Efficient Evolutionary Algorithms for Optimal Control

Irineo Lorenzo López Cruz

NN08201, 3222

Irineo Lorenzo López Cruz

Efficient Evolutionary Algorithms for Optimal Control

PROEFSCHRIFT
ter verkrijging van de graad van doctor
op gezag van de rector magnificus
van Wageningen Universiteit,
prof. dr. ir. L. Speelman,
in het openbaar te verdedigen
op vrijdag 14 juni 2002
des namiddags te half twee in de Aula

1651200

CIP-Data Koninklijke Biblioteek, DEN HAAG
López-Cruz, I.L.
Efficient Evolutionary Algorithms for Optimal Control/I. L. López-Cruz
[S.I.:s.n.]
Thesis Wageningen University.-With ref.- With summary in Dutch and Spanish

ISBN 90-5808-649-6

[illegible]

Propositions attached to the thesis:
"Efficient Evolutionary Algorithms for optimal control"
by Irineo L. Lopez Cruz

1. Optimal control problems with multiple local minima are challenging problems, which makes them particularly suitable for testing the efficiency of global optimization algorithms.

2. Differential Evolution algorithms are the most efficient evolutionary algorithms designed so far.

3. "The goal of an efficient mutation scheme (in evolutionary algorithms) is to generate increments or steps that move existing object variables in the right direction by the right amount at the right time". K. V. Price, *An introduction to Differential Evolution*, 1999.

4. Mathematical models are not only essential in control but in general they are fundamental to enlarging knowledge and helping with practical applications.

5. The no-free-lunch (NFL) theorem implies that it is more important to investigate which class of EAs is suitable to solve which class of optimization problems instead of trying to design an algorithm able to solve all the classes of optimization problems.

6. "Evolution provides the solution to the problem of how to solve problems". David B. Fogel, *Evolutionary Computation. Toward a new Philosophy of Machine Intelligence*, 1995.

7. That Mayan mathematicians invented independently the number Zero was a remarkable achievement. Even more admirable is the evidence that suggests they were familiar with the concept of Matrix as well.

8. Dehumanization of the humankind does not mean that human race is evil by nature but only that mankind is not as advanced, civilized and developed as many people believe.

Abstract

Lopez-Cruz I.L. (2002). Efficient Evolutionary Algorithms for Optimal Control. PhD Thesis, Wageningen University, Wageningen, The Netherlands.

The purpose of this study was to investigate and search for efficient evolutionary algorithms to solve optimal control problems that are expected to have local solutions. These optimal control problems are called multi-modal. Evolutionary algorithms are stochastic search methods that use a population of potential solutions and three evolutionary operators: mutation, recombination and selection. The goal was achieved by studying and analysing the performance of Differential Evolution (DE) algorithms a class of evolutionary algorithms that not only do not share theoretical and practical limitations that Genetic Algorithms have as global optimisers, but also they overcome those drawbacks.

However, at the beginning of this research a genetic algorithm with real-valued individuals and specialized genetic operators (GENOCOP) was studied by solving some hard optimal control problems. Although results showed that the evolutionary approach is feasible to solve high-dimensional, multivariable, multimodal and non-differentiable control problems, some limitations regarding computational efficiency were found.

Differential Evolution algorithms were chosen and used to solve two multi-modal (benchmark) optimal control problems. Also some Breeder Genetic Algorithms (BGA) and the Iterative Dynamic Programming (IDP) algorithm were applied for comparison purposes. The comparison confirmed that DE algorithms stand out in terms of efficiency as compared to the Breeder Genetic algorithms. Moreover, in contrast to the majority of Evolutionary Algorithms, which have many algorithm parameters that need to be selected or tuned, DE has only three algorithm parameters that have to be selected or tuned. These are the population size (μ), the crossover constant (CR) and the differential variation amplification (F). All the investigated DE algorithms solved the multi-modal optimal control problems properly and efficiently. The computational efficiency achieved by the DE algorithms in solving the first low multi-modal problem, was comparable to that of IDP. When applied to the second highly multi-modal problem, the computational efficiency of DE was slightly inferior to the one required by IDP, after tuning of the algorithm parameters. However, the selection or tuning of the algorithm parameters for IDP is more difficult and more involved.

Some guidelines for the selection of the DE algorithm parameters were obtained. Take the population size less than or equal to two times the number of variables to be optimised that result from the control parameterisation of the original optimal control problem ($\mu \le 2n_u$). Highly multi-modal optimal control problems require a large value of the differential variation amplification ($F \ge 0.9$) and a very small or zero value for the crossover constant ($0 \le CR \le 0.2$). Low multi-modal optimal control problems need a medium value for the differential variation amplification ($0.4 \le F \le 0.7$) and a large or medium value for the crossover constant ($0.2 \le CR \le 0.5$). To improve further the performance of DE algorithms a parameter control strategy was proposed and evaluated on the algorithm *DE/rand/1/bin*. Results show that computational efficiency can be significantly improved.

Finally, some possibilities of using DE algorithms to solve some practical optimal control problems were investigated. The algorithm *DE/best/2/bin* was applied to solve the optimal control of nitrate in lettuce and results were compared with local optimisation algorithms of optimal control. A combination of a DE algorithm and a first order gradient algorithm was proposed in order to exploit the advantages of both approaches. The DE algorithm is used to approximate the global solution sufficiently close after which the gradient algorithm can converge to it efficiently. The feasibility of this approach, which is especially interesting for multi-modal optimal control problems, was demonstrated.

To my parents Genaro and Imelda

To my beloved wife Nora

To my children Sacnite, Noel and Aaron

Acknowledgements

Many people have contributed either directly or indirectly to the end of this thesis.

First of all, I would like to express my deeply and sincere gratitude to my promoter Prof. Dr. ir. Gerrit van Straten who gave me the opportunity to pursue my doctoral studies at the Systems and Control Group and for his permanent support, continuous advice and guidance during the whole work. It has been a great and wonderful experience working with him.

My deep appreciations and sincere gratitude to Dr. Gerard van Willigenburg my co-promoter and daily supervisor for his sharp criticism, wonderful guidance, and always right suggestions to improve my work. It was a pleasure but also a real challenge to me working with him.

I would like to express my gratitude to Dr. Hans Stigter who always openly and kindly answered many questions I used to ask him, for his fruitful discussions and always interesting talking.

I would like to thank all the participants of the European research project NICOLET, in particular to Dr. Ido Seginer from Israel because I have learnt a lot working from time to time for this project.

My deep appreciations to my former supervisor Dr. John Goddard from UAM-I (Mexico) who introduced me to the Evolutionary Algorithms field and encouraged me to come to Holland.

I am grateful also of my former supervisor Dr. Jose Negrete from UNAM (who now is a lecturer and researcher of the MSc. Program in Artificial Intelligence of Universidad Veracruzana, Mexico) for taught me his Philosophy of Science.

I would like to mention the support I received from all the staff members of the Systems and Control Group. They provided a lot of warm support during my hard adaptation period to the "Dutch life". I am obliged especially with Ilse Quirijns. Thanks for all your support. But also I am indebted to Frank Tap (former PhD student), Leo Lukasse (former PhD student) and Camile Hol (now in Delft), Rachel van Ooteghem and S. de Graaf for his help with the ACW algorithm. Thanks to the technical staff the former computer system manager Henk Veen and the secretaries Marja, Corrie and Miranda.

I would like to thank my former office-mate and friend A. Abusam (former PhD student) for the time we spent discussing and for his friendship.

I would like to mention my appreciations to my best Mexican friend Armando Ramirez who supported me (by e-mail) with his friendship and comradeship.

