Efficient Differential Evolution algorithms for multimodal optimal control problems

I.L. López Cruz*, L.G. Van Willigenburg, G. Van Straten
Systems and Control Group, Wageningen University, Manholtlaan 10, 6708 PA, Wageningen, Netherlands

Received 29 April 2002; received in revised form 24 August 2002; accepted 16 December 2002

Abstract

Many methods for solving optimal control problems, whether direct or indirect, rely upon gradient information and therefore may converge to a local optimum. Global optimisation methods like Evolutionary algorithms, overcome this problem. In this work it is investigated how well novel and easy to understand Evolutionary algorithms, referred to as Differential Evolution (DE) algorithms, and claimed to be very efficient when they are applied to solve static optimisation problems, perform on solving multimodal optimal control problems. The results show that within the class of evolutionary methods, Differential Evolution algorithms are very robust, effective and highly efficient in solving the studied class of optimal control problems. Thus, they are able of mitigating the drawback of long computation times commonly associated with Evolutionary algorithms. Furthermore, in locating the global optimum these Evolutionary algorithms present some advantages over the Iterative Dynamic Programming (IDP) algorithm, which is an alternative global optimisation approach for solving optimal control problems. At present little knowledge is available to the selection of the algorithm parameters in the DE algorithm when they are applied to solve optimal control problems. Our study provides guidelines for this selection. In contrast to the IDP algorithm the DE algorithms have only a few algorithm parameters that are easily determined such that multimodal optimal control problems are solved effectively and efficiently.

Keywords: Optimal control; Evolutionary algorithms; Differential Evolution algorithms; First-order gradient algorithm; Iterative Dynamic Programming

1. Introduction

Indirect numerical methods for optimal control based on Pontryagin’s minimum principle (PMP) use gradient information and local search methods. Therefore, if the optimal control problem is multimodal, convergence to a local optimum is likely. Deterministic direct methods for optimal control parameterize the controls and also use gradient information and local search methods to solve the resulting non-linear programming (NLP) problem. Consequently, they may also converge to a local solution. The simplest way to increase the chances of finding the global solution by these approaches is by repeating them several times with different control initialisations. Doing so, there still are optimal control problems that require a very close guess to the global optimum. To locate the global optimum or a sufficiently close approximation, global optimal control approaches are needed. An approximate global solution may be used to initialise a direct or indirect local optimisation method to obtain the global solution accurately.
Global optimisation methods or multimodal optimisation algorithms can roughly be divided into two general groups: deterministic [1–6] and stochastic [2,7–13]. Simulated annealing is another well-known global stochastic optimization approach [14,15]. Although Evolutionary algorithms (Genetic algorithms [16–18], Evolution Strategies [19,20], Evolutionary Programming [21]) can be considered as heuristic–stochastic global search methods, they have received limited attention among the global optimisation research community [22]. One possible explanation is the extended belief that all Evolutionary algorithms are not efficient enough to solve continuous global optimisation problems [23]. However, recently, new algorithms inspired by evolution have been developed that are rather different from a simple Genetic algorithm [24,25].

Moreover, though global optimisation methods generically are deterministic and stochastic, one can find that actually four approaches have been attempted in the solution of multimodal optimal control problems, namely, Deterministic algorithms, Stochastic algorithms, Iterative Dynamic Programming, and Evolutionary algorithms.

Rosen and Luus [26] proposed a method based on line search techniques to determine starting points for their NLP solver. Their approach fails when the problem is highly multimodal. Strekalovskiy and Vasilev [27] presented a search method for convex objectives but convergence of their approach is guaranteed for only a limited class of problems. Esposito and Floudas [28] have proposed a Deterministic algorithm to address the non-linear optimal control problem to global optimality. The approach is based on a branch and bound algorithm (cnBB). By using some benchmark problems from non-linear optimal control a theoretical guarantee of attaining the global optimum of the multimodal optimal control is offered as long as rigorous values of the parameters needed or rigorous bounds on the parameters are obtained. However, the issue of global optimality remains open. Papamichail and Adjiman [29] have proposed a new Deterministic global optimization algorithm for dynamic optimization problems, based on BB techniques for NLP problems. However, their algorithm has not been evaluated with highly multimodal optimal control problems as the authors acknowledged. One advantage of Deterministic algorithms is that convergence proofs can be presented. The same does not happen in case of probabilistic algorithms. One disadvantage is that it demands certain structure of the NLP problem in order to guarantee the global optimum is achieved.

Ali et al. [30] applied and evaluated the performance of several Controlled Random Search [8–10] and modified controlled random search algorithms [31] on the solution of a low multimodal optimal control problem. This class of algorithms is based on the use of a population of search points. Within this class, Banga and Seider [32] proposed the Integrated Controlled Random Search for Dynamic Systems Engineering. They applied their algorithm to some dynamic optimization problems of bioprocesses and bioreactors [33]. Carrasco and Banga [34] proposed another Stochastic algorithm for global optimization named Adaptive Randomly Directed Search for Dynamic Systems. Both algorithms use only one search point and utilize normal and uniform random distributions, respectively, in order to generate new solutions. The first algorithm did not achieve the global solution of a highly multimodal optimal control problem, so a combination of Stochastic algorithms and a deterministic approach has been proposed recently [35].

Luus and co-workers have attempted to solve multimodal optimal control problems by Dynamic Programming [36]. Luus and Galli [37] showed the multimodality of a CSTR dynamic optimization by using dynamic programming. Luus and Bojkov [38] applied dynamic programming iteratively to solve the bifunctional catalyst problem which is a highly multimodal optimal control problem [39]. Bojkov and Luus [40] found advantages in using random values for control in their Iterative Dynamic Programming approach. Several classes of optimal control problems have been solved by using IDP Singular optimal control problems [41], problems with final state constraints [42], problems with state inequality constraints [42], time optimal control problems [43] and problems from chemical engineering like the optimal control of fed-batch bioreactors. A summary of the IDP approach to dynamic optimization problems is presented in Luus [44]. Recently, Mekarapiruk and Luus [45] applied both randomly and deterministically selected candidates for control in the IDP algorithm. They found some advantages in using this hybrid approach when solving some highly non-linear and multimodal chemical engineering optimal control problems.
problems. Two types of deterministic control candidates (shifting and smoothing candidates) are chosen on the basis of the control policy obtained in the previous iteration. In spite of its success IDP has many algorithm parameters that have to be tuned.

Although several researchers have applied Evolutionary algorithms to optimal control problems in the past, to our best knowledge neither there are previous studies on the performance of Evolutionary algorithms nor on the use of Differential Evolution algorithms to solve multimodal optimal control problems. Michalewicz et al. [46] applied floating-point Genetic algorithms to solve discrete time optimal control problems. Sewald and Kumar [47] applied canonical Genetic algorithms to solve optimal control problems with linear controls. Yamashita and Shima [48] used classical Genetic algorithms to solve free final time optimal control problems with terminal constraints. Smith [49] proposed an evolution program for continuous optimal control problems. Dakev and co-workers [50,51] used algorithms based on the Breeder Genetic algorithm to solved optimal control problems. Pohlheim and Heitjier [52,53] applied Genetic algorithms and also Evolution Strategies to solve dynamic optimization problems of greenhouse climate. Sim et al. [54] used a combination of Genetic algorithms and a shooting method to solve continuous optimal control problems. Roubos et al. [55] applied Genetic algorithms with floating-point chromosomes to solve continuous time optimal control problems of bioreactors. Hashem et al. [56] applied a modified Evolution Strategy on the solution of discrete time optimal control problems. Recently, Wang and Chiou [57] applied a Differential Evolution algorithm to the optimal control and location time problems of differential algebraic equations. Lee et al. [59] used a modified Differential Evolution algorithm to the dynamic optimization of continuous polymer reactor. Chiou et al. [58] proposed two new operators to improve the speed of convergence and to avoid convergence to local minima of Differential Evolution algorithms and they solved some static and dynamic optimization of fed-batch fermentation. An extended review on the applications of evolutionary algorithms for optimal control and additional information on Differential Evolution algorithms is given in reference [60].

The present paper, studies Evolutionary algorithms (EAs) which are used to solve two optimal control problems that are known to have several local minima. Firstly, a First-order gradient algorithm from classical optimal control theory is used to solve both problems. The objective is to illustrate some limitations of this approach in solving multimodal optimal control problems. Next, the performance of four Evolutionary algorithms is compared with that obtained by Iterative Dynamic Programming. It is well known that many Evolutionary algorithms (i.e. Genetic algorithms) tend to be inefficient computationally when they are applied to continuous parameter optimisation problems. Since the computation time is often critical in solving optimal control problems, the design of more efficient evolutionary algorithms is an important challenge. In this work it is investigated how well novel and easy to understand evolutionary algorithms, referred to as Differential Evolution [24,61–64] algorithms, and claimed to be very efficient when they are applied to solve static optimisation problems, perform on solving multimodal optimal control problems. Additionally, almost no knowledge is available on how to choose the algorithm parameters that steer the optimisation process, when Differential Evolution algorithms are applied to solve multimodal dynamic optimisation problems. Hence, in this work, it is investigated how the DE algorithm parameters ‘population size’, ‘crossover constant’ and ‘differential variation coefficient’ act upon its efficiency and effectiveness in solving the selected benchmark problems.

