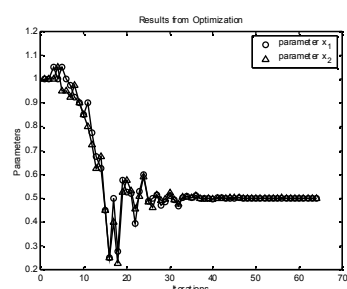


a) time variation of x_1 and x_2

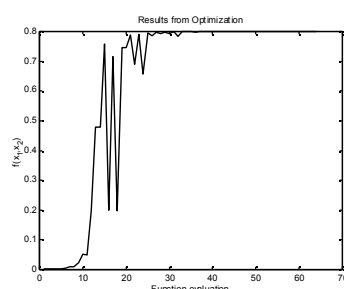


b) time variation of $f(x_1, x_2)$

Fig. 3. Sequential quadratic programming.

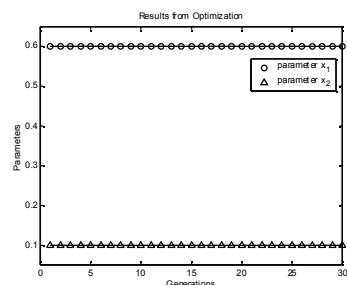


a) time variation of x_1 and x_2

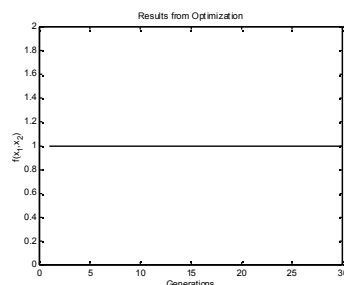


b) time variation of $f(x_1, x_2)$

Fig. 4. Simplex method.



a) time variation of x_1 and x_2



b) time variation of $f(x_1, x_2)$

Fig. 5. Modified GA.

3.2. Parameter estimation of a nonlinear fed-batch fermentation model

The mathematical formulation of a nonlinear fed-batch fermentation model is described according to the mass balance as follows:

$$\frac{dX}{dt} = \mu_{max} \frac{S}{k_s + S} X - \frac{F}{V} X \quad (7)$$

$$\frac{dS}{dt} = -\frac{1}{Y_{X/S}} \mu_{max} \frac{S}{k_s + S} X + \frac{F}{V} (S_{in} - S) \quad (8)$$

$$\frac{dV}{dt} = F, \quad (9)$$

where: X is biomass concentration, [g/l]; S – substrate concentration, [g/l]; F – feeding rate, [l/h]; V – bioreactor volume, [l]; S_{in} – substrate concentration in the feeding solution, [g/l]; μ_{max} – maximum value of the specific growth rate, [h^{-1}]; k_s – saturation constant, [g/l]; $Y_{S/X}$ – yield coefficient, [g/g].

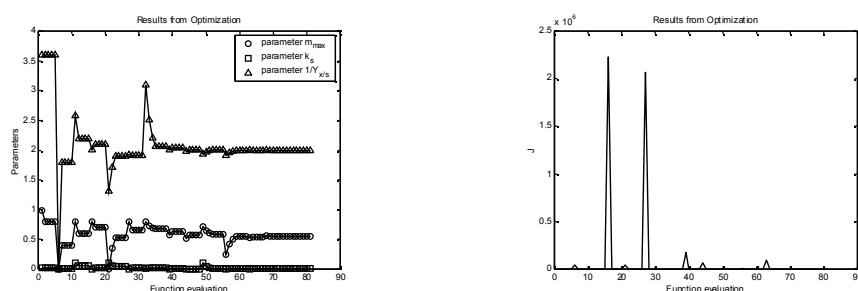
In this case generated data for state variables X and S are used. The data are obtained based on the following values of model parameters: $\mu_{max} = 0.55 \text{ h}^{-1}$, $k_s = 0.01 \text{ g/l}$, $1/Y_{S/X} = 2.00 \text{ g/g}$. Substrate concentration in the feed is 100 g/l. The initial values of variables are: $X(0)=1.25 \text{ g/l}$, $S(0) = 0.81 \text{ g/l}$ and $V(0) = 1.35 \text{ l}$.