Last but not least, I would like to thank the support of my family. I am deeply indebted to my wife Nora, my daughter Sacnite and my sons Noel and Aaron. Without their support, encouragement, companionship, love, comprehension and kindness I could not bring to an end this project. The time we have spent in Holland has been a beneficial and wonderful experience for my family for two main reasons. Nora was operated of her eyes successfully and now she can see almost normally. My children have experienced the contact with another culture, which I hope will be very important and positive for their future. I also like to thank my parents, brothers and sisters in Mexico for their support and encouragement. Special mention deserves the support and encouragement I always received from mother in law.

Institutional Acknowledgements

I would like to express my gratitude to the Universidad Autónoma Chapingo (Mexico) who provided me part of the financial support for the development of this research.

I also would like to express my gratitude to the National Council of Science and Technology of Mexico who supported me with a fellowship (grant 116076) for the development of this work.

Table of contents

1. General introduction

1.1 Motivation

Convergence to local solutions is likely, if optimal control problems are solved by means of gradient-based local search methods. Recently there has been an increasing interest in the use of global optimisation algorithms to solve optimal control problems which are expected to have local solutions. These optimal control problems are called multi-modal. Evolutionary Algorithms (EAs) are global optimisation algorithms that have mainly been applied to solve static optimisation problems. Only rarely Evolutionary Algorithms have been used to solve optimal control problems. This may be due to the belief that their computational efficiency is insufficient to solve this type of problem. In addition the application of Evolutionary Algorithms is a relatively young area of research. Together with my personal interest in the application of EA's this motivates the research in this thesis which concerns a search for the feasibility and efficiency of evolutionary algorithms to solve multi-modal optimal control problems.

The efficiency is a critical issue when applying EA's. Even more so when optimal control problems are solved, since in this case, each function evaluation involves a system simulation, which is computationally expensive. Therefore in this research we tried to focus on EA's that are known or proved to be efficient. The application of these algorithms to multi-modal optimal control problems, in most cases, presents a new area of research.

Numerical methods for the solution of optimal control problems can be roughly divided into two groups: indirect and direct methods [1]. The first group is based on finding a solution that satisfies the Pontryagin's maximum principle or the related necessary conditions through solving a two-point boundary-value problem [2]. Direct methods are based on an approximation of the infinite dimensional optimal control problem by a non-linear programming (NLP) problem. This can be done by either control and state parameterisation or control vector parameterisation only [3]. The non-linear programming problem that results after the parameterisation is often multi-modal. Gradient-based optimisation algorithms are known to converge to local optima. To surmount this problem, global optimisation algorithms can be used. Since it is well known that Dynamic Programming is hardly ever feasible due to the curse of the dimensionality [2], Iterative Dynamic Programming (IDP) has been proposed [4]. Other global optimisation methods applied recently to solve multimodal optimal control problems are Stochastic Algorithms [5, 6]. Our work is motivated by the potential that Evolutionary Algorithms (EAs) have, as global optimisers, to solve multimodal optimal control problems. Since the computation time is critical in solving optimal control problems and EAs are known not to be very efficient the issue of efficiency is addressed. Some of the state of the art evolutionary algorithms will be the focus of our investigations.

1.2 Background

1.2.1 Brief description of mainstreams of evolutionary algorithms

In this section a generic description of the most prominent evolutionary algorithms is provided. Basically our portrayal follows the work of Bäck [7] who has proposed, in our view, a rather generic framework to describe global stochastic search algorithms inspired by evolution. The next meta-algorithm gives a generic description for a wide class of evolutionary algorithms:

Outline of an Evolutionary Algorithm

```
$g := 0$; generate $P(0) := \{\vec{a}_1(0),\ldots,\vec{a}_\mu(0)\} \in I^\mu$;
evaluate $P(0): \{\Phi(\vec{a}_1(0)),\ldots,\Phi(\vec{a}_\mu(0))\}$;
while ($\iota$(P(g))$\neq$true) do
        Recombine $P'(g) := r\Theta_r(P(g))$;
        Mutate $P''(g) := m\Theta_m(P'(g))$;
        Evaluate $P''(g): \{\Phi(\vec{a}''_1(g)),\ldots,\Phi(\vec{a}''_\lambda(g))\}$;
        Select $P(g+1) := s\Theta_s(P''(g) \cup Q)$;
        $g := g+1$;
end
```

An Evolutionary Algorithm (EA) is a stochastic search method, which maintains a population $P(g) := \{\vec{a}_1(g),\ldots,\vec{a}_\mu(g)\}$ of individuals $\vec{a}_i \in I$, $i = 1,\ldots,\mu$ at generation g, where I denotes a space of individuals, and $\mu \in N$ is the parent population size. Each individual represents a potential solution to the problem at hand and it is implemented as some generic data structure (i.e. 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 $\Phi : I \to \Re$, such that a real value is assigned to each potential solution, which represents a measure of how well 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 [8]. From a set $\{w\Theta_1,\ldots,w\Theta_z \mid w\Theta_i : I^\lambda \to I^\lambda\} \cup \{w\Theta_0 : I^\mu \to I^\lambda\}$ of probabilistic evolutionary $w\Theta_i$ operators (for instance: crossover, mutation), each one specified by parameters given in the sets $\Theta_i \subset \Re$, some operators are applied to the population and a new evaluation of its fitness is calculated. The evolutionary operators: recombination (crossover) $r\Theta_r : I^\mu \to I^\lambda$, mutation $m\Theta_m : I^\lambda \to I^\lambda$ and selection $s\Theta_s : \left(I^\lambda \cup I^{\mu+\lambda}\right) \to I^\mu$ are used to transform the population $P(g)$. $\lambda \in \mathrm{N}$ represents the number of offspring or new solutions in the population. The set $Q \subset P(g)$ denotes an additional set of individuals, which can be the empty set, or a subset of the parent population $P(g)$. The function $\iota : I^\mu \to \{true, false\}$ specifies the termination criterion. After a number of generations, it is expected that the best individual of the population represent a near-optimum solution.

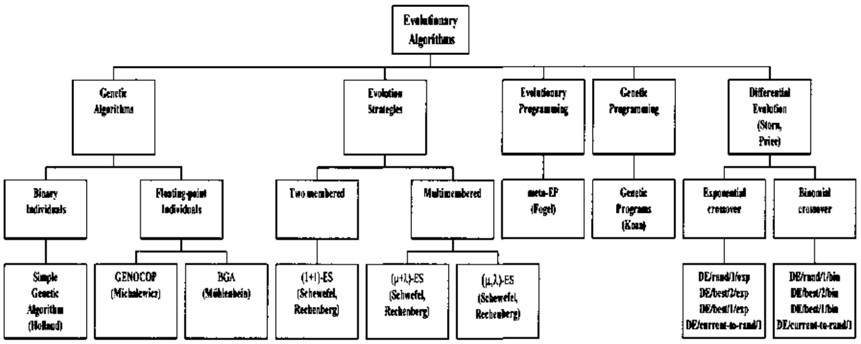


Figure 1. Family tree showing the most relevant Evolutionary Algorithms

Traditionally, three main examples of this generic algorithm have been identified: Genetic Algorithms [9, 10, 11], Evolution Strategies [1, 12] and Evolutionary Programming [1, 13]. However, other algorithms inspired by evolution share similarities with the three original EAs, for instance, Differential Evolution [14, 15], Genetic Programming [16] and possibly others. The next subsections summarize main properties of Genetic Algorithms, Evolutionary Programming, Evolution Strategies and Differential Evolution since they are more important ones from an optimisation viewpoint. Figure 1 presents a family tree with a classification of the most important Evolutionary Algorithms.