The paper is organised as follows: in Section 2, a general description of the class of optimal control problems we are interested in is given. In Section 3, a general description of an Evolutionary algorithm, and the specific characteristics of both a real-valued Genetic algorithm with sub-populations and the Differential Evolution algorithm are provided. In Section 4, a brief description of a first-order gradient algorithm for the solution of optimal control problems is given and also the main properties of the Iterative Dynamic Programming algorithm are described. Section 5 presents results obtained when the studied evolutionary algorithms were applied to two benchmark optimal control problems belonging to the class of interest. These results are then compared to those obtained with the indirect and gradient method and the direct Iterative Dynamic Programming algorithm. A general discussion is presented at the end of the paper.
2. The class of optimal control problems

Consider the class of optimal control problems where the system is described by the non-linear time-varying dynamic equation:

\[
\dot{x}(t) = f(x(t), u(t), t),
\]

where \( x(t) \in \mathbb{R}^n \) is the state and \( u(t) \in \mathbb{R}^m \) is the control. The control inputs are constrained,

\[
a_i(t) \leq u_i(t) \leq \beta_i(t), \quad i = 1, 2, \ldots, m,
\]

where \( a_i(t) \) and \( \beta_i(t) \) are known time functions. Furthermore,

\[
x(0) = x_0,
\]

is the known initial condition. Then, the optimal control problem is to find the input \( u(t), t \in [t_0, t_f] \) that drives the plant along the trajectory \( x(t), t \in [t_0, t_f] \) such that the cost function

\[
J(u(t)) = \phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t) \, dt
\]

is minimised where the final time \( t_f \) is fixed [65]. There are two general approaches to solve these problems numerically: indirect and direct methods [66]. The first group is based on the solution of a calculus of variations problem through the use of the Pontryagin’s minimum principle (PMP) [67]. In a direct approach, on the other hand, the optimal control problem (1)-(4) is approximated by a finite dimensional optimisation problem, which can be cast in a non-linear programming (NLP) form and solved accordingly [68,69]. This is achieved through control parameterisation. In our case the control \( u(t) \) is assumed to be piecewise constant

\[
u(t_k) = u_{k}, \quad t \in [t_k, t_{k+1}], \quad k = 0, 1, \ldots, N - 1,
\]

\[
t_0 = 0, \quad t_N = t_f.
\]

This is a realistic assumption in the case of digital control. As a result \( N \times m \) parameters determine the control over \([0, t_f]\). The NLP problem is to find the stacked control vector \( \tilde{u} \in \mathbb{R}^{m \times N} \) defined by

\[
\tilde{u} = [u_1^T, u_2^T, \ldots, u_N^T] \in \mathbb{R}^{m \times N},
\]

where \( u_i, i = 1, 2, \ldots, m \times N \) are scalars.

3. Two classes of Evolutionary algorithms: Breeder Genetic algorithms and Differential Evolution

Following Back [20] a generic description of an Evolutionary algorithm is presented in Fig. 1. An Evolutionary algorithm is a stochastic search method, which maintains a population \( P(\theta) := \{\tilde{u}_1(\theta), \ldots, \tilde{u}_\mu(\theta)\} \) of individuals (chromosomes) \( \tilde{u}_i \in I; i = 1, \ldots, \mu \), at generation \( g \), where \( I \) is a space of individuals, and \( \mu \) is the parent population size. Each individual represents a potential solution of the problem and is implemented as some generic data structure (strings of bits in Genetic algorithms, real numbers in Evolution Strategies). By means of the manipulation of a family of solutions, an Evolutionary algorithm implements a survival of the fittest strategy in order to try to find the best solution to the problem. Each individual is evaluated by a fitness function \( \Phi : I \rightarrow \mathbb{R} \), such that a real value is assigned to each potential solution, which is a measure of how individuals perform in the problem domain. Next, an iterative process starts in which a set of evolutionary operators is applied to the population in order to generate new individuals.

From a set \( \{\omega_\Theta_1, \ldots, \omega_\Theta_z\} \cup \{\omega_\Theta_0 : I^\lambda \rightarrow I^\mu\} \) of probabilistic evolutionary \( \omega_\Theta \) operators [70] (for instance: crossover, mutation), each one specified by parameters given in the sets \( \Theta_i \in \mathbb{R} \), some operators are applied to the population and a new evaluation of its fitness is calculated. The main evolutionary operators applied to the population \( P(\theta) \)

\[
g := 0
\]

Generate \( P(0) := \{\tilde{u}_1(0), \ldots, \tilde{u}_\mu(0)\} \)

Evaluate \( \Phi(P(0)) := \{\Phi_1(0), \ldots, \Phi_\mu(0)\} \)

While \( \Phi(\theta) \) is true do

Recombine \( P(\theta) := \rho_\Theta(P(\theta)) \)

Mutate \( P(\theta) := \varrho_\Theta(P(\theta)) \)

Evaluate \( \Phi(\theta) := \{\Phi_1(\theta), \ldots, \Phi_\mu(\theta)\} \)

Select \( P(g+1) := \delta_\Theta(P^g(\theta)) \)

\( g = g + 1 \)

End

Fig. 1. Procedure Evolutionary algorithm.
are recombination (crossover) $r_{Θ} : I^λ \rightarrow I^λ$ and mutation $m_{Θ} : I^λ \rightarrow I^λ$. A selection operator $s_{Θ} : (I^λ \cup I^μ) \rightarrow I^μ$ which may modify the number of individuals from $λ$ or $λ + μ$ to $μ$, is applied as well, where $λ, μ \in N$, $λ$ the number of offspring. As before, the selection operator may be governed by a set of parameters defined in $Θ_s$. The set $Q \subset P(g)$ denotes an additional set of individuals. The function $ι : I^μ \rightarrow \{\text{true, false}\}$ represents the termination criterion for the evolutionary algorithm. After a number of generations, it is expected that the best individual of the population represents a near-optimum global solution.

3.1. Two Evolutionary algorithms based on the Breeder Genetic algorithm

The Breeder Genetic algorithm (BGA) is one of the most efficient Genetic algorithms available in the domain of continuous parameter optimisation. In addition, an extended theory has been proposed that verifies some practical results [71]. BGA utilises a real number representation for a chromosome. That is, $μ$ real vectors of dimension $d$ make up the population $P(g)$. According to the previous notation: $\vec{a} = \vec{x} = (x_1, \ldots , x_d) \in R^d$. Each potential solution in the evolutionary framework consists of the vector $\vec{u} = [u_1, u_2, \ldots u_{N\times d}]$ of parameters obtained from the transformation of the continuous optimal control problem into a NLP problem. The only necessary modification is a rearrangement of parameters, in order to ensure that consecutive realizations of one single element of the control vector appear in adjacent positions in a chromosome. That is, a chromosome is implemented specifically as the vector of floating-point numbers: $\vec{u} = \vec{a}$, so $d = m \times N$.

The genetic operators that have been chosen to make up the Evolutionary algorithm are (i) crossover by discrete recombination (see Fig. 2), (ii) mutation by the operator of the Breeder Genetic algorithm (see Fig. 2), and (iii) selection by Stochastic Universal Sampling. Also the option of sub-populations, to be described later, has been implemented in order to increase the chances to find the global optimum. According to Mühlenbein and Schlierkamp-Voosen [71], the discrete recombination $r_d : I^2 \rightarrow I$ (crossover operator) is defined as follows: let $\vec{a}_1 = (a_{11}, \ldots , a_{1d})$ and $\vec{a}_2 = (a_{21}, \ldots , a_{2d})$ be two parent chromosomes. Then each element of the offspring $\vec{a}_3 = (a_{31}, \ldots , a_{3d})$ is computed by

$$
a_{3i} = \begin{cases} 
a_{1i} & \text{if } \text{rand}(i) < 0.5 \\
a_{2i} & \text{otherwise}
\end{cases}, \quad i = 1, \ldots , d, \quad (6)
$$

![Fig. 2. A graphical representation of both operators mutation of the BGA and discrete recombination in the two-dimensional space.](image-url)
where rand() is a uniform random number from [0,1]. This operator is applied \( \mu \) times by picking up parents randomly in order to create an offspring population.

The mutation operator of the Breeder Genetic algorithm \( m(\mu,\alpha) \) is defined as follows: let \( \tilde{a} = (a_1, \ldots, a_d) \) be a parent solution. Then, for a given probability of mutation \( p_m \in [0,1] \) and a specified ‘shrinking mutation range’ \( r_i \in [0,1] \) a gene (variable) \( a_i' \) is selected and modified to generate a new variable according to:

\[
a_i' = \begin{cases} a_i + m_i \text{ range} \cdot \delta, & \text{if } \text{rand}() < 0.5 \\ a_i - m_i \text{ range} \cdot \delta, & \text{otherwise} \end{cases}, \quad i = 1, \ldots, d,
\]

where

\[
\text{range}_i = \frac{1}{2} (\beta_i - \alpha_i), \\
m_i = \begin{cases} 1, & \text{if } \text{rand}() < p_m, \\ 0, & \text{otherwise} \end{cases}, \quad \delta = \sum_{j=0}^{19} 2^{-j}.
\]

rand() is a uniform random number from [0,1], \( \gamma_j \in [0,1] \) with probability 0.05, \( p_m = 1/d \) normally, and \( \alpha_i, \beta_i \) denote the lower and upper boundaries of the variable \( a_i \). With the given settings for \( \delta \) the mutation operator is able to locate the optimum up to a precision of range \( r_i \times 2^{-19} \).