Parameter estimation problem of presented nonlinear dynamic system is stated as the minimization of a distance measure J between generated and model predicted values of the considered state variables (X and S):

$$J = \sum_{i=1}^n [X_{gen}(i) - X_{mod}(i)]^2 + \sum_{i=1}^n [S_{gen}(i) - S_{mod}(i)]^2 \rightarrow \min, \quad (10)$$

where X_{gen} and S_{gen} are generated data; X_{mod} and S_{mod} are model predicted data.

Due to the solution quality of *Sequential quadratic programming*, *Simplex method*, *Steepest descent*, *Quasi-Newton* and *Minimax method* is heavily dependent on starting points, different starting points are here examined. Detailed results of parameter optimization are given in Table 2. Time variations of the criterion, as well as parameters variation are presented in Fig. 6 – Fig. 8. Figures present results using some of methods, namely *Sequential quadratic programming method*, *Simplex method* and *Quasi-Newton method*. The results show that the all methods, except *GAs*, get into the local minimum ($J = 3.3 \cdot 10^{-3}$). Still more there are conditions that some of methods could not find an appropriate or any decision (see Table. 2). In contrast to these results, the *GAs* find the decision with high accuracy. Using *Modified GA* the objective function obtain the value of $J = 1.007 \cdot 10^{-4}$.



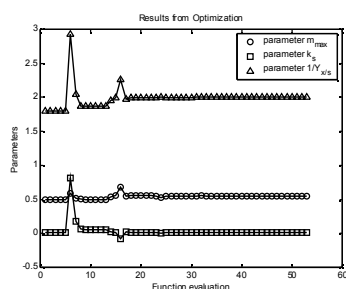
a) time variation of parameters

b) time variation of criterion

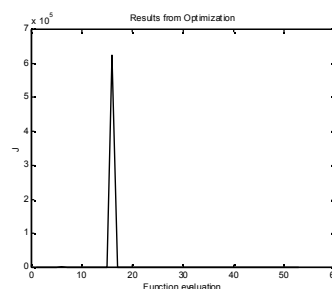
Fig. 6. Sequential quadratic programming (+80%).

Table 2. Optimization results

Method		Number of iterations	CPU time, s	J	Parameter value		
					μ_{max}	k_s	$1/Y_{S/X}$
Sequential quadratic programming	+80%	69	3.125	0.0033	0.5506	0.0101	1.9999
	-80%	65	3.000	0.0033	0.5506	0.0101	1.9999
Simplex method	+80%	269	12.000	0.0033	0.5506	0.0101	1.9999
	-80%	253	11.218	0.0033	0.5506	0.0101	1.9999
Steepest descent	+20%	—	—	—	—	—	—
	-20%	1502	67.031	0.0033	0.5506	0.0101	1.9999
	+80%	1505	126.891	1.58e+03	0.001e+03	0	0.004e+03
	-80%	—	—	—	—	—	—
Quasi-Newton method	+10%	1505	68.141	17.1009	0.4973	0.0015	2.0994
	-10%	40	1.829	0.0033	0.5506	0.0101	1.9999
	+80%	1504	864.359	88.2164	0.9256	-0.1101	1.8333
	-80%	—	—	—	—	—	—
Minimax method	+80%	99	4.656	0.0033	0.5506	0.0101	1.9999
	-80%	306	14.070	0.0033	0.5506	0.0101	1.9999
Simple genetic algorithm	—	200	81.359	2.161e-04	0.5495	0.0098	1.9999
Multipopulation GA	—	200	110.156	1.417e-04	0.5497	0.0101	1.9999
Modified genetic algorithm	—	200	75.000	1.007e-04	0.5499	0.0100	2.0000

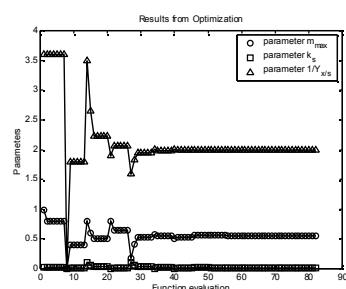


a) time variation of parameters

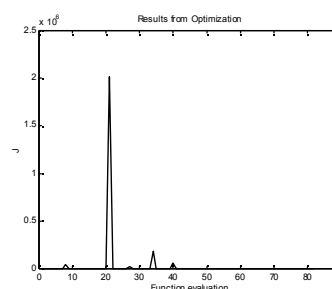


b) time variation of criterion

Fig. 7. Quasi-Newton method (-10%).



a) time variation of parameters



b) time variation of criterion

Fig. 8. Minimax method (+80%).