1.2.1.1 **Genetic Algorithms (binary and floating-point representation)**

Binary representation

In canonical Genetic Algorithms (GAs) [9, 10] an individual is represented by strings of binary numbers $\vec{a} \in I$, where I is the binary space $\{0,1\}$. An individual or chromosome is just a binary string $\vec{a} = (a_1, ..., a_l)$, and l is the length or the used number of bits. As this approach is applied to solve continuous parameter optimisation problems with n variables $x_i, i = 1,2,...,n$ to be optimised $\vec{x} = [x_1, x_2, ..., x_n] \in D \subseteq R^n$, where $D : x_i \in [u_i, v_i]$, $i = 1,2,...,n$ here u_i and v_i denote lower and upper limits of the variable interval x_i. Using a binary code [7,11] each element x_i of $\vec{x}$ can be coded by elements of $\vec{a}$. This is represented by $\vec{x} = \psi(\vec{a})$. Regarding the calculation of the fitness function $\Phi(\vec{a}) = \Phi'(\psi(\vec{a})) : R^n \rightarrow R$ where Φ' is a function that guarantees positive values, since the standard selection mechanism of GAs requires positive fitness values.

Mutation

Consistent with the binary representation of the solutions a mutation operator $m_{\{p_m\}} : I \rightarrow I$ modifies an individual $\vec{a}' = m_{\{p_m\}}(\vec{a})$ according to

$a_i' = \begin{matrix} a_i & if\ \chi_i > p_m \\ 1 - a_i & if\ \chi_i \le p_m \end{matrix}$, $\forall_i \in [1,...,l]$ where $\chi_i \in [0,1]$ is a uniform random variable,

and p_m is a probability of mutation and a_i means the i^{th} bit of the string.

Crossover

The simplest recombination operator is the so-called one-point crossover $r_{\{p_c\}}: I^2 \to I^2$, which combines two strings $\vec{s}$ and $\vec{v}$ to generate two new individuals $\vec{s}' = (s_1,...,s_{k-1}, s_k, v_{k+1},...,v_l)$ and $\vec{v}' = (v_1,...,v_{k-1}, v_k, s_{k+1},...,s_l)$, where $k \in \{1,...,l-1\}$ and p_c specifies the probability of selecting a pair of strings to be mated. It seems that more commonly applied crossover operators are multi-point crossover and uniform crossover. The multi-point crossover operator $r_{\{p_c,m\}}: I^2 \to I$ generates a new individual according to: $a_i' = \begin{matrix} a_{s,i}, & \forall_i(\chi_k < i \le \chi_{k+1}), k \le m \\ a_{v,i}, & otherwise \end{matrix}$, where $(\chi_1,...,\chi_m) \in [1,...,l-1]$ denote random crossover positions and m is the number of crossover points. The uniform crossover operator $r: I^2 \to I$ generates a new individual according to $a_i' = \begin{matrix} a_{s,i}, \lambda_i > 1/2 \\ a_{v,i}, \lambda_i \le 1/2 \end{matrix}, \forall_i \in [1,...,l]$, where $\lambda_i \in [0,1]$ is a uniform random variable. Other binary oriented recombination operators are found in the literature [7, 9, 10].

Selection

The selection operator $s: I^\mu \to I^\mu$ implements a probabilistic survival strategy. First selection probabilities are computed $p_s(\vec{a}_j(g)) = \Phi(\vec{a}_j(g)) / \sum_{k=1}^{\mu} \Phi(\vec{a}_k(g))$, $j = 1,2,...,\mu$, which reflect the relative fitness in the population of each individual. Using these probabilities a population is chosen according to a sampling mechanism. Generally, the Stochastic Universal Sampling scheme [7], which determines the number of copies (samples) of each individual from the current to the next population, is applied. An example is the Simple Genetic algorithm [9, 10] (see Figure 1).

Floating-point representation

In genetic algorithms with a floating-point representation, [11, 17] an individual is given by a vector of real numbers such that $\vec{a} = x \in \Re^n$. The fitness function is just $\Phi(\vec{a}): R^n \to R$. However, genetic operators are different in order to deal with this representation.

Mutation

As far as mutation is concerned more known operators are: uniform, boundary and non-uniform mutation as well as mutation of the Breeder Genetic algorithm. Uniform mutation $m: I \to I$ alters an individual $\vec{a}$ into $\vec{a}'$ according to $a_i' = \begin{cases} r, \ if & i = j \\ a_i, & otherwise \end{cases}$ where $r \in [u_i, v_i]$ is a uniform random value within the interval for the i^{th} variable. The boundary mutation $m_{\{r\}}: I \to I$ modifies slightly the previous operator in which the mutated variable is generated by $a_i' = \begin{cases} u_i, \textit{ if } i = j, r < 0.5 \\ v_i, \textit{ if } i = j, r \geq 0.5 \\ a_i, \quad \textit{otherwise} \end{cases}$, where $r \in [0,1]$ is a uniform random number. The non-uniform mutation $m_{\{r,b,g,G\}} : I \rightarrow I$ on the other hand, generates a new individual $\vec{a}' = (a_1,...,a_i,...,a_n)$ where $a_i' = \begin{matrix} a_i + \delta(g, v_i - a_i) \textit{ if } rnb == 0 \\ a_i - \delta(g, a_i - u_i) \textit{ if } rnb == 1 \end{matrix}$, $\delta(g,y) = y \cdot r \cdot (1 - g/G)^b$ and $r \in [0,1]$ is a uniform random number, *rnb* is a random binary digit, G is the maximal generation number and b is a parameter.

The breeder genetic algorithm mutation operator $m_{\{\chi\}} : I \rightarrow I$ creates a new individual $\vec{a}'$ according to $a_i' = \begin{cases} a_i + s \cdot (v_i - u_i) \cdot \delta \textit{ if } \chi_i < 0.5 \\ a_i - s \cdot (v_i - u_i) \cdot \delta \textit{ if } \chi_i \geq 0.5 \end{cases}$, $\delta = \sum_{j=0}^{15} 2^j \alpha_j$, $\alpha_i \in [0,1]$, $s = 0.1$, where $\chi_i \in [0,1]$ is a uniform random value.

Crossover

Some floating-point crossover operators are: simple, arithmetic, and heuristic. The simple crossover $r_{\{b\}} : I^2 \rightarrow I^2$ combines two individuals $\vec{a}^1$ and $\vec{a}^2$ to generate two new feasible individuals $\vec{a}'^1 = (a_1^1,...,a_i^1, a_{i+1}^2 \cdot b + a_{i+1}^1(1-b),...,a_n^2 \cdot b + a_n^1 \cdot (1-b))$ and $\vec{a}'^2 = (a_1^2,...,a_i^2, a_{i+1}^2 \cdot b + a_{i+1}^1 \cdot (1-b),...,a_n^2 \cdot b + a_n^1 \cdot (1-b))$, where $b \in [1,0]$ is a uniform random value and $i \in [1,n]$ is a randomly chosen index. The arithmetic crossover $r_{\{b\}} : I^2 \rightarrow I^2$ combines two parents $\vec{a}_1$ and $\vec{a}_2$ in order to generate two new feasible solutions $\vec{a}_1' = b \cdot \vec{a}_1 + (1-b)\vec{a}_2$ and $\vec{a}_2' = (1-b)\vec{a}_1 + b\vec{a}_2$, where $b \in [0,1]$ is a random number. Heuristic crossover $r_{\{d\}} : I^2 \rightarrow I$ combines two parent solutions $\vec{a}_1$ and $\vec{a}_2$ so as to generate a new individual $\vec{a}_3 = \vec{a}_1 + d \cdot (\vec{a}_1 - \vec{a}_2)$ where $d \in [0,1]$ is a random number, and $\vec{a}_l$ and $\vec{a}_2$ are selected such that $\Phi(\vec{a}_1) \leq \Phi(\vec{a}_2)$.