The selection operator \( s : P^{t+1} \rightarrow P^t \) consists of a combination of an elitist selection mechanism and the stochastic universal sampling algorithm. Firstly, the objective function \( f(\tilde{a}_i) = J(\tilde{a}_i), i = 1, \ldots, \mu \) is calculated, which is equal to the cost function of the optimal control problem. The cost \( J(\tilde{a}) \) is evaluated through integration of the dynamic equation (Eq. (1)) given parameterised control. Then, the fitness function \( \Phi(\tilde{a}) \) is calculated using a linear ranking scheme:

\[
\Phi(\tilde{a}_i) = 2 - s_p + 2(s_p - 1) \frac{f(\tilde{a}_i) - 1}{\mu - 1}, \quad i = 1, \ldots, \mu,
\]

with the selection pressure coefficient \( s_p = 2.0 \) and \( f(\tilde{a}_i) \) the index position in the descending ordered population of the objective function value of individual \( i \). The stochastic universal sampling algorithm picks the parent chromosomes for the new population such that the probability for \( \tilde{a}_i \) being picked equals \( p_r(\tilde{a}_i) \), \( p_s(\tilde{a}_i) \) are calculated according to:

\[
p_r(\tilde{a}_i) = \frac{\Phi(\tilde{a}_i)}{\sum_{j=1}^{\mu} \Phi(\tilde{a}_j)}, \quad s_p = \sum_{j=1}^{\mu} \Phi(\tilde{a}_j),
\]

where \( \Phi(\tilde{a}) \) is the fitness of individual \( \tilde{a} \). To implement an elitist selection scheme new individuals are generated as a fraction of the population size \( \lambda = \mu s_p \) where \( s_p \) is termed the generation gap, a parameter determined by the user. Once offsprings are generated and their fitness functions calculated they are inserted into the new population. An insertion function replaces the worst performing individuals allowing the best previous solutions to belong to the new population in order to maintain the size of the original population \( \mu \).

Fig. 3 shows a flow chart of the Breeder Genetic algorithm in which the main operators are indicated. When the algorithm does not consider several populations migration does not take place.

The sub-population methodology divides the whole population in multiple subpopulations or demes. The evolutionary operators evolve during a number of generations for each sub-population. From time to time some individuals migrate from one sub-population to another. Three parameters have to be specified: the migration rate, the manner of selection of individuals for migration and the topology over which migration takes place. The migration rate is only a scalar number, which specifies the number of individuals to be migrated. The individuals to be migrated can be selected randomly or according to their fitness. There are three main migration topologies: a ring in which only adjacent sub-populations can interchange individuals, a neighbourhood migration, which is an extension of the previous one where migration in each adjacent sub-population is allowed. Finally, unrestricted migration topology, in which individuals may migrate from any sub-population to another. There is some evidence showing that sub-populations help evolutionary algorithms to locate the global optimum [72].

The computer implementation of these Evolutionary algorithms is given in the Genetic algorithm toolbox for use with MATLAB [73]. The integration of the dynamic
3.2. Differential Evolution algorithms

Differential Evolution algorithms are evolutionary algorithms that have already shown appealing features as efficient methods for the optimisation of continuous space functions. Storn [24] have reported impressive results that show DE outperformed other evolutionary methods and the stochastic differential equations approach on solving some benchmark continuous parameter optimisation problems. These algorithms use a floating-point representation for the solutions in the population. However, the evolutionary operators are completely different from those used in methods known as Genetic algorithms, Evolution Strategies and Evolutionary Programming. In DE, the mutation operator $m_1(x) : P^* \rightarrow P^*$ consists of the generation of $\mu$ mutated vectors according to:

$$
\tilde{v}_i = \tilde{a}_i + F(\tilde{a}_j - \tilde{a}_k), \quad i = 1, 2, \ldots, \mu,
$$

where $\tilde{a}_i$, $\tilde{a}_j$, and $\tilde{a}_k$ are random points from the current population, and $F$ is a scaling factor. This mutation process is followed by crossover and mutation operators as shown in Fig. 4.
where the random indices \( r_1, r_2, r_3 \in [1, 2, \ldots, \mu] \) are mutually different and also different from the index \( i \). \( F \in [0, 2] \) is a real constant parameter that affects the differential variation between two vectors (see Fig. 4). Greater values of \( F \) and/or the population size (\( \mu \)) tend to increase the global search capabilities of the algorithm because more areas of the search space are explored.

The crossover operator \( rCR : I^2 \rightarrow I \) combines the previously mutated vector \( \vec{v}_i = [v_1, v_2, \ldots, v_d] \) with a so-called target vector (a parent solution from the old population) \( \vec{a}_i = [a_1, a_2, \ldots, a_d] \) to generate a so-called trial vector \( \vec{a}_i' = [a_1', a_2', \ldots, a_d'] \) according to:

\[
a_{ji}' = \begin{cases} 
  v_{ji}, & \text{if } (\text{randb}(j) \leq \text{CR}) \text{ or } j = \text{rnbr}(i) \\
  a_{ji}, & \text{if } (\text{randb}(j) > \text{CR}) \text{ and } j \neq \text{rnbr}(i)
\end{cases}
\]

where \( \text{randb}(j) \in [0, 1] \) is the \( j \)th evaluation of a uniform random number generator, \( \text{rnbr}(i) \in 1, 2, \ldots, d \) is a randomly chosen index. \( \text{CR} \in [0, 1] \) is the crossover constant, a parameter that increases the diversity of the individuals in the population. The ‘hyper-cube’ boundaries generated by the crossover operator represented by the dotted rectangle in Fig. 4. \( \vec{a}_i' \) can take one of the three corners except for \( \vec{a}_i \). Greater values of \( \text{CR} \) give rise to a child vector \( \vec{a}_i' \) more similar to the mutated vector \( \vec{v}_i \). Therefore, the speed of convergence of the algorithm is increased.

As can be seen from Eq. (11), each member of the population plays once the role of a target vector. It is important to realise that even when \( \text{CR} = 0 \), Eq. (11) ensures that parent and child vectors differ by at least one gene (variable). The three algorithm parameters that steer the search of the algorithm, are the population size (\( \mu \)), the crossover constant (\( \text{CR} \)) and the differential variation factor (\( F \)). They remain constant during an optimisation. The selection operator \( s : P^i \rightarrow P^i \) parees the cost function value of the target vector \( \vec{a}_i \) with that of the associated trial vector \( \vec{a}_i' \), \( i = 1, 2, \ldots, \mu \) and the best vector of these two becomes a member of the population for the next generation. That is,

\[
\text{if } \Phi(\vec{a}_i'(g)) < \Phi(\vec{a}_i(g)), \text{ then }
\vec{a}_i(g+1) := \vec{a}_i'(g), \text{ else }
\vec{a}_i(g+1) := \vec{a}_i(g), \quad i = 1, \ldots, \mu,
\]

where \( g \) denotes the current generation. Fig. 5 presents a flowchart with the main operators of a Differential Evolution algorithm.

---

**Fig. 5. Flowchart of a Differential Evolution algorithm.**
It appears that the differential mutation as designed in DE fulfills three properties that are crucial for an efficient mutation scheme [61]. Firstly, DE guarantees a distribution with zero mean by randomly sampling difference vectors, i.e. \( \vec{x}_i - \vec{x}_j \) has the same probability of being selected as the opposite \( \vec{x}_j - \vec{x}_i \) has. Secondly, the distribution of vector differentials is automatically self-scaling. DE scales mutation step sizes by scaling their relative magnitude. Thirdly, DE is rotational invariant since the mutation distribution generated by difference vectors will always have the same orientation as the level lines of the objective function. In addition, it seems that DE algorithms have a property that Price has called a universal global optimiser.

Different Differential Evolution (DE) algorithms can be distinguished by the properties of the mutation and crossover operators. In this research two DE algorithms that are claimed to be the most efficient are tested. First, the Differential Evolution algorithm denoted by DE/rand/1/bin [24,61–63] that stands for: while only one difference vector is considered by the mutation operation as described by Eq. (10) while the crossover scheme is due to independent binomial experiments. The second DE algorithm studied is shortly described as DE/best/2/bin [24,62,63]. In this case the mutated vectors instead of (10) are described by

\[
\vec{v}_i = \vec{a}_{best} + F(\vec{a}_i + \vec{a}_j - \vec{a}_k - \vec{a}_l),
\]

where \( \vec{a}_{best} \) is the current best solution in the population. Since originally DE algorithms were designed to solve unconstrained static optimisation problems [24], a modification is required in order to deal with constraints for the controls (Eq. (2)). A clipping technique has been introduced to guarantee that only feasible trial vectors are generated after the mutation and crossover operators:

\[
d_{ji}^*(g) = \begin{cases} 
\beta_j, & \text{if } d_{ji}^*(g) > \beta_j \\
\alpha_j, & \text{if } d_{ji}^*(g) > \alpha_j
\end{cases}, \quad j = 1, 2, \ldots, d, \quad i = 1, 2, \ldots, \mu,
\]

where \( \alpha_j \) and \( \beta_j \) represent the lower and upper boundaries of the control variables, respectively. A remarkable advantage of Differential Evolution algorithms is its simplicity, which means that it is relatively easy to understand how they work. Also they are easy to program. Our computer implementation is based on the MATLAB environment. The core of the algorithm is an m-file that calls a Simulink model programmed as a C-MEX s-function, which contains the dynamic equation of the system and the objective function.

4. The first-order gradient algorithm and the Iterative Dynamic Programming algorithm

4.1. The gradient algorithm

The numerical solution of the optimal control problem described in Section 2 can be accomplished by means of a first-order gradient algorithm properly modified with a clipping technique to deal with constraints for the controls. The basis is the algorithm described by Bryson [67]. However, a line search procedure was introduced in order to calculate the value of the step size parameter \( k \), which in Bryson’s algorithm is constant. The gradient algorithm is described next and applies to a Mayer formulation of the optimal control problem [67].