5.4.3. Parameter estimation of fed-batch fermentation model of *E. coli* MC4110

The cultivation condition of the fed-batch cultivation of *E. coli* and the experimental data are presented in [19]. The mathematical formulation of the nonlinear dynamic model of *E. coli* cultivation is described according to the mass balance as follows:

$$\frac{dX}{dt} = \mu_{max} \frac{S}{k_S + S} X - \frac{F}{V} X \quad (11)$$

$$\frac{dS}{dt} = -\frac{1}{Y_{S/X}} \mu_{max} \frac{S}{k_S + S} X + \frac{F}{V} (S_{in} - S) \quad (12)$$

$$\frac{dA}{dt} = \frac{1}{Y_{A/X}} \mu_{max} \frac{S}{k_S + S} X - \frac{F}{V} A \quad (13)$$

$$\frac{dV}{dt} = F, \quad (14)$$

where: X is biomass concentration, [g/l]; S – substrate concentration, [g/l]; A – acetate concentration, [g/l]; F – feeding rate, [l/h]; V – bioreactor volume, [l]; S_{in} – substrate concentration in the feeding solution, [g/l]; μ_{max} – maximum specific growth rate, [h⁻¹]; k_S – saturation constant, [g/l]; $Y_{S/X}$ and $Y_{A/X}$ – yield coefficients, [g/g].

As it shown, the model consisted of a set of four differential Eqs. ((11) - (14)) thus represented: three dependent state variables – $\mathbf{x} = [X \ S \ A]$ and four parameters – $\mathbf{p} = [\mu_{max} \ k_S \ Y_{S/X} \ Y_{A/X}]$. Parameter estimation problem of the system (11) - (14) is stated as the minimization between experimental and model predicted values of the considered state variables:

$$J = \sum_{i=1}^n [X_{exp}(i) - X_{mod}(i)]^2 + \sum_{i=1}^n [S_{exp}(i) - S_{mod}(i)]^2 + \sum_{i=1}^n [A_{exp}(i) - A_{mod}(i)]^2 \rightarrow \min \quad (15)$$

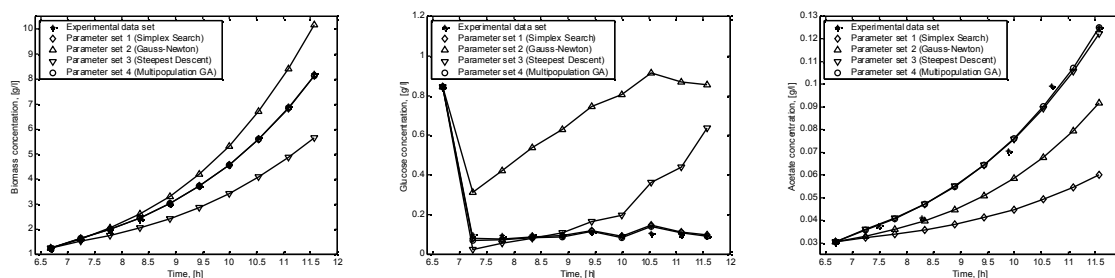
where N is the number of data for each state variable; X_{exp} , S_{exp} and A_{exp} represent the known experimental data; X_{mod} , S_{mod} and A_{mod} are model predictions with a given set of the parameters.

Off-line measurements of biomass and acetate concentrations and on-line measurements of the glucose concentration are used for the estimation of model parameters [19]. A parameter identification using *Multipopulation GA*, *Quasi-Newton method*, *Simplex Search* and *Steepest Descent methods* is performed. Detailed results for values of considered criteria (objective function value J , number of iterations, parameter values and CPU time) are given in Table 3. The model predictions of the state variables, based on four sets of search parameters are compared to experimental data points of *E. coli* cultivation. The simulation results are presented in Fig. 9.