The discrete recombination operator $r_{\{d\}} : I^2 \rightarrow I$ combines two vectors $\vec{a}^1$ and $\vec{a}^2$ to obtain only one new individual $\vec{a}^3$ where $a_i^3 = \begin{cases} a_i^1, \textit{ if } \ d_i < 0.5 \\ a_i^2, \quad \textit{otherwise} \end{cases}$, $i = 1,...,n$ and $d_i \in [0,1]$ is a uniform random number. Similarly the extended intermediate recombination operator $r_{\{\alpha\}} : I^2 \rightarrow I$ generates a new individual by $a_i^3 = a_i^1 + \alpha_i(a_i^2 - a_i^1)$, $i = 1,...,n$ and α_i is a uniform random variable from the interval $[-0.25, 1.25]$. In case that only one coefficient α is applied to the whole difference vector, the operator is called extended line recombination.

Selection

In addition to the same scheme of selection as in GAs with a binary representation, in case of real-valued vectors several other selection operators have been reported in the literature [11, 17]. Two important examples of genetic algorithms with floating-point representation of the individuals are the GENOCOP (GEnetic algorithm for Numerical Optimization for Constrained Problems) system [11] and the Breeder Genetic algorithm (BGA) [17] as can be seen in Figure 1.

1.2.1.2 **Evolution Strategies and Evolutionary Programming**

A set of Evolution Strategies (ES) can be identified [7, 12]. Since more advanced Evolutionary Programming (EP) algorithms share properties of ES here only the differences between both approaches are mentioned. All ES use a complex representation of the chromosomes in the population $\vec{a} = (\vec{x}, \vec{\sigma}, \vec{\alpha})$, where $\vec{x}$ denotes the vector of to be optimised variables, $\vec{\sigma}$ the strategy vector of standard deviations associated to $\vec{x}$ and $\vec{\alpha}$ rotation angles also associated to $\vec{x}$. The fitness function becomes $\Phi(\vec{a}) : R^n \rightarrow R$. In ES and EP not only the object vector but also the strategy vector and rotation angles vector are subjected to the evolutionary process. In case of Evolutionary Programming [7, 13] generally an individual $\vec{a} = (\vec{x}, \vec{v})$ is represented as a vector of object variables and one vector of standard deviations.

Mutation

The mutation operator $m_{\{\tau,\tau',\beta\}} : I \rightarrow I$ yields a new individual $\vec{a}' = (\vec{x}', \vec{\sigma}', \vec{\alpha}')$ according to: $\sigma_i' = \sigma_i \cdot \exp(\tau_2 \cdot N(0,1) + \tau_1 \cdot N_i(0,1))$, $\alpha_j' = \alpha_j + \beta \cdot N_j(0,1)$, $\vec{x}' = \vec{x} + \vec{N}(0, \vec{\sigma}', \vec{\alpha}')$, $\forall_i \in \{1,...,n\}, \forall_j \in \{1,...,n \cdot (n-1)/2\}$, where $N(0,1)$ stands for a random variable having expectation zero and standard deviation one, $\vec{N}(0, \vec{\sigma}', \vec{\alpha}')$ denotes a multivariate normal distribution with specified covariance matrix, and τ_1, τ_2 are algorithm parameters depending on n, and β is a constant [1, 7].

In case of Evolutionary Programming the mutation operator $m_{\{\zeta\}} : I \rightarrow I$ produces a new individual $\vec{a}' = (\vec{x}', \vec{v}')$ as follows: $x_i' = x_i + \sqrt{v_i} \cdot N_i(0,1)$, $v_i' = v_i + \sqrt{\zeta v_i} \cdot N_i(0,1)$, $\forall_i \in \{1,...,n\}$, where ζ denotes an algorithm parameter.

Crossover

Modern ES may use several recombination operators [7, 12], and they may be different for object variables, standard deviations and rotating angles. Yet, in general, two recombination operators are commonly applied. First, the discrete recombination operator defined above. And also the intermediate recombination operator (applied here only on $\vec{x}$) $r : I^2 \rightarrow I$ that combines two different randomly selected parents $\vec{x}_V$, $\vec{x}_S$ from the population, to generate a new individual $\vec{x}'$, in which, $x_i' = x_{S,i} + (x_{V,i} - x_{S,i})/2$. Recently, Schwefel [12] has proposed some generalizations for recombination operators in which each element of the new vector is selected probabilistically from all the individuals in the population. Then, the intermediate recombination operator $r : I^\mu \rightarrow I$ is given by $x_i' = x_{S,i} + (x_{V_i,i} - x_{S,i})/2$ where $x_{V_i,i}$ denotes that a new parent x_V is selected for each element of the vector. There are no recombination operators in Evolutionary Programming.

Selection

The selection mechanism in Evolution Strategies is deterministic. There are two general operators. An operator $s_{\{\mu+\lambda\}} : I^{\mu+\lambda} \rightarrow I^{\mu}$ selects the best μ individuals out of the union of parents and offspring while the operator $s_{\{\mu,\lambda\}} : I^{\lambda} \rightarrow I^{\mu}$ selects the best μ individuals out of the offspring only. The selection mechanism is used to denote multimembered evolution strategies $(\mu+\lambda)$-ES and (μ,λ)-ES respectively. In case $\mu = 1$, and $\lambda = 1$ the two membered evolution strategy is obtained which was the first designed ES and is denoted traditionally by $(1,1)$-ES (see diagram 1).

In Evolutionary Programming the selection operator $s_{\{q\}} : I^{2\mu} \rightarrow I^{\mu}$, on the other hand, uses a tournament selection mechanism in order to generate a new population. For each individual $\vec{a}_j, j \in \{1,...,2\mu\}$, q individuals are chosen randomly from the union of the parents μ and the offspring $\lambda = \mu$. A score $w_j \in \{0,...,q\}$ is obtained from counting how many of those individuals perform worse than $\vec{a}_j$. All the individuals $\vec{a}_j, j \in \{1,...,2\mu\}$ are ranked in descending order of their score and the best μ of them are selected to form the next population. An example of an Evolutionary Program is the meta-EP algorithm proposed by Fogel (see Figure 1).

1.2.1.3 **Differential Evolution algorithms**

All Differential Evolution (DE) algorithms use vectors of floating-point numbers to represent the individuals in the population [14, 15]. Using the previous notation we have: $\vec{a} = x \in \Re^n$. The fitness function is $\Phi(\vec{a}) : R^n \rightarrow R$.

Mutation

There are several mutation operators in DE algorithms. A mutation mechanism $m_{\{F\}} : I \rightarrow I$ yields a mutated individual $\vec{a}' = m_{\{F\}}(\vec{a})$ by modifying the vector $\vec{a}$ according to: $\vec{a}_i' = \vec{a}_{r_1} + F \times (\vec{a}_{r_2} - \vec{a}_{r_3})$, $\forall_i \in \{1,...,\mu\}$, where $r_1 \neq r_2 \neq r_3 \neq i$ denote mutually different indices. The vector $\vec{a}_i$ is named the target vector, which clearly is a parent individual. The vector $\vec{a}_{r_1}$ is the to be mutated individual which is selected randomly from the population. Vectors $\vec{a}_{r_2}$ and $\vec{a}_{r_3}$ form a difference vector. F is an algorithm parameter that affects the differential variation. A second mutation operator takes the to be mutated vector equal to the target vector as follows: $\vec{a}_i' = \vec{a}_i + F \times (\vec{a}_{r_1} - \vec{a}_{r_2})$, $\forall_i \in \{1,...,\mu\}$ where $r_1 \neq r_2 \neq i$ are mutually different indices.