(i) Guess \( u(t) \) at \( N+1 \) points \( t - t_0 = 0, \ldots, N \Delta T \), \( \Delta T = (t_f - t_0)/N \). \( N \) is an even number.
(ii) Integrate the state equations forward. Store \( x(t) \) at \( t - t_0 = \Delta T, \ldots, N \Delta T \).
(iii) Evaluate \( \phi(t, t) \) and \( \lambda^T(t) = (\partial\phi/\partial x)(t) \).
(iv) Compute and store \( \lambda(t) \) and the function \( (\partial H(x, u, \lambda, t)/\partial u) \) at \( t - t_0 = \Delta T, \ldots, N \Delta T \), by integrating backward in time \( \dot{\lambda} = -\lambda^T \lambda \) \( (\partial\phi/\partial x)(t, t)(\partial x(t), u(t))/\partial x \), starting at \( \lambda(t_f) \), where \( (\partial\phi/\partial x)(t, t)(\partial x(t), u(t))/\partial x \).
(v) Apply a line search algorithm to determine the step size parameter \( k \).
(vi) Compute $lu(t)$ and the new $u(t)$ according to:

$$lu(t) = -k_i (\partial H^2/\partial u)(t), u(t) = u(t) + lu(t).$$

(vii) Clip the controls if necessary in accordance with:

$$u(t) = \begin{cases} 
  at, & \text{if } u(t) > a(t) \\
  bt, & \text{if } u(t) > \beta(t) 
\end{cases}$$

(viii) If $|u_{\text{max}}| > e_k$ stop. Otherwise go to step (ii), where $e_k > 0$ is a desired precision and $u_{\text{max}} = \sqrt{\frac{1}{(1/t_k)^2 \text{tr}(H)lu(t)}} \, dt$.

The previous algorithm was implemented in an enhanced MATLAB-Simulink program with a C-MEX tile $s$ function so as to speed up the simulation of the dynamic system.

4.2. Iterative Dynamic Programming algorithm

An iterative version of the Dynamic Programming algorithm has been proposed by Luus [44] as a highly reliable method for locating the global optimum in optimal control problems. A brief description of this approach is given next. For more details one is referred to [44].

**Step 0.** Initialisation: The time interval $[0, \, \tilde{t}_1]$ is divided into $N$ time intervals, each of length $L$. The control is approximated by the piecewise constant control policy $u(t) = u(t_k) \in \{t_k, \, t_{k+1}\}, \, k = 0, \ldots, \, N - 1, \, t_{k+1} - t_k = L, \, t_0 = 0, \, t_N = \tilde{t}_1$. Choose $u_0(0), \ldots, u_0(N-1)$ and $r_0(0), \ldots, r_0(N-1)$, where $r_0(k), \, k = 0, \ldots, \, N -1$ specifies the range $u_0(k) \pm r_0(k)$ of allowable values of control for the next iteration $u_1(k), \, k = 0, \ldots, \, N - 1$. Select the number of allowable values for control $R > 1$ to be tried at each stage $k = 0, \ldots, \, N - 1$. Choose the region contraction factor $0.5 \leq \gamma < 1.0$ and the number of grid points $M$ for the states. Finally, specify the number of iterations $I$. Set iteration number $i = 1$.

**Step 1.** Use the best control policy from the previous iteration (the guess solution at iteration 1) $u^*_1(0), \ldots, u^*_1(N-1)$, and generate $M = 1$ other control policies within the region $u^*_1(k) \pm r_{i-1}(k), \, k = 0, \ldots, \, N - 1$. Integrate the dynamic equation (Eq. (1)) from $t = 0$ to $\tilde{t}_1$ for all $M$ control policies. The $M$ values of $x_{\text{ini}}(k), \, k = 0, \ldots, \, N - 1,$ $m = 1, \ldots, \, M$ at the beginning of each time stage are stored.

**Step 3.**

(a) At stage $N$, for each of the $M$ stored values for $x^{\text{ini}}(N - 1)$, integrate the dynamic equation (Eq. (1)) from $t_i = -L$ to $t_i$ with each of the $R$ allowable values for the control, which are generated by:

$$u_i(N - 1) = u^*_{i-1}(N - 1) + D \cdot r_{i-1}(N - 1),$$

(15)

where $u^*_{i-1}(N - 1)$ is the best control value obtained in the previous iteration and $D$ is a diagonal matrix of different uniform random numbers between $-1$ and $1$. To deal with constraints of the controls whenever an unfeasible solution is generated it is set to the violated limit, according to:

$$u_i(N - 1) = \begin{cases} 
  u(t), & \text{if } u_i(N - 1) < a(t) \\
  \beta(t), & \text{if } u_i(N - 1) > \beta(t) 
\end{cases}$$

(16)

(b) From the $R$ values of the control choose the one $u^*_i(N - 1)$ that gives the best performance index and store these values.

**Step 4.**

(a) Step back to stage $N - 1$ and repeat step 3a were $N$ is replaced by $N - 1$.

(b) The integration is continued from $x(N - 1)$ over the last time interval $\tilde{t}_i - L$ to $\tilde{t}_i$ using the stored value for $u^*_i(N - 1)$ corresponding to the state grid point closest to the value of the calculated state vector at time $\tilde{t}_i - L$. From the $R$ values of the control select $u^*_i(N - 2)$ that gives the best performance over the time interval $[N - 2, \, N]$.

**Step 5.** Continue the procedure until stage $N = 1$ is reached corresponding to the initial time $t = 0$. Here, there is only one state grid point that corresponds to the initial conditions (Eq. (3)). Store the trajectories $u^*_i(k), \, s^*_i(k) = 0, \, 1, \ldots, N - 1$. 

106

Step 6. Reduce the region for allowable values of the control

\[ r_{i+1}(k) = y - r_i(k), \quad k = 0, 1, \ldots, N - 1. \quad (17) \]

Select the best control obtained from step 5 as the midpoint for the allowable values for control. Set \( i = i + 1 \) and go to step 1.

The previous algorithm is continued for the specified number of iterations \( I \) and after that the results are analysed. Sometimes, the allowable values for controls are selected from a uniform distribution (evenly spaced grid) instead of randomly. Also for some problems it is desirable to use a multi-pass method, which selects the value of the region contraction factor \( y \) equal to a fraction of its size at the beginning of the previous pass. This is implemented to prevent a premature collapse of the search region \([44]\). In that case another parameter called region restoration factor \( 0 \leq \eta \leq 1.0 \) is used. The number of passes \( (P) \) must be defined as well.

From the previous description is apparent that the IDP algorithm has numerous algorithm parameters that can be varied. The region contraction factor \( (\gamma) \), number of allowable values for control \( (R) \), number of grid points \( (N) \), initial region size values \( (r_0(k)) \), and restoration factor \( (\eta) \) in case of multiple passes. Some insight has been obtained about their values as one is applying IDP to a particular problem, but in general a parameter tuning approach is required.

Luus has reported \([44]\) that with too small values of the region contraction factor \( (\gamma) \) premature collapse of the region \( r(k), k = 0, 1, \ldots, N - 1 \) is very likely and too large values give rise to a very slow convergence rate or no convergence at all. Also it is known that small values of \( \gamma \) work properly with sufficiently large values of the allowable values for control \( (R) \).

Conversely, when small values of \( R \) are used, high values of \( \gamma \) are required to increase the chances of finding the global optimum. The allowable number for controls should be chosen as small as possible in order to reduce the computational load. Regarding the number of grid points \( (M) \) is known that for some problems \( M = 1 \) works fine, but in other problems \( M > 1 \) may be necessary. Our computer program of the described Iterative Dynamic Programming algorithm for the MATLAB-Simulink environment is an enhanced code, which uses a C-MEX file for \( s \)-function to speed up the simulation of the dynamic equations.

5. Benchmark problems

5.1. The optimal control of a non-linear stirred tank reactor

A multimodal optimal control problem has been used by Luus \([44]\) to evaluate his Iterative Dynamic Programming algorithm. Ali et al. \([30]\) solved this problem by stochastic global optimisation algorithms. Also, this problem is a member of the list of benchmark problems proposed in the Handbook of Test Problems in Local and Global Optimization \([74]\). A first-order irreversible chemical reaction carried out in a continuous stirred tank reactor (CSTR) has been modelled by two non-linear differential equations that are the result of a heat and mass balance of the process.

\[
\begin{align*}
\dot{x}_1 &= -(2 + u)(x_1 + 0.25) \\
&\quad + (x_2 + 0.5) \exp \left( \frac{25x_1}{x_1 + 2} \right), \quad (18) \\
\dot{x}_2 &= 0.5 - x_2 - (x_2 + 0.5) \exp \left( \frac{25x_1}{x_1 + 2} \right), \quad (19)
\end{align*}
\]

where \( x_1 \) represents the deviation from dimensionless steady-state temperature and \( x_2 \) stands for the deviation from the dimensionless steady-state concentration. The control \( u(t) \) represents the manipulation of the flow-rate of the cooling fluid, which is inserted in the reactor through a coil. The optimal control problem is to determine the unconstrained \( u^*(t) \) that minimises the performance index:

\[
J = \int_0^T (x_1^2 + x_2^2 + 0.1u^2) \, dt, \quad (20)
\]

where \( t_0 = 0.78 \). The initial conditions are \( x(0) = [0.09 \quad 0.09]^T \). It can be shown that this problem has two solutions. In solving this problem numerically the integration of the dynamic system was performed with the ode45 routine available in MATLAB, with the relative tolerance error set to \( 1 \times 10^{-8} \). The initial guesses for the controls of the different algorithms were selected from the interval \( 0 \leq u(t) \leq 5.0 \).
5.1.1. First-order gradient method

A solution of this problem by the gradient method was obtained keeping the step size parameter \( k = 0.12 \) constant. So the line search was not used. Since this is an optimal control problem with fixed final time and without bounds for the controls, and because the partial derivatives can be calculated analytically, Bryson’s Matlab code for Continuous Dynamic Optimization [67] without constraints was applicable. The accuracy of the criterion of convergence was specified as \( \varepsilon_g = 0.0001 \). The convergence of the algorithm was straightforward. The convergence of the first order gradient algorithm to the local or global optimum depends on the initial values for the control. Actually, by using a constant pattern as initial guess, \( u_0(t) = c \), \( 0 \leq t \leq t_f \) the gradient algorithm always converged to the local optimum \( J^* = 0.2444 \) if \( u_0(t) \leq 1.8 \); otherwise it converges to the global optimum \( J^* = 0.1330 \). Fig. 6 shows the two optimal control trajectories associated with the two values of the functional \( J \). It can be seen that the local and global optimal control have completely different shapes.