The results show that the estimated values of the considered parameters are admissible [19]. In the most cases, graphical comparisons clearly show the existence or absence of systematic deviations between model predictions and measurements. It is evident that a quantitative measure of the differences between calculated and measured values is an important criterion for the adequacy of a model. The best result ($J = 5.2529$) is obtained using the *Multipopulation GA* after total computation time of 16.83 min. The second best search method is the *Simplex search method*, which converged to a value of $J = 5.9167$. This result is 13% worst than the result obtained using *Multipopulation GA*. Concerning biomass and glucose prediction once again it could be seen that the *Simplex search* has achieved very good results, close to *Multipopulation GA* results, but the prediction of acetate concentration is not satisfactory (see Fig. 9 c).

Table 3. Optimization results

Method	Number of iterations	CPU time, s	J	Parameter value			
				μ_{max}	k_S	$1/Y_{S/X}$	$1/Y_{A/X}$
Quasi-Newton method	766	2.6196e+003	2.3633e+003	0.47	0.014	0.67	0.0072
Simplex Search	248	0.8091e+003	5.9167	0.52	0.028	0.49	0.0048
Multipopulation GA	150	1.0098e+003	5.2529	0.53	0.029	0.49	0.014
Steepest descent	523	2.08871e+003	3.2895e+003	0.39	0.02	0.33	0.021



a) biomass concentration b) substrate concentration c) acetate concentration

Fig. 9. Time profiles of the state variables.

As it can be seen from the simulation results, the *Multipopulation GA* and the *Simplex search method* have achieved almost equal results about description of biomass and substrate variation (Fig. 9 a and b). However, the *Simplex search* could not find appropriate solution for acetate (Fig. 9 c). The results also reveal an interesting fact. The *Steepest descent* gives a satisfactory result for the acetate fit. Unfortunately this method is failed in the modelling of biomass and substrate. The presented graphical results show that only *Multipopulation GA* coping with the present problem. It is worth to note the very good correlation between the experimental and predicted data.

The presented solutions of considered three problems show that no one of the algorithms tested could reach the vicinity of the *GAs* solutions, especially for the problem of parameter estimation of fermentation processes models. This is a clear sign of the very challenging nature of these problems. Because of the nonlinear and constrained nature of the system dynamics, the parameter estimation problems are very often multimodal (nonconvex). Therefore, if this inverse problem is solved via standard local methods, such as the standard *Quasi-Newton method*, it is very likely that the solution found will be of local nature.

4. CONCLUSION

The concurrent nature implies that *GAs* are much more likely to locate a global peak than the traditional techniques. The conventional search methods work extremely well provided it knows where to start. In the problem considered here the proper initial values of parameters are unknown. Due to the parallel nature of the genetic algorithm, the performance is much less sensitive to the initial conditions. In fact, *GAs* make hundreds, or even thousands, of initial guesses. Compared with traditional optimization methods, *GA* simultaneously evaluates many points in the parameter space. It is more probable to converge towards the global solution. A *GA* does not assume that the space is differentiable or continuous and can also iterate many times on each data received. A *GA* requires only information concerning the quality of the solution produced by each parameter set (objective function value information). This characteristic differs from optimization methods that require derivative information or, worse yet, complete knowledge of the problem structure and parameters. Since *GAs* do not demand such problem-specific information, they are more flexible than

most search methods. Also *GAs* do not require linearity in the parameters which is needed in iterative searching optimization techniques. Simulation results reveal that accurate and consistent results can be obtained using *GAs*. The *GAs* property makes them suitable and more applicable for parameter estimation of fermentation processes models.

ACKNOWLEDGEMENTS:

The present work is supported by the Bulgarian National Science Fund (grants № DMU 02/4, 2009 and DID 02-29/2009).