A third mutation operator is given by $\vec{a}_i' = \vec{a}_{best} + F \times (\vec{a}_{r_1} - \vec{a}_{r_2})$, $\forall_i \in \{1,...,\mu\}$ where $r_1 \neq r_2 \neq i$ denote mutually different indices. In this case the to be mutated vector is the best individual in the population ($\vec{a}_{best}(g)$) at the current generation (g), namely $\Phi(\vec{a}_{best}) \leq \Phi(\vec{a}_j)$, $\forall_j, j = 1,...,\mu$. Another mutation mechanism combines two difference vectors $\vec{a}_i' = \vec{a}_{best} + F \times (\vec{a}_{r_1} + \vec{a}_{r_2} - \vec{a}_{r_3} - \vec{a}_{r_4})$, $\forall_i \in \{1,...,\mu\}$ where $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i$ are mutually different indices. A more complicated mutation operator uses the to be mutated vector to built the difference vectors: $\vec{a}_i' = \vec{a}_i + F \times (\vec{a}_{r_1} - \vec{a}_{r_2}) + K \times (\vec{a}_{r_3} - \vec{a}_i)$, $\forall_i \in \{1,...,\mu\}$ where $r_1 \neq r_2 \neq r_3 \neq i$ are mutually different indices and K is another algorithm parameter.

Crossover

The recombination operator $r_{\{CR\}} : I^2 \to I$ acts on two parents, the mutated vector ($\vec{a}_i'$) and the target vector ($\vec{a}_i$), which can be considered as a parent individual, so as to form a trial vector ($\vec{a}_i''$) whose elements are given by: $a_{ji}'' = \begin{matrix} a_{ji}' \text{ if } randb(j) \leq CR \text{ or } j = rnbr(i) \\ a_{ji} \text{ if } randb(j) > CR \text{ and } j \neq rnbr(i) \end{matrix}$, $i = 1,2,...,\mu$, $j = 1,2,...,n$, where CR is the crossover constant, $randb(j) \in [0,1]$ denotes the jth evaluation of an uniform random number generator, and $rnbr(i) \in [1,..,n]$ is a randomly selected index. This operator is called binomial crossover.

A second recombination operator in DE is the so-called exponential crossover. In this case each element of the trial vector is generated according to: $a_{ji}'' = \begin{matrix} a_{ji} \text{ while } randb(j) > CR \text{ and } j \neq rnbr(i) \\ a_{ji}' \quad \text{afterward} \end{matrix}$, $\forall_i \in \{1,...,\mu\}$, for $j = 1,...,n\}$

Selection

The selection operator $s_{\{\mu+\lambda\}} : I^{\mu+\lambda} \to I^{\mu}$ picks the μ best individuals from the union of parents and offspring ($\mu + \lambda$) to form the next population, where $\mu = \lambda$. This is done by a simple comparison of the fitness of the target ($\vec{a}_i$) and trial ($\vec{a}_i''$) vectors $i = 1,...,\mu$, in such a way that only if the condition $\Phi(\vec{a}_i'') \leq \Phi(\vec{a}_i)$ is satisfied then $\vec{a}_i''$ becomes a member of the new population otherwise $\vec{a}_i$ (the parent individual) is selected. An excellent discussion on Differential Evolution algorithms is provided by Price [15]. Each Differential Evolution algorithm can be identified by the notation *DE/x/y/z* [14], where x denotes the choice of the vector to be mutated, y is the number of difference vectors used for mutation and z specifies the type of crossover scheme. Some instances of DE algorithms are listed in Figure 1.

1.2.2 **On the theory of Evolutionary Algorithms**

Although several theories have been proposed to account mainly for the behaviour of Genetic Algorithms and Evolution Strategies, it seems that there still is no definite theory that explains thoroughly why and how evolutionary algorithms work [18, 19]. However, based on the concept of Random Heuristic Search [20] a set of definitions, theorems and formal proofs has been developed that mathematically formalize evolutionary algorithms. In contrast to other theories, Random Heuristic Search is a paradigm that would explain the behaviour of the most important evolutionary algorithms: Genetic Algorithms, Evolutionary Programming, Evolution Strategies and Genetic Programming. Roughly, Random Heuristic Search is considered to be a discrete dynamical system consisting of two parts: a collection of elements (population) chosen from a search space, which can be any finite set, and also a heuristic search or transition rule which from any population P_i will produce another population P_{i+1}. Since the transition rule is stochastic, a heuristic function is defined, which given the current population, produces a vector whose *j*-th component is the probability that the *j*-th element of the search space is chosen as a member of the next population P_{i+1}. A characterization of Random Heuristic Search can be given in terms of Markov Chains. An important challenge is not only to show that a particular evolutionary algorithm is an instance of Random Heuristic Search but also to find its corresponding heuristic function. So far a detailed analysis of the behaviour of the Simple Genetic Algorithm has been presented recently [21]. In addition, several theoretical results have recently been discussed in the literature [8, 22, 23] based on the application of Markov Chains theory to Evolution Strategies.

1.2.3 **Direct optimisation methods in optimal control and Evolutionary Algorithms**

Numerical methods for optimal control can be classified into two generic groups: indirect and direct methods [1, 25]. The first group is based on finding a solution that satisfies the Pontryagin's Maximum Principle or the related necessary optimality conditions, which constitute a two-point boundary-value problem. Generally, gradient and shooting methods are applied [2, 24]. Direct methods attempt a direct minimization of the objective functional of the optimal control problem by control parameterisation or control and state parameterisation. Through parameterisation the dynamic optimisation problem is transformed into a Non-Linear Programming problem. Then both local and global optimisation algorithms to solve this type of problems may be applied. In this work only control parameterisation will be considered.

In this thesis we will consider general optimal control problems where the system may be non-linear and the cost functional need not be quadratic. Consider the system

$$\dot{x} = f(x,u,p,t) \tag{1.1}$$

where $x \in R^n$ is the state vector, $u \in R^m$ is the control vector, $p \in R^l$ the fixed parameter vector and t represents time. The optimal control problem is to find the control trajectory $u(t)$, $t_0 \le t \le t_f$ which minimizes the cost functional

$$J = \phi(x(t_f)) + \int_{0}^{t_f} L(x,u,t)dt \tag{1.2}$$

subject to the system dynamics (1.1), with known initial conditions $x(t_0) = x_0$. In equation (1.2) $\phi \in R^1$ represents costs associated to the final state $x(t_f)$ and $L \in R^1$ represents the running costs. The system description (1.1) is in state-space form. Any causal system can be easily put into this form, which has many advantages both from a theoretical and computational point of view.

The optimal control problem (1.1), (1.2) in general is infinite dimensional because the control trajectory $u(t)$ is continuous and infinite dimensional. To turn it into a finite dimensional problem we will apply control parameterisation. This can be done e.g. using piecewise polynomials or a piecewise constant or linear parameterisation. In the case of computer control, the control is truly piecewise constant. Therefore, this type of control parameterisation is used throughout the thesis. Furthermore a piecewise constant control is easily implemented. It is described by,

$$u(t) = u(t_k),\ t \in [t_k, t_{k+1}),\ k = 0,1,\dots,N-1 \qquad (1.3)$$

where $t_N = t_f$, and t_k, $k = 0,1,\dots,N$ are so called sampling instants which are usually equidistant i.e. $t_{k+1} - t_k = T_S$, $k = 0,1,2,\dots,N-1$, where T_S is the so-called sampling period. N is the number of time intervals. Introducing $u_k = u(t_k)$, $k = 0,1,\dots,N-1$ the control trajectory $u(t)$, $t_0 \le t \le t_f$ is now fully determined by u_k, $k = 0,1,\dots N-1$ and we may define $\tilde{u} = [u_1^T, u_2^T, \dots, u_{N-1}^T]$ the so-called control parameter vector which fully determines the control trajectory $u(t)$, $t_0 \le t \le t_f$. Given $\tilde{u}$ using the initial condition x_0 and numerical integration from (1.1) and (1.2) we may compute J. Therefore, the optimal control problem (1.1), (1.2) constitutes the minimization of J w.r.t. $\tilde{u}$.