5.1.2. Iterative Dynamic Programming

In order to solve this problem by means of direct methods, the time interval \([0, t_f]\) was discretized in \( N = 13 \) time intervals since it has been reported that a good approximation to the continuous-time optimal control is obtained by doing so [30, 44]. A piecewise constant approximation for the control was used at each of those time intervals. In the IDP algorithm the parameters values suggested by Luus [44] were chosen: the number of state grid points \( M = 1 \), the number of allowable values for control \( R = 15 \). The region reduction factor was \( \gamma = 0.80 \), the number of iterations \( I = 20 \), the number of passes \( P = 3 \) and the region restoration factor \( \eta = 0.5 \) after each pass. The allowable values for control were generated randomly (see Section 4.2). In order to achieve comparable conditions among all the direct methods, firstly, the initial control trajectory was chosen constant, i.e. \( u_0(t_k) = c \), \( k = 0, 1, \ldots, N - 1 \) with \( c \) a value randomly taken from the control interval \( 0 \leq u(t) \leq 5 \). A similar procedure was followed for the selection of the initial region value \( r_0(t_k) \), \( k = 0, 1, \ldots, N - 1 \) which was

Fig. 6. Optimal control trajectories of the CSTR problem calculated by a first-order gradient method.
Table 1 shows the results obtained by the IDP algorithm. From the values presented in the first four columns it is evident that IDP may still converge to the local optimum \( J^* = 0.2444 \). This occurs when both the initial value for the control and the initial region size are too small. Otherwise IDP converges to the global optimum \( J^* = 0.1365 \). However, Luus and Bojkov [38] have reported that when the state grid is equal to \( M = 1 \), it is beneficial to use a greater region size. Therefore, by selecting a sufficiently large initial region size value \( r_0(t_k) \geq 4 \), convergence to the global optimum is always obtained regardless of the initial value for the control \( u_0(t_k) \). This is shown in Table 1; columns five to seven. Repeated optimisation is necessary since the IDP algorithm (see Section 4.2) generates randomly the allowable values for control \( u(t_k) \).

Table 1
Optimal control of a multimodal CSTR by Iterative Dynamic Programming

<table>
<thead>
<tr>
<th>( u_0(t_k) )</th>
<th>( r_0(t_k) )</th>
<th>( J^* )</th>
<th>CPU time (s)</th>
<th>( J^* )</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4990</td>
<td>2.2999</td>
<td>0.135369</td>
<td>600.79</td>
<td>1.0000</td>
<td>0.1353852</td>
</tr>
<tr>
<td>2.3744</td>
<td>3.1557</td>
<td>0.1355769</td>
<td>620.46</td>
<td>4.2823</td>
<td>0.1355622</td>
</tr>
<tr>
<td>0.9718</td>
<td>4.1646</td>
<td>0.1355676</td>
<td>600.16</td>
<td>2.0389</td>
<td>0.1355806</td>
</tr>
<tr>
<td>1.1568</td>
<td>4.1646</td>
<td>0.1355876</td>
<td>600.16</td>
<td>2.0389</td>
<td>0.1355806</td>
</tr>
<tr>
<td>3.8732</td>
<td>4.4331</td>
<td>0.1355806</td>
<td>590.77</td>
<td>0.7900</td>
<td>0.1355806</td>
</tr>
<tr>
<td>4.0504</td>
<td>3.4722</td>
<td>0.1355905</td>
<td>590.77</td>
<td>0.7900</td>
<td>0.1355905</td>
</tr>
<tr>
<td>4.7105</td>
<td>3.7785</td>
<td>0.1355806</td>
<td>589.58</td>
<td>2.3551</td>
<td>0.1355905</td>
</tr>
<tr>
<td>4.6422</td>
<td>2.2919</td>
<td>0.1355666</td>
<td>589.58</td>
<td>2.3551</td>
<td>0.1355666</td>
</tr>
<tr>
<td>4.6494</td>
<td>0.8660</td>
<td>0.1355282</td>
<td>588.51</td>
<td>2.9137</td>
<td>0.1355811</td>
</tr>
</tbody>
</table>

* Measured on a Pentium III 700 MHz PC, function evaluations: 2100.

selected from the interval \( 0 \leq r_0(t_k) \leq 5 \). Since the control has no constraints Eq. (16) was not used.

Table 1 shows the results obtained by the IDP algorithm. From the values presented in the first four columns it is evident that IDP may still converge to the local optimum \( J^* = 0.2444 \). This occurs when both the initial value for the control and the initial region size are too small. Otherwise IDP converges to the global optimum \( J^* = 0.1365 \). However, Luus and Bojkov [38] have reported that when the state grid is equal to \( M = 1 \), it is beneficial to use a greater region size. Therefore, by selecting a sufficiently large initial region size value \( r_0(t_k) \geq 4 \), convergence to the global optimum is always obtained regardless of the initial value for the control \( u_0(t_k) \). This is shown in Table 1; columns five to seven. Repeated optimisation is necessary since the IDP algorithm (see Section 4.2) generates randomly the allowable values for control \( u(t_k) \).

5.1.3. Evolutionary algorithms

In order to solve the CSTR optimal control problem by Evolutionary algorithms, firstly a convergence criterion for all of them was defined. A measure of similarity of the population seems to be a good criterion [19]. This involves a way to measure in absolute or relative sense how similar solutions in the population are. Sometimes, researchers use the value to reach (VTR) as a stopping criterion, which evidently can be applied only when a solution is already known. Since the states (s), control (u), and also \( J \) are dimensionless in this problem it is a good option to select an absolute convergence criterion. It was defined as follows: let \( J_b \) be the best objective function value in the population \( J_b = \min J(u_i), i = 1, \ldots, \mu \), and \( J_w \) the worst function value \( J_w = \max J(u_i), i = 1, \ldots, \mu \), then an absolute convergence criterion can be defined by \( J_w - J_b < \epsilon_c \). In the current application \( \epsilon_c \) was selected to be \( \epsilon_c = 0.00001 \), which guarantees good accuracy of the solution. Since this optimal control problem is unconstrained Eq. (14) was not used.

Four variants of evolutionary algorithms were implemented and evaluated. Two of them are based on the Breeder Genetic algorithm and two are Differential Evolution algorithms. Tables 2–4 present the main results for various values of the population size \( \mu (20, 15, 10) \). The results reported in the tables are averaged from 10 runs. EA\( _\alpha \) means the Breeder Genetic algorithm with only one population, EA\( _\alpha \) denotes the Breeder Genetic algorithm with sub-populations, and EA\( _\alpha \) stands for the DE algorithm DE/rand/1/bin and EA\( _\alpha \) stands for the DE algorithm DE/best/2/bin. Evolutionary algorithms are evaluated on the basis of four criteria: (i) the number of function evaluations, where each evaluation involves the integration of the dynamic equations (Eq. (1)), (ii) the CPU time (measured on a Pentium III personal computer at 700 MHZ), (iii) the performance index value \( J^* \), and (iv) the convergence efficiency (C.E.%) to the global optimum which is measured by the percentage of times that the algorithm found the global solution.
Table 2
Optimal control of a multimodal CSTR by Evolutionary algorithms (population size $\mu = 20$)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>EA1 BGA</th>
<th>EA2 BGA-subpopulations</th>
<th>EA3 DE/rand/1/bin</th>
<th>EA4 DE/best/2/bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function evaluations</td>
<td>7401.80</td>
<td>7143.6</td>
<td>3494</td>
<td>2270</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>430.02</td>
<td>451.10</td>
<td>242.51</td>
<td>160.44</td>
</tr>
<tr>
<td>$J^*$</td>
<td>0.1358249</td>
<td>0.1355985</td>
<td>0.1355966</td>
<td>0.1355850</td>
</tr>
<tr>
<td>Variance</td>
<td>$9.818 \times 10^{-7}$</td>
<td>$2.3192 \times 10^{-7}$</td>
<td>$2.3503 \times 10^{-6}$</td>
<td>$4.7643 \times 10^{-12}$</td>
</tr>
<tr>
<td>Iterations</td>
<td>410.1</td>
<td>397.3</td>
<td>174.70</td>
<td>113.50</td>
</tr>
<tr>
<td>Parameters</td>
<td>$p_m = 0.09$</td>
<td>$p_m = 0.09, m_r = 0.8$</td>
<td>CR = 0.5</td>
<td>CR = 0.5</td>
</tr>
</tbody>
</table>

* Measured on a Pentium III 700 MHz PC.