REFERENCES

1. Alander, J. T. (1996) Genetic Algorithms and Other “Natural” Optimisation Methods to Solve Hard Problems – A Tutorial Review, available at <ftp.uwasa.fi/cs/report96-1>.
2. Bäck, T. (1993) An Overview of Evolutionary Algorithms for Parameter Optimisation, *Evolutionary Computation*, , 1, pp. 1-23.
3. Bates, D.M., D.G. Watts (1988) *Nonlinear Regression Analysis and Its Applications*, John Wiley&Sons, Inc., New York.
4. Bertsekas, D.P. (1982) *Constrained Optimisation and Lagrange Multiplier Methods*, Academic Press, New York.
5. Beyer, H.G., H.P. Schwefel (2002) *Evolution Strategies – A Comprehensive Introduction*, *Natural Computing*, 1, pp. 3-52.
6. Carrillo-Ureta, G.E., P.D. Roberts, V.M. Becerra (2001) Genetic Algorithms for Optimal Control of Beer Fermentation. *Proceedings of the 2001 IEEE International Symposium on Intelligent Control*, Mexico City, Mexico, pp. 391-396.
7. Charbonneau, P. (2002) *An Introduction to Genetic Algorithms for Numerical Optimisation*, NCAR Technical Note, Boulder, Colorado.
8. Corne, D., M. Dorigo, F. Glover (1999) *New Ideas in Optimisation*, McGraw-Hill, New York.
9. Dasgupta, D., Z. Michalewicz (1996) *Evolutionary Algorithms in Engineering Applications*, Springer-Verlag.
10. Dennis, J.E., R.B. Schnabel (1983) *Numerical Methods for Unconstrained Optimisation and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, NJ.
11. Dorigo, M., G. Di Caro (1999) The Ant Colony Optimisation Meta-heuristic, D. Corne, M. Dorigo, F. Glover (Eds), *New Idea in Optimisation*, McGraw-Hill, pp. 11-32.
12. Fidanova, S. (2002) Ant Colony Optimisation: Additional Reinforcement and Convergence. Tech. report IRIDIA-2002-30, Free university of Bruxelles, Belgium, pp. 12.
13. Fletcher, R. (1987) *Practical Methods of Optimisation*, 2nd edition, John Wiley & Sons, Inc., New York.
14. Floudas, C., P.M. Pardalos (2003) *Encyclopedia of Optimisation*, Kluwer Academic, available at <http://reference.kluweronline.com/>.
15. Fogel, D. B. (2000) *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, New York.

16. Fogel, L.J., A.J. Owens, M. J. Walsh (1966) Artificial Intelligence through Simulated Evolution, Wiley, New York.
17. Fonseca, C.M., P.J. Fleming (1998) Multiobjective optimisation and Multiple Constraint Handling with Evolutionary Algorithms – Part II: Application Example, IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 28, pp. 38-47.
18. Frandsen, P.E., K. Jonasson, H.B. Nielsen, O. Tingleff (2004) Unconstrained Optimisation, 3th edition, Informatics and Mathematical Modelling, Technical University of Denmark.
19. Georgieva, O., M. Arndt, B. Hitzmann (2001) Modelling of Escherichia coli Fed-batch Fermentation, International Symposium “Bioprocess Systems 2001 – BioPS’01”, Sofia, Bulgaria, October 1-3, pp. I.61-I.64.
20. Gill, P.E., W. Murray, M.H. Wright (1981) Practical Optimisation, Academic Press, New York.
21. Glover, F., M. Laguna (1993) Tabu Search, Modern Heuristic Techniques for Combinatorial Problems, C. R. Reeves editor, John Wiley & Sons, Inc.
22. Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimisation and Machine Learning, Addison Wesley Longman, London.
23. Gray P., W.E. Hart, L. Painton, C. Phillips, M. Trahan, J. Wagner (1998) A Survey of Global Optimisation Methods, Sandia National Laboratories, available <http://www.cs.sandia.gov/opt/survey/intro.html>.
24. Griewank, A. (2000) Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, SIAM, Philadelphia.
25. Hansen, E. (1992) Global Optimisation using Interval Analysis. Dekker, New York.
26. Holland, J.H. (1992) Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, Cambridge, MA.
27. Kirkpatrick, S., C.D. Gelatt, M.P. Vecchi (1983) Optimisation by Simulated Annealing, Science, 220, pp. 671-680.
28. Madsen K., H. B. Nielsen, O. Tingleff (2004) Methods for Non-Linear Least Squares Problems, IMM, DTU, available at <http://www.imm.dtu.dk/courses/02611/NLS.pdf>
29. Michalewicz, Z. (1994) Genetic Algorithms + Data Structures = Evolution Programs, Second, Extended Edition, Springer-Verlag, Berlin, Heidelberg.
30. Mockus, J., W. Eddy, A. Mockus, L. Mockus, G. Reklaitis (1996) Bayesian Heuristic Approach to Discrete and Global Optimisation, Kluwer Academic Publishers.
31. Moles, C.G., P. Mendes, J.R. Banga (2003) Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimisation Methods, Genome Research, 13, pp. 2467-2474.
32. Na, J.-G., Y.K. Chang, B.H. Chung, H.C. Lim (2002) Adaptive Optimisation of Fed-batch Culture of Yeast by using Genetic Algorithms, Bioprocess & Biosystems Eng, 24, pp. 299-308.
33. Neumaier, A., Global Optimisation Techniques, available at <http://www.mat.univie.ac.at/~neum/glopt/techniques.html>.
34. Nocedal, J., S. J. Wright (1999) Numerical Optimisation, Springer-Verlag New York, Inc.
35. [Pintér, J. D.](#), Continuous Global Optimisation: An Introduction to Models, Solution Approaches, Tests and Applications, available at <http://catt.bus.okstate.edu/itorms/pinter/paper.htm>