If terminal state constraints $\psi(x(t_f)) = 0$ have to be satisfied the problem becomes a constrained function minimization problem. If the final time t_f instead of a-priori fixed is to be optimised as well, this is possible if we apply time scaling. In that case the interval $t_{k+1} - t_k = T_S = t_f / N$ over which $u(t_k) = u_k$ is applied, varies with t_f. To satisfy general state constraints, both the integral [3] and grid approximation approaches [25, 26] can be applied. Again this results in a constraint function minimization problem.

The resulting, possibly constraint function minimization problem often has a large number of variables and local minima. Because of this, local optimisers often fail in computing the true (global) minimum. Evolutionary Algorithms as investigated in this work might be able to overcome this difficulty because they are global in nature. Normally, only minor modifications of the previous description are necessary when evolutionary algorithms are applied to solve optimal control problems.

1.3 **Research objectives**

The main goal of this research is to investigate the possible advantages of the application of Evolutionary Algorithms as direct methods to solve optimal control problems.

The feasibility of Evolutionary Algorithms will be investigated to solve high dimensional, non-linear, multivariable and multimodal (with multiple local minima) optimal control problems. The optimal control problems are benchmark problems and an optimal control problem concerning greenhouse cultivation.

The expected advantage concerns mainly the ability to find the global optimal solution for multimodal problems having multiple (local) solutions. The efficiency of Evolutionary Algorithms in general compares unfavourably to the efficiency of other optimisation methods. Therefore the research will focus on evolutionary algorithms that are very efficient compared to other EA's. Application of such algorithms to solve multi-modal optimal control problems is a rather new area of research. Since algorithm parameter selection is an important practical issue when using any Evolutionary Algorithm this topic will be addressed too.

Given the objective to locate the global solution of multimodal optimal control problems and the lack of efficiency of Evolutionary Algorithms an approach where an EA algorithm is combined with a more efficient algorithm will also be investigated. Finally, Evolutionary Algorithms will be compared with other algorithms that have the potential of locating the global solution such as Iterative Dynamic Programming.

1.4 **Contributions of the thesis**

In this thesis efficient Differential Evolution algorithms, which are global optimisation methods, are proposed to solve multimodal optimal control problems. Differential Evolution algorithms are considerably more efficient and effective to solve optimal control problems than the majority of EAs. DE algorithms have advantages over other global search methods such as an Iterative Dynamic Programming or Controlled Random Search. They are very easy to implement and are easily adapted to solve constrained optimal control problems.

Many researchers believe that Evolutionary Algorithms are all inefficient in solving continuous optimisation problems. By showing some advantages of efficient evolutionary algorithms in solving hard optimal control problems this research will contribute to the acceptance of some state of the art evolutionary algorithms like Differential Evolution to solve practical problems, especially in the area of optimal control.

1.5 **Focus and limitations of this research**

Since numerical solutions for optimal control problems generally demand a high number of function evaluations, which involve a simulation of the system, they are computationally expensive. Theoretically and empirically it has been shown that GAs solve separable functions that are $O(n)$ hard in $O(n \ln n)$ time [17, 27], where $O()$ notation denotes the asymptotic order of growth of a function, e.g. the order of the largest term in $an^2 + bn + c$ is $O(n^2)$. In our case $O(n)$ refers to the number of function evaluations and n specifies the dimension of the optimisation problem. In case of functions with highly correlated variables traditional GAs tend to be even more inefficient. The cause of this is the high recombination probabilities and small mutations ($p_m = 1/n < 1$) that are common settings in GAs. Therefore, some of the state of the art evolutionary algorithms that have been proposed recently as good candidates to surmount these drawbacks are investigated. In contrast to Genetic Algorithms, Differential Evolution algorithms are efficient since it seems they use only $O(n)$ complexity. Also they are rotationally invariant [15] which means they do not lose performance due to correlated variables. Evolution Strategies and Evolutionary Programming are relatively efficient evolutionary algorithms but they are not considered in this work since they demand a higher computational complexity $O(n^2)$ than Differential Evolution as they include rotational angles. DE as well was ES and EP use a mutation probability $p_m = 1$ all the variables are mutated. This work does not provide theoretical results but is rather based on the analysis of some engineering applications of EAs, especially some classes of difficult optimal control problems.

1.6 Organization of the thesis

The thesis is organized in several chapters which can be grouped into three parts. The first part (chapter 2.1 and chapter 2.2) presents a general introduction to Evolutionary Algorithms (chapter 2.1) and discusses the issue of their application to some hard optimal control problems (chapter 2.2). The main purpose of chapter 2.1 is to summarize the most relevant work reported in the literature up until now regarding the application of the Evolutionary Algorithms (Genetic Algorithms, Evolution Strategies and Differential Evolution) to solve optimal control problems (OCP). Different kinds of representations of the individuals for some classes of OCP and the corresponding evolutionary operators are described. In chapter 2.2 the possibility of using Evolutionary Algorithms, with real-valued chromosomes representation and specialized evolutionary operators is studied. Some optimal control problems from chemical engineering characterized by being high-dimensional, non-linear, multivariable, multi-modal and non-differentiable are solved and results are compared with other direct methods commonly applied in optimal control.

The second part focuses on the study of Differential Evolution algorithms, which are considered as the state of the art evolutionary algorithms, designed in the field of continuous parameter optimisation. In contrast to genetic algorithms, DE algorithms are considerably more efficient and therefore constitute good candidates for solving hard dynamic optimisation problems. In chapter 3, DE's are studied by analysing how they perform on two multimodal (benchmark) optimal control problems. The performance of some evolutionary algorithms based on the Breeder Genetic Algorithm (BGA) is also analysed and results are compared to those obtained by DE algorithms. Finally, the results are also compared with Iterative Dynamic Programming, a global optimisation approach specifically designed for optimal control problems. Improvements of the DE algorithms are presented and tested in chapter 4. DE algorithms are efficient and easy to use evolutionary algorithms but require some tuning of the algorithm parameters: population size, mutation and crossover constants. Generally these parameters are kept constant during the optimisation process. A more effective algorithm may be obtained if they are adaptively tuned [15]. A parameter control strategy that adjusts the crossover and mutation constant in accordance with the diversity of the population is proposed and evaluated by using the benchmark multimodal dynamic optimisation problem studied in chapter 3.

The third part (chapters 5.1-5.3) of this work presents some applications concerning optimal control of greenhouse cultivation. Chapter 5.1 presents the use of a genetic algorithm with both binary and floating representations for the chromosomes, to estimate some of the parameters of a dynamic model of a lettuce crop. A two-state dynamic model of a lettuce crop (NICOLET) that predicts the nitrate concentration at harvest time is described first. Then, evolutionary algorithms are used to optimally fit the model parameters to measurements of dry weight of a lettuce crop. Results are compared against those obtained by some local search methods. In chapter 5.2 an optimal control problem of nitrate (NO_3) in lettuce is presented and solved by a first order gradient algorithm. First, A modified two-state dynamic model of a lettuce crop (NICOLET B3) is described. Next, an optimal control problem with fixed final time control constraints and terminal state constraints is put forward. Subsequently, a Differential Evolution algorithm is applied to get an approximate global solution. The DE algorithm is extended in order to deal with this. In chapter 5.3 a combination of a DE and a first order gradient algorithm is proposed to solve the optimal control problem of nitrates in lettuce. Finally, in chapter six the thesis ends with an overall discussion, conclusions and some suggestions for future research.

1.7 References

[1] Von Stryk O. and Bulirsch R., Direct and indirect methods for trajectory optimization, Annals of Operations Research 37(1992) 357-373.

[2] Bryson A.E, Jr., Dynamic Optimization, Addison-Wesley, 1999.

[3] Goh C.J. and Teo, K.L., Control Parametrization: a unified approach to optimal control problems with general constraints, *Automatica* Vol. 24, No. 1, pp 3-18, 1988.

[4] R. Luus, *Iterative Dynamic Programming* (Boca Raton, FL: Chapman & Hall/CRC, 2000).