Table 3
Optimal control of a multimodal CSTR by evolutionary algorithms (population size $\mu = 15$)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>EA1 BGA</th>
<th>EA2 BGA-subpopulations</th>
<th>EA3 DE/rand/1/bin</th>
<th>EA4 DE/best/2/bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function evaluations</td>
<td>6985.60</td>
<td>5272.6</td>
<td>3535.5</td>
<td>1783.5</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>406.85</td>
<td>314.93</td>
<td>235.70</td>
<td>134.26</td>
</tr>
<tr>
<td>$J^*$</td>
<td>0.1362200</td>
<td>0.1362479</td>
<td>0.1355920</td>
<td>0.1355970</td>
</tr>
<tr>
<td>Variance</td>
<td>$8.8488 \times 10^{-7}$</td>
<td>$3.1354 \times 10^{-7}$</td>
<td>$4.2769 \times 10^{-10}$</td>
<td>$5.5529 \times 10^{-10}$</td>
</tr>
<tr>
<td>Iterations</td>
<td>497.9</td>
<td>438.80</td>
<td>235.70</td>
<td>118.90</td>
</tr>
<tr>
<td>Parameters</td>
<td>$p_m = 0.09$</td>
<td>$p_m = 0.09, m_r = 0.8$</td>
<td>CR = 0.5</td>
<td>CR = 0.5</td>
</tr>
</tbody>
</table>

* Measured on a Pentium III 700 MHz PC.

A parameter tuning approach was applied in order to determine what combination of algorithm parameters gives the best performance index with the least number of function evaluations. Storn [24] have suggested values for the population size from the interval $5^d \leq \mu \leq 10^d$ for static optimisation problems, where $d$ is the dimension of the problem. Price [61] proposed selecting the population size from the interval $2^d \leq \mu \leq 20^d$. Figs. 7–9 show the corresponding optimal control trajectories obtained by one realization of the four Evolutionary algorithms, according to the population size that was applied. From Figs. 7 and 8, it is apparent that all EAs converge to the global solution. From Fig. 9, observe the small deviations from

Table 4
Optimal control of a multimodal CSTR by evolutionary algorithms (population size $\mu = 10$)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>EA1 BGA</th>
<th>EA2 BGA-subpopulations</th>
<th>EA3 DE/rand/1/bin</th>
<th>EA4 DE/best/2/bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function evaluations</td>
<td>2225.8</td>
<td>7925</td>
<td>2907</td>
<td>1719</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>135.57</td>
<td>449.50</td>
<td>200.89</td>
<td>134.80</td>
</tr>
<tr>
<td>$J^*$</td>
<td>0.1440189</td>
<td>0.1365219</td>
<td>0.1356056</td>
<td>0.1356052</td>
</tr>
<tr>
<td>Variance</td>
<td>$9.8458 \times 10^{-5}$</td>
<td>$1.4124 \times 10^{-4}$</td>
<td>$4.6864 \times 10^{-4}$</td>
<td>$1.3844 \times 10^{-4}$</td>
</tr>
<tr>
<td>Iterations</td>
<td>240.20</td>
<td>792</td>
<td>200.70</td>
<td>171.9</td>
</tr>
<tr>
<td>Parameters</td>
<td>$p_m = 0.09$</td>
<td>$p_m = 0.09, m_r = 0.8$</td>
<td>CR = 0.5</td>
<td>CR = 0.5</td>
</tr>
</tbody>
</table>

* Measured on a Pentium III 700 MHz PC.
the global optimum of the optimal control trajectories obtained by EA1 and EA2 but still they did not converged to the local minima.

Since it is apparent that for optimal control problems greater population sizes may increase the computation time dramatically, in this work the use of population sizes around the dimension of the optimisation problem $d = mn$ was chosen. After a population size was fixed other parameters of the algorithms were tuned in order to obtain the best performance index with the least number of function evaluations. The values reported in the tables are obtained this way. In Tables 2–4, the variance of the 10 runs is reported as well.

5.2. The bifunctional catalyst blend optimal control problem

A very challenging multimodal optimal control problem has been studied by Luus [38,44]. Luus [44] showed that this problem has many local optima (25). Esposito and Floudas [28] found recently 300 local minima. This problem is also proposed as a benchmark in the Handbook of Test Problems in Local and Global Optimization [74]. A chemical process converting methylocyclopentane to benzene in a tubular reactor is modelled by a set of seven differential equations:

$$\dot{x}_1 = -k_1 x_1,$$
$$\dot{x}_2 = k_1 x_1 - (k_2 + k_3) x_2 + k_4 x_5,$$
$$\dot{x}_3 = k_2 x_2,$$
$$\dot{x}_4 = -k_4 x_4 + k_5 x_5,$$
$$\dot{x}_5 = k_3 x_2 + k_6 x_4 - (k_4 + k_1 + k_8 + k_9) x_5 + k_7 x_6 + k_{10} x_7,$$
$$\dot{x}_6 = k_8 x_5 - k_3 x_6,$$
$$\dot{x}_7 = k_9 x_5 - k_3 x_6.$$
\[ \dot{x}_7 = k_{9} x_5 - k_{10} x_7, \]

where \( x_i, i = 1, \ldots, 7 \) are the mole fractions of the chemical species, and the rate constants \( (k_i) \) are cubic functions of the catalyst blend \( u(t) \):

\[ k_i = c_{i1} + c_{i2} u + c_{i3} u^2 + c_{i4} u^3, \quad i = 1, \ldots, 10. \]

The values of the coefficients \( c_{ij} \) are given in [44].

The upper and lower bounds on the mass fraction of the hydrogenation catalyst are: \( 0.6 \leq u(t) \leq 0.9 \), and the initial vector of mole fraction is \( x[0] = [0.0000000] \). This is a continuous process operated in steady state, so that 'time' in Eqs. (21)-(28) is equivalent to travel time and thus length along the reactor. The optimal control problem is to find the catalyst blend along the length of the reactor, which in the control problem formulation is considered at times \( 0 \leq t \leq t_f \) where the final effective residence time \( t_f = 2000 \text{g.h/mol} \) such that the concentration in the reactor is maximised:

\[ J = x_7(t_f) \times 10^5. \]

5.2.1. First-order gradient algorithm

In order to solve this problem by means of a first-order gradient algorithm a clipping technique was added to the basic gradient algorithm so as to deal with control constraints. A line search method as described before was added to adjust the step size parameter \( k \) efficiently. The convergence tolerance was set to \( \varepsilon = 0.000001 \). Despite both enhancements the classical method failed to locate the global optimum as can be seen in Table 5 that shows the results of twenty optimisations. The best solutions (emphasized in Table 5) are clearly far from the global solution which equals \( J^* = 10.0942 \) for a piece-wise constant approximation for the controls. However, when the gradient method was started using a solution generated with a direct method (for example IDP or any evolutionary algorithm) it...
Fig. 9. Optimal control trajectories calculated by a realization of the four EAs for the CSTR optimal control problem with a population size of 10 individuals.

Table 5
Optimal control of bifunctional catalyst blend by a first-order gradient algorithm (N = 10)

<table>
<thead>
<tr>
<th>Constant u0(t)</th>
<th>J</th>
<th>Iterations</th>
<th>CPU time (s)</th>
<th>J*</th>
<th>Iterations</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.80</td>
<td>9.6419</td>
<td>16</td>
<td>166.66</td>
<td>8.7627</td>
<td>22</td>
<td>2052.40</td>
</tr>
<tr>
<td>0.70</td>
<td>8.1215</td>
<td>20</td>
<td>3899.00</td>
<td>8.0054</td>
<td>37</td>
<td>6811.80</td>
</tr>
<tr>
<td>0.85</td>
<td>9.6419</td>
<td>8</td>
<td>97.86</td>
<td>8.4409</td>
<td>24</td>
<td>5430.10</td>
</tr>
<tr>
<td>0.85</td>
<td>8.1214</td>
<td>23</td>
<td>1593.30</td>
<td>8.1691</td>
<td>46</td>
<td>6076.10</td>
</tr>
<tr>
<td>0.90</td>
<td>9.6419</td>
<td>6</td>
<td>36.03</td>
<td>8.5083</td>
<td>30</td>
<td>6235.30</td>
</tr>
<tr>
<td>0.70</td>
<td>9.7577</td>
<td>38</td>
<td>494.68</td>
<td>8.4300</td>
<td>21</td>
<td>1931.10</td>
</tr>
<tr>
<td>0.60</td>
<td>8.1214</td>
<td>27</td>
<td>6928.60</td>
<td>9.3883</td>
<td>17</td>
<td>1344.30</td>
</tr>
<tr>
<td>0.72</td>
<td>8.1223</td>
<td>40</td>
<td>9072.10</td>
<td>8.2718</td>
<td>22</td>
<td>3860.80</td>
</tr>
<tr>
<td>0.78</td>
<td>9.6374</td>
<td>31</td>
<td>3355.40</td>
<td>9.1816</td>
<td>38</td>
<td>3396.10</td>
</tr>
<tr>
<td>0.82</td>
<td>9.6419</td>
<td>9</td>
<td>131.05</td>
<td>9.0628</td>
<td>89</td>
<td>13156.00</td>
</tr>
</tbody>
</table>

converged quickly to the value $J^* = 10.1042$. Clearly, due to the presence of many local minima in this problem, a first-order gradient algorithm is easily trapped by one of them. The gradient algorithm is able to converge to the global optimum only if the initial control trajectory is in the vicinity of the true solution. Therefore, the use of a global optimization method such as Evolutionary algorithms to approximate the global solution followed by a local optimisation method such as a first-order gradient algorithm to reach the global optimum is recommended.
optimum exactly seems a good approach in solving multi-modal optimal control problems.