36. Pohlheim, H., The Multipopulation Genetic Algorithm: Local Selection and Migration, available at http://www.pohlheim.com/Papers/mpga_gal95/gal2_6.html.
37. Rastrigin, L. A., Y. Rubinstein (1969) The Comparison of Random Search and Stochastic Approximation while Solving the Problem of OPTIMISATION, Automat. Control, 2, pp. 23-29.
38. Roeva, O. (2005) A Modification of Simple Genetic Algorithms, International Symposium "Bioprocess Systems 2005 – BioPS'05", Sofia, Bulgaria, October 25-26, I.1-I.14.
39. Roeva, O. (2006) A Modified Genetic Algorithm for a Parameter Identification of Fermentation Processes, Biotechnology and Biotechnological Equipment, 20(1), pp. 202-209.
40. Roeva, O. (2003) Application of Genetic Algorithms in Fermentation Process Identification, Journal of the Bulgarian Academy of Sciences, CXVI, 3, pp. 39-43.
41. Roeva, O. (2005) Genetic Algorithms for a Parameter Estimation of a Fermentation Process Model: A Comparison, Bioautomation, 3, pp. 19-28.
42. Roeva, O., St. Tzonkov (2003) Parametric Identification of Fermentation Processes using Multy-population Genetic Algorithms, Technical Ideas, XL(3-4), pp. 18-26.
43. Schwefel, H. P. (1995) Evolution and Optimum Seeking, Wiley, New York.
44. Senin, N., D. R. Wallace, N. Borland (1999) Mixed Continuous and Discrete Catalogbased Design Modeling and Optimisation, CIRP International Design Seminar, University of Twente, Enschede, The Netherlands.
45. Törn, A., Global Optimisation, available at <http://www.abo.fi/~atorn/ProbAlg/Page510.html>.
46. Törn, A., M. Ali, S. Viitanen (1999) Stochastic Global Optimisation: Problem Classes and Solution Techniques, Journal of Global Optimisation, 14, pp. 437-447.
47. ToËrn, A. A. (1973) Global Optimisation as a Combination of Global and Local Search, Proceedings of Computer Simulation Versus Analytical Solutions for Business and Economic Models, Gothenburg, Sweden, pp. 191-206.
48. Wang, T. (2001) Global Optimisation for Constrained Nonlinear Programming, Thesis, University of Illinois at Urbana-Champaign.
49. What is Optimisation?, Optimisation Technology Center, 49 Guide, available at <http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/opt.html>.
50. Yen, J., L. Liao, B. Lee, D. Randolph (1998) A Hybrid Approach to Modeling Metabolic Systems using a Genetic Algorithm and Simplex Method, IEEE Transaction on Systems, Man and Cybernetics, 28, pp. 173-191.