[5] M.M. Ali, C. Storey, A. Törn, Application of stochastic global optimisation algorithms to practical problems, *Journal of optimization theory and applications 95*, 1997, 545-563.

[6] Banga J.R. and Seider W.D., Global optimization of chemical processes using stochastic algorithms, in State of the Art in Global Optimization, Floudas C.A. and Pardalos PM (Eds.), 563-583, Kluwer Academic, 1996.

[7] Bäck T., Evolutionary Algorithms in Theory and Practice, Oxford University Press, NY, 1996.

[8] Rudolph G., Convergence Properties of Evolutionary Algorithms, Verlag Dr. Kovač, Hamburg 1997.

[9] Holland J., Adaptation in Natural and Artificial Systems, Univ. of Michigan Press, Ann Arbor, MI, 1975.

[10] Goldberg D.E., Genetic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley Publishing Company, Inc 1989.

[11] Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin, 1996.

[12] Schwefel H.P., Evolution and optimum seeking, John Wiley, Chichester, UK, 1995.

[13] Fogel D., Evolutionary Computation: Toward a new philosophy in Machine Learning, Los Alamitos CA, 1995.

[14] Storn R. and Price K., Differential Evolution -A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, *Journal of Global Optimization* 11:341-359, 1997.

[15] Price K. V. An Introduction to Differential Evolution, in Corne D, Dorigo M., and Glover F. (Editors), New Ideas in Optimization, Mc Graw Hill, 1999.

[16] Koza J. Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press 1992.

[17] Mühlenbein, H. Schlierkamp-Voosen, D., Predictive models for the Breeder Genetic Algorithm: I Continuous Parameter Optimization, *Evolutionary Computation 1*(1), 1993, 25-49.

[18] Eiben A.E., Rudolph G., Theory of evolutionary algorithms: a bird's eye view, *Theoretical Computer Science* 229, 1999, pp. 3-9.

[19] Mitchell M. An introduction to genetic algorithms, The MIT Press, Cambridge Massachusetts, 1996.

[20] Vose, M.D. Random Heuristic Search, *Theoretical Computer Science* 229, 1999, 103-142.

[21] Vose M. D. The simple genetic algorithm: foundations and theory, The MIT Press, Cambridge Massachusetts, 1999.

[22] Rudolph G. Finite Markov Chain results in Evolutionary Computation: a tour d'Horizon, *Fundamenta Informaticae* 35, 1998, 67-89.

[23] Beyer H.G., The theory of Evolution Strategies, Springer-Verlag, 2000.

[24] Bryson, A .E. Jr., Ho, Y. Ch. Applied optimal control. Optimization, Estimation and Control, Hemisphere, 1975.

[25] Kraft D., TOMP- Fortran Modules for Optimal Control Calculations, *ACM Transactions on Mathematical Software*, Vol. 20, No. 3, September 1994, 262-281.

[26] Fabien, B.C., Some tools for the direct solution of optimal control problems, *Advances in Engineering Software*, 29, 45-61, 1998.

[27] Salomon R., Re-evaluating genetic algorithm performance under coordinate

rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms, *BioSystems 39* (1996), 263-278.

PART I

EVOLUTIONARY ALGORITHMS IN OPTIMAL CONTROL

2. Evolutionary algorithms for optimal control

2.1. Evolutionary Algorithms for optimal control: a survey

2.1.1 Abstract

The purpose of this survey is to present a summary of the most relevant research reported in the literature regarding the application of Evolutionary Algorithms (Genetic Algorithms, Evolution Strategies, and Differential Evolution) to solve optimal control problems. Emphasis is put on benefits and drawbacks of the proposed evolutionary algorithms. In addition, some general remarks concerning the main properties of the designed and applied evolutionary operators are underlined.

2.1.2 Introduction

To solve finite-horizon optimal control problems in continuous-time, by means of evolutionary algorithms, control parameterisation is applied to obtain a finite dimensional approximate description of the continuous-time control. A finite-horizon digital optimal control problem concerns the control of a continuous-time system by means of a digital computer. Due to the use of a computer, the control is piecewise constant (equation 1.3) and therefore finite dimensional. Within finite-horizon discrete-time optimal control problems the system description and the control are discrete in time. Again in this case, the control is finite dimensional. Therefore, without making any approximations, both finite-horizon digital and discrete-time optimal control problems can be solved by means of genetic algorithms. Finite horizon discrete-time optimal control problems are usually equivalent to, or an approximation of, digital optimal control problems [1, 2].

A certain type of continuous-time optimal control problems, with bounded control, is known a-priori to have an optimal control which is always at the bounds, except for certain switching times where it switches from one bound to the other. This type of control is called bang-bang control and is fully characterized by the switch times which are therefore the only variables that need to be optimised. Assuming the number of switches to be finite, again the control is finite-dimensional.

In the case of digital optimal control problems the sampling instants t_k, $k = 0,1,..$ are a-priori known. When applying direct methods for optimal we are also able to vary these sampling instants during the optimisation, to try to find a so called optimal sampling scheme, where the number of sampling instants is fixed but their values are free. Although this is uncommon in the control literature, people who solved optimal control problems by means of GA's, on several occasions, have done just this. Clearly optimising both the sampling scheme and the control significantly complicates the nature of the optimal control problem. In terms of control parameterisation, optimising the sampling scheme may be viewed as a special case of optimising the control parameterisation.

Table 1 presents an overview of different types of GA's that have been applied to solve the different types of optimal control problems mentioned above. In this table we distinguish CT referring to continuous-time optimal control problems, DT referring to discrete-time optimal control problems. SGA stand for Simple Genetic Algorithm, GENOCOP for GEnetic algorithm for Numerical Optimization for Constrained Problems, BGA for Breeder Genetic Algorithm, DE for Differential Evolution, ES for Evolution Strategy, GAs for Genetic Algorithms, and EP-SS for Evolutionary Program, with state-space representation.

This survey follows the information summarized in table 1. In section 2.1.3 Evolutionary Algorithms with binary representation are considered while chapter 2.1.4 considers genetic algorithms with floating-point representation. In section 2.1.5 Evolution Strategies are considered and finally in section 2.1.6 Differential Evolution algorithms. Instead of providing a detailed description of the evolutionary operators of each algorithm a general description is given together with possible advantageous and drawbacks.

Table 1. Evolutionary algorithms and types of optimal control problems

GAs	Binary	Bang-Bang optimal control *SGA* Seywald et al. 1995			CT singular optimal control *SGA* Yamashita & Shima, 1997		CT optimal control *SGA* Lee et al. 1997	
	Floating-point	DT *GENOCOP* Michalewicz 1994	DT & CT *BGA* Dakev et al. 1995	CT *EP-SS* Smith 1995	CT with changing controls *EP* Bobbin 1997	CT Multipopulation *BGAs* Polheim & Heibner 1996	CT singular smoother *GENOCOP* Roubus et al. 1999	CT Initial costates *SGA* Sim et al., 2000
ES	DT *Modified ES* Hashem et al. 1998		DT & CT *Modified ES* Pham, 1998				CT *Multi-population ES* Polheim & Heibner, 1997	
DE	CT Optimal time location DAE *DE/best/2/bin* Wang & Chiou, 1997		CT *Modified DE* Lee et al. 1999				CT *Hybrid DE* Chiou et al. 1999	

SGA : Simple Genetic Algorithm,GENOCOP : GEnetic algorithm for Numerical Optimization for Constrained Problems, BGA: Breeder Genetic Algorithm, DE: Differential Evolution, **ES**: Evolution Strategy, GAs: Genetic Algorithms, EP-SS: Evolutionary Program, with state-space representation.