5.2.2. Iterative Dynamic Programming

In order to solve this problem by means of direct methods, the time interval was divided in \( N = 10 \) time subintervals and the control was approximated by a piecewise constant signal at each time interval. In solving it by Iterative Dynamic Programming, the initial control trajectory \( u_0(t_k); k = 0, \ldots, N-1 \) was chosen constant with a value randomly chosen from the control interval \( 0 \leq u(t) \leq 0.9 \).

The algorithm parameters of the IDP algorithm were: number of state grid points \( M = 1 \), number of allowable values for the control \( R = 15 \). The allowable values for control were generated randomly. The parameter region contraction factor was \( \gamma = 0.80 \), which was selected according to Luus’ suggestions [44]. The maximum number of iterations was \( I = 30 \), but the optimisation was stopped when it reached the condition \( J^* - J \leq 0.00005 \). Table 6 shows the main results, which show indeed the convergence to the global optimum all the time and the associated number of function evaluations and iterations. It is clear from the table that if the initial region size is large enough, IDP always finds the global optimum.

However, the sensitivity of IDP to the choice of the initial region value \( r_0(t_k) \) can be illustrated by choosing different values for this parameter. Table 7a shows that with a value of \( r_0(t_k) = 0.27 \) the IDP algorithm

Table 6

<table>
<thead>
<tr>
<th>( u_0(t_k) )</th>
<th>( r_0(t_k) )</th>
<th>FE</th>
<th>( J^* )</th>
<th>Iterations</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6658</td>
<td>0.7814</td>
<td>1500</td>
<td>10.0942</td>
<td>20</td>
<td>174.36</td>
</tr>
<tr>
<td>0.8727</td>
<td>0.6677</td>
<td>1550</td>
<td>10.0942</td>
<td>18</td>
<td>165.58</td>
</tr>
<tr>
<td>0.6533</td>
<td>0.6370</td>
<td>1125</td>
<td>10.0942</td>
<td>15</td>
<td>126.59</td>
</tr>
<tr>
<td>0.8574</td>
<td>0.8600</td>
<td>600</td>
<td>10.0942</td>
<td>8</td>
<td>72.41</td>
</tr>
<tr>
<td>0.7537</td>
<td>0.7033</td>
<td>1125</td>
<td>10.0942</td>
<td>15</td>
<td>121.19</td>
</tr>
<tr>
<td>0.6842</td>
<td>0.7090</td>
<td>1500</td>
<td>10.0942</td>
<td>20</td>
<td>137.52</td>
</tr>
<tr>
<td>0.6357</td>
<td>0.6099</td>
<td>825</td>
<td>10.0942</td>
<td>11</td>
<td>92.58</td>
</tr>
<tr>
<td>0.8515</td>
<td>0.7711</td>
<td>1425</td>
<td>10.0942</td>
<td>19</td>
<td>152.20</td>
</tr>
<tr>
<td>0.8035</td>
<td>0.8417</td>
<td>1575</td>
<td>10.0942</td>
<td>21</td>
<td>169.65</td>
</tr>
<tr>
<td>0.7915</td>
<td>0.6154</td>
<td>1050</td>
<td>10.0942</td>
<td>14</td>
<td>138.23</td>
</tr>
</tbody>
</table>

\( \text{Mean.} \)

Table 7

<table>
<thead>
<tr>
<th>( r_0(t_k) )</th>
<th>Function evaluations</th>
<th>Iterations</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.27</td>
<td>1084.70</td>
<td>13.70</td>
<td>150.64</td>
</tr>
<tr>
<td>0.30</td>
<td>1207.5</td>
<td>16.1</td>
<td>135.65</td>
</tr>
</tbody>
</table>

\( \text{Measured on a Pentium III 700MHz PC.} \)
Table 8
Optimal control of the bifunctional catalyst blend problem by EAs (population size \( \mu = 25 \))

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>EA1 BGA</th>
<th>EA2 BGA-subpopulations</th>
<th>EA3 DE/rand/1/bin</th>
<th>EA4 DE/best/2/bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function evaluations</td>
<td>7007.10</td>
<td>4890</td>
<td>3172.5</td>
<td>3607.5</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>1186.60</td>
<td>515.14</td>
<td>549.75</td>
<td>632.84</td>
</tr>
<tr>
<td>( J^* )</td>
<td>10.0942</td>
<td>10.0929</td>
<td>10.0941</td>
<td>10.0941</td>
</tr>
<tr>
<td>Variance</td>
<td>0.0012</td>
<td>0.0019</td>
<td>4.8889 \times 10^{-9}</td>
<td>9.8889 \times 10^{-9}</td>
</tr>
<tr>
<td>CE (%)</td>
<td>70</td>
<td>80</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Iterations</td>
<td>278.28</td>
<td>202.50</td>
<td>126.90</td>
<td>144.30</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \sigma_{\text{gap}} ) = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \sigma_{\text{gap}} ) = 1, ( m_r ) = 0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( F ) = 0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_m ) = 0.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_m ) = 0.45, subpop = 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR = 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( a \) Measured on a Pentium III 700 MHZ PC.

converged to the global optimum only in 20% of the cases. By using \( r_0(t_k) = 0.30 \) this percentage is increased to 60%. With greater values than \( r_0(t_k) \geq 0.40, k = 0, 1, \ldots, N - 1 \) for the initial region size the IDP is capable to always converge to the global optimum as it is shown in Table 7b. Table 7b shows the average of 10 optimisations with different random allowable values for control.

5.2.3. Evolutionary algorithms

In order to solve this problem by the selected Evolutionary algorithms, first a proper and common convergence criterion based on a measure of the quality of the solutions in the population was chosen. In contrast to example one, where an absolute criterion was selected, here the following relative convergence criterion was applied:

\[
\frac{\mu}{\varepsilon_d} (J_w - J_b) \leq \sum_{i=1}^{\mu} |J(\tilde{u}_i)|
\]

(29)

where \( J_w \) and \( J_b \) are defined as before, \( J(\tilde{u}_i) \) is the performance index, and \( \varepsilon_d = 0.001 \) is a constant value selected according to the desired precision. The initialisation for all the EAs was done randomly from the control input domain \( 0.6 \leq u_0(t_k) \leq 0.9 \). As before, a parameter tuning approach was applied and the best results obtained with the selected parameter values are reported. Tables 8–10 show the averages of 10 runs for three values of the population size (\( \mu = 15, 20, 25 \)). Reported values of the performance (\( J^* \)) are averages over successful optimisations. Figs. 10–12 show the optimal control trajectory obtained by one realization of the evolutionary algorithms.

Looking at Figs. 10 and 11 we can see only tiny differences among the optimal control trajectories for the four evaluated evolutionary algorithms. This is the trajectory is the associated with the global optimum value of the performance index \( J = 10.0942 \). In case of Fig. 12, a local minima is plotted which was obtained by EA1. The Differential Evolution algorithms always approached the global optimum solution.

Table 9
Optimal control of the bifunctional catalyst blend problem (population size \( \mu = 20 \))

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>EA1 BGA</th>
<th>EA2 BGA-subpopulations</th>
<th>EA3 DE/rand/1/bin</th>
<th>EA4 DE/best/2/bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function evaluations</td>
<td>11493</td>
<td>18227.5</td>
<td>2406</td>
<td>2776</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>1896.90</td>
<td>3237.20</td>
<td>453.92</td>
<td>537.62</td>
</tr>
<tr>
<td>( J^* )</td>
<td>10.0944</td>
<td>10.0916</td>
<td>10.0941</td>
<td>10.0941</td>
</tr>
<tr>
<td>Variance</td>
<td>0.0064</td>
<td>3.9754 \times 10^{-4}</td>
<td>6.2222 \times 10^{-9}</td>
<td>8.4444 \times 10^{-9}</td>
</tr>
<tr>
<td>CE (%)</td>
<td>60</td>
<td>80</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Iterations</td>
<td>572.66</td>
<td>910.12</td>
<td>124.8</td>
<td>136.80</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \sigma_{\text{gap}} ) = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \sigma_{\text{gap}} ) = 1, ( m_r ) = 0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F = 0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_m ) = 0.28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_m ) = 0.45, subpop = 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR = 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( a \) Measured on a Pentium III 700 MHZ PC.
Table 10
Optimal control of the bifunctional catalyst blend problem (population size $\mu = 15$)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>EA1, BGA</th>
<th>EA2, BGA-subpopulations</th>
<th>EA3, DE/rand/1/bin</th>
<th>EA4, DE/best/2/bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function evaluations</td>
<td>6552.90</td>
<td>12718</td>
<td>1752</td>
<td>2268</td>
</tr>
<tr>
<td>CPU time (s)$^a$</td>
<td>578.99</td>
<td>2433.40</td>
<td>341.28</td>
<td>385.44</td>
</tr>
<tr>
<td>$J^*$</td>
<td>10.0937</td>
<td>10.0854</td>
<td>10.0940</td>
<td>10.0939</td>
</tr>
<tr>
<td>Iterations</td>
<td>434.85</td>
<td>793.60</td>
<td>116.80</td>
<td>151.20</td>
</tr>
<tr>
<td>Parameters</td>
<td>$g_{gap} = 1$</td>
<td>$g_{gap} = 1, m = 0.2$</td>
<td>$F = 0.9$</td>
<td>$F = 1.0$</td>
</tr>
<tr>
<td></td>
<td>$p_{m} = 0.29$</td>
<td>$p_{m} = 0.2$, subpop = 4</td>
<td>$CR = 0.0$</td>
<td>$CR = 0.0$</td>
</tr>
</tbody>
</table>

*a Measured on a Pentium III 700 MHz PC.

Fig. 10. Optimal control trajectories calculated by a realization of the EAs for the highly multimodal OCP using a population size of 25 individuals.

6. Discussion

6.1. Non-linear stirred tank reactor

The first-order gradient method is capable of solving low-multimodal optimal control problems like the multimodal CSTR analysed only by repeated initialization of the guess control trajectory. A gradient-based algorithm requires the partial derivatives (gradients) of the problems and also a initial control trajectory in the neighbourhood of a local minima or global optimum.
The Iterative Dynamic Programming algorithm due to its stochastic nature (random values for the control) can potentially solve a multimodal optimal control problem since it is not based on the calculation of the gradients. However, this problem has shown that this approach is rather sensitive to other parameter of IDP such as the initial region size value. Similar behaviour of IDP was reported by Luus and Galli [37] in the past.

Concerning the Evolutionary algorithms several remarks need to be made. Firstly, the four evaluated algorithms converged to the global optimum. Even in the case that a population size of ten individuals was chosen, an acceptable value for the performance index in the neighbourhood of the global optimum was calculated, in contrast to the expectation that with a smaller value of the population size the algorithms might converge to the local optimum. Secondly, the Differential Evolution algorithms turned out to be more efficient than those based on the Breeder Genetic algorithm taking into account the accuracy of the solutions. Thirdly, within the Differential Evolution algorithms the one that solved the problem with the lowest number of function evaluations was DE/best/2/bin.

In both algorithms EA1 and EA2 the mutation rate (\( p_m \)) was selected a bit greater than the default value frequently chosen \( p_m = 1/(m \cdot N) \), in order to improve the probability of the algorithm to converge to the global optimum. Clearly, this gives rise to a higher number of function evaluations. In case of EA2 it was not possible to obtain a better solution by increasing the number of subpopulations by more than two. For EA2 the migration rate (\( m_r \)) between populations was allowed each 20 generations.

As far as parameter tuning of the Differential Evolution algorithms is concerned, the heuristic rules applied to determine the values of the algorithm parameters ‘amplification variation’ (\( F \)) and ‘crossover constant’ (\( CR \)) were as follows. The initial values were \( F = 0.5 \) and \( CR = 0.1 \) according to Storn [24], and
then depending on whether a good convergence was observed the value of the crossover constant was increased in order to improve the efficiency of the algorithm. Also smaller values for $F$ were tried. It can be seen from the tables that values for $F$ within the range $0.4 \leq F \leq 0.6$ were sufficient to obtain a convergence efficiency of 100%. Greater values of CR resulted in solutions with worst (less accurate) performance but not necessarily to a local optimum. Also it was noticed that the change of CR from 0.1 to 0.5 resulted in considerable faster convergence. In contrast to the population size values suggested by Storn and Price relatively small populations also allowed to find a good solution. It seems that because the problem is not highly multimodal a relative small value of the differential variation parameter $F$ suffices to explore properly the whole search space.

Compared to results obtained with the IDP algorithm the DE/best/2/bin algorithm was able to solve this problem with a smaller number of function evaluations (cf. Table 1). This shows that DE algorithms are actually very efficient evolutionary algorithms. To determine the values of the three algorithm parameters that steer the optimisation only a few experiments are required. In the IDP algorithm there are more algorithm parameters to be tuned than in DE. In contrast to the IDP algorithm, the algorithm parameters that guarantee a convergence efficiency of 100% are easily obtained for the DE algorithms considered here.

6.2. The bifunctional catalyst blend problem

This problem showed that a first-order gradient algorithm can find the global optimum if and only if the initial control trajectory is at the vicinity of the global optimum. The difficulty is that one can not know in advance were that vicinity is. Therefore,
highly multimodal optimal control problems cannot be solved satisfactorily with this approach.

The Iterative Dynamic Programming algorithm performed very efficiently in solving the multimodal optimal control problem. However, from our experiments, one can see clearly a high sensitivity to the initial value of the region size. Therefore, in order to solve a new dynamic optimisation problem it is required to attempt several values of this parameter of IDP. These previous experimentation should be included in the evaluation of the efficiency of the algorithm. Shiaw and Hwang [75] have attempted to improve the convergence of this approach by using robust sequence random number generators. Iterative Dynamic Programming is indeed a powerful and efficient method to solve difficult highly multimodal optimal control problems.

Regarding evolutionary algorithms EA1 and EA2 did not reach 100% of convergence efficiency when small population sizes were used. However, it was found that by increasing adequately the population size, EA3 improves remarkably. As a matter of fact, by using a population size of $\mu = 60$ with four subpopulations and 15 individuals each one, a generation gap $g_{gap} = 0.9$, a migration rate $m_t = 0.2$, mutation rate $p_m = 0.1$ and a elapsed time of 10 generations between migrations, EA2 converged always to a value $J^* = 10.042$. The average required number of function evaluations was 7048.6 with a variance of $1 \times 10^{-9}$. EA1 converged only 80% of the times to the global optimum with population size $\mu = 40$ and 50. These results illustrate the benefits of using sub-populations. The Differential Evolution algorithms always reached a convergence efficiency of 100%. Moreover, both DE algorithms were considerably more efficient than the Breeder Genetic algorithms.

By comparing Tables 8–10 it can be seen that both EA1 and EA2 require a greater value of the mutation rate parameter $p_m$ to obtain reasonable solutions when the population size is diminished. This sometimes leads to an increased number of function evaluations. In contrast, the Differential Evolution requires less function evaluations, and usually has better convergence to the global optimum. As for the number of function evaluations the best algorithm in this case is EA1. But, EA4 is not significantly less efficient than EA3, while it performs better when the population size is further reduced (Table 10).

As before the DE algorithms were tuned by applying some heuristic rules to determine the differential variation parameter ($F$) and crossover constant (CR) that lead to the global optimum efficiently. For the chosen population sizes, starting with initial values $F = 0.5$, and $CR = 0.1$ premature convergence (convergence to a local solution) was observed. Therefore, the value of the parameter $F$ was increased. It was discovered that increasing CR neither improves the speed of convergence nor locating of the global optimum. On the contrary, it was observed that neglecting completely the effect of the crossover operator, by setting $CR = 0$, the effectiveness increases considerably. This value ($CR = 0$) gave the best convergence of the algorithms at the expense of more function evaluations.

It seems that the large multimodality of this problem demands an extensive exploration of the search space. It is recommended to select a value close to one for the differential variation parameter ($F$), and a crossover rate (CR) of zero in highly multimodal problems.

It must be said that in this problem DE algorithms required more function evaluations than the IDP algorithm. However, one could argue that this difference is not significant when taking into account that the IDP algorithm requires more previous experiments to tune its critical algorithm parameters than in the straightforward procedure associated to the DE algorithms.

7. Conclusions

Evolutionary algorithms are robust search methods capable of locating the global optimum of multimodal optimal control problems. These algorithms are not sensitive to the initial control trajectory. They can be initialised randomly. Evolutionary algorithms based on the Breeder Genetic algorithm are able to solve complex multimodal optimal control problems but they demand a large population size or a high mutation rate (probability of mutation). Both properties give rise to an increased number of function evaluations (simulations) and hence a long computation time. The use of sub-populations can improve their convergence to the global optimum as problem 2 of this research has shown.

This research has shown that within the family of Evolutionary algorithms, Differential Evolution algorithms stand out in terms of efficiency as compared
to the Breeder Genetic algorithm. In contrast to the majority of Evolutionary algorithms, where many algorithm parameters have to be tuned, in DE only three algorithm parameter values (the population size, the crossover constant and the differential variation coefficient) have to be selected. The population size plays a crucial role in solving optimal control problems. Selecting a too small population size reduces the probability of finding the global solution. Increasing the population size increases the chances that the algorithm finds the global optimum but the computation time increases. The two investigated Differential Evolution algorithms solved the two benchmark multimodal optimal control problems properly and efficiently. In solving the first problem the efficiency achieved by DE was clearly comparable to that of the (non-Evolutionary) IDP algorithm. As for the second problem the efficiency of DE was slightly inferior to the one required by the IDP algorithm when the algorithm parameters have been tuned. On the other hand, the determination of appropriate values of the algorithm parameters for IDP is more difficult and more involved [cf. 30]. In summary, Differential Evolution algorithms are reliable and relatively efficient to solve multimodal optimal control problems. Clearly, improving the efficiency of the DE algorithms further remains an important issue for future research.

The guidelines to select the algorithm parameter values crossover constant (CR) and amplification of the differential variation (F) in the DE algorithms obtained from this investigation can be summarized as follows. Adopt a smaller population size than in static optimisation; a population size less than or equal to two times the dimension of the optimisation problem (µ ≤ 2(N × m)) is desirable for optimal control problems. Highly multimodal optimal control problems may require greater values of the amplification variation coefficient (F) and a very small or zero value for the crossover constant (CR). Low multimodal optimal control problems may need medium values of the mutation parameter (F) and greater or medium values for the crossover constant (CR). Further research is needed if one is interested in finding more generic rules for parameter tuning.

In order to solve multimodal optimal control problems more efficiently and accurately, an efficient Evolutionary algorithm like Differential Evolution may be used to approximate the global minimum. Next, a classical local optimisation algorithm can be applied to accurately compute the global optimum. The development of such a combined method is the aim of our future work.

Acknowledgements

The first author wishes to mention the support of the National Council of Science and Technology (CONACyT) of Mexico to this research.

References

algorithms, in: A. Munack, H.J. Tantau (Eds.), Preprints Mathematical and Control Applications in Agriculture and Horticulture, IFAC Workshop, Hannover, Germany, 1997, pp. 113–118.