2.1.3 Genetic Algorithms with binary representation in optimal control

A genetic algorithm with binary individuals (see chapter 1) was applied to solve optimal control problems [3] in which the cost functional is given by:

$$J = \phi(x(t_f), t_f) \tag{1}$$

The dynamic system is linear in the controls,

$$\dot{x}(t) = a(x(t),t) + \sum_{i=1}^{m} b_i(x(t),t)u_i(t) \tag{2}$$

with initial condition

$$x(t_0) = x_0 \tag{3a}$$

Terminal state constraints are represented by,

$$\psi(x(t_f), t_f) = 0 \tag{3b}$$

and the control constraints by

$$u_i(t) \in [0,1];\ i = 1,\dots,m \tag{4}$$

The final time t_f is free. According to optimal control theory the optimal control is bang-bang. Therefore one bit (0 or 1) of each individual was used to represent each control parameter. Additional bits were used to represent the unknown final time. The genetic algorithm was only applied to generate a solution by which a subsequent gradient method was initialised. Although knowing the bang-bang structure only switching times need to be optimised, in this paper the authors chose to use a piecewise constant approximation (equation 1.3.) of the control in conjunction with time-scaling to accommodate for the free final time.

A binary genetic algorithm was applied to solve an optimal control problem with singular arcs, terminal state constraints and free final time [4]. The mathematical description is given by equations (1)-(3). The control constraints are more complicated in this case,

$$u_{i,j\min}(x(t)) \leq u_{i,j}(t) \leq u_{i,j\max}(x(t)) \qquad (5)$$

The time interval $t \in [0, t_f]$ was scaled to $t \in [0,1]$ and the control inputs $u(t)$ to the interval $[0,1]$. Then, they were approximated by means of cubic Splines functions that used a minimum number of bits. A long string of bits was used to represent multi-input systems. Auxiliary cost functions and associated Lagrange multipliers were added to the individuals. Although in this way singular optimal control problems can be handled two important drawbacks remain: the inherent limitations of a string of bits to accurately represent variables and the poor efficiency of a simple genetic algorithm.

A binary genetic algorithm combined with heuristic constraints for the controls was applied to solve a time-optimal control problem of a continuous co-polymerisation reactor [5]. The continuous time optimal control problem is given by equations (1.1) and (1.2) as specified in chapter 1. However, the controls are constrained

$$u_{i,\min} \leq u_i(t) \leq u_{i,\max} \qquad (6)$$

and the final time (t_f) is free. A piecewise constant control parameterisation was used in conjunction with time-scaling to accommodate for the free final time. The vector $\widetilde{u} = [u_1^1, ..., u_1^n, ..., u_m^1, ..., u_m^n]$ containing all the control parameters was represented by a long binary string. In order to alleviate the computational load demanded by the GA, a two level hierarchical time-optimal control was implemented. At the highest level, an upper bound for the transition time (t_f) and steady state control inputs ($u_{i,ss}$) were calculated. At the lower level, the optimal control inputs and the minimum transition time were found using the steady state control from the highest level and a heuristic rule that reduces the range of control inputs. Two types of computations were performed. One in which the values of t_k, $k = 0,1,..,N$, apart from time-scaling, are fixed and one in which they are free to be able to exactly compute the switching times. Regarding efficiency, the number of function evaluations required by the GA without the heuristic rule turned out to be less efficient than Iterative Dynamic Programming, but the GA using the heuristic rule clearly outperformed IDP. The main drawback of this approach is the low efficiency associated with a binary GA.

2.1.4 Genetic Algorithms with floating-point individuals in optimal control

Michalewicz [6, 7] designed and applied genetic algorithms with a floating-point representation of the individuals and specialized operators to solve the linear-quadratic problem of the discrete-time scalar system,

$$x_{k+1} = ax_k + bu_k\,,\ k = 1,\dots,N-1 \qquad (7)$$

with the quadratic performance index,

$$J = qx_N^2 + \sum_{k=0}^{N-1}(sx_k^2 + ru_k^2) \qquad (8)$$

A discrete-time optimal control problem for the same scalar system with a non-quadratic performance index and an additional equality constraint was also solved. Finally, also a discrete-time optimal control problem where the system is second order and the performance index quadratic was solved.

The evolutionary algorithm termed GENOCOP in chapter 1 was evaluated against a Simple Genetic Algorithm (SGA with binary individuals) in solving the above-mentioned problems. GENOCOP was more efficient than SGA by several orders of magnitude. This is due to a more appropriate representation of the problem and the use of specialized evolutionary operators. Successive extensions of GENOCOP (GENOCOP-II, GENOCOP-III) confirmed that GENOCOP is one of the most efficient evolutionary algorithms. GENOCOP's main disadvantage is its large number of operators and algorithm parameters that a user has to specify before solving a particular optimal control problem.

An evolutionary program was proposed to solve continuous time optimal control problems, with constraints for the control inputs and fixed or free final time [8]. This approach uses a so-called state-space representation of the individuals (which is something else then a state-space realization of a system). The optimisation performed by this algorithm in addition to optimising the control approximated by splines and the time nodes $t_k,\ k = 0,1,..,N$ also optimises the number of time nodes N. It is argued that by optimising the time nodes and their number the algorithm is able to concentrate on areas were the control changes rapidly. This would allow for instance the exact solution of bang-bang optimal control problems. In this way, a better performance can be obtained by optimising the time nodes and their number. Accordingly the individual's representation several evolutionary operators were proposed: *random, perturbation, simple crossover and arithmetic crossover* (*blend*). In a subsequent paper [9] the approach was extended to solve constrained optimal control problems using penalty functions with time-varying coefficients. Although this evolutionary program worked well on the problems presented in the paper, its main limitation is that the number of the time nodes N may become very large or very small. The associated solutions, in general, are undesired.

A general approach to solve optimal control problems with general constraints by genetic algorithms with either binary or floating-point representation was proposed recently [10, 11]. The performance index is given by:

$$J = \phi(x(t_f)) \qquad (9)$$

subjected to the system dynamics:

$$\dot{x} = f(x,u,p,t)\,,\ x(t_o) = x_0 \qquad (10)$$

and general equality and inequality constraints represented by,

$$g(x(t),u(t),p,t)=0 \tag{11}$$

$$h(x(t),u(t),p,t)\le 0 \tag{12}$$

$$u_{i,\min}\le u_i(t)\le u_{i,\max} \tag{13}$$

In equation (9)-(12) p are the parameters of the dynamic system. The proposed evolutionary operators are those contained in the Genetic Algorithm Toolbox for use with Matlab [12]. Basically, this implementation follows the Breeder Genetic algorithm [13]. However, an important characteristic is the use of sub-populations with several topologies, which is argued, can improve the quality of the search and alleviate the high computational load. The main disadvantage of this approach is the lack of efficiency associated with the Breeder Genetic Algorithm, which is known nowadays to be efficient only in solving decomposable optimisation problems [14].

A method for the optimisation of dynamic processes by means of genetic algorithms with an integer-valued representation of the chromosomes was reported recently [15]. The controls were approximated by piecewise linear functions. Several optimal control problems with fixed final time and control constraints from bioengineering were solved. The genetic algorithm used a relatively small population of individuals. A so called elitist selection strategy with the roulette-wheel method was applied to select four individuals, which replace the worst individuals in the population. Classical crossover and mutation operators were used with a small probability of mutation and a high probability of crossover. The main drawback of this approach is the extensive tuning of algorithm parameters that is required by the algorithm. Also instead of an integer-valued representation a floating-point representation seems much more appropriate for solving optimal control problems.

The application of Evolutionary Algorithms to solve an optimal control problem with a control that can only take on certain discrete values and a cost associated to each switching time was reported lately [16]. This is a mixed continuous and discrete-time optimisation problem which is complex and non-convex. The dynamic system is described by,

$$\dot{x}(t)=f(x(t),u(t)),x(t_0)=x_0 \tag{14}$$

where $x(t)\in R^n, u(t)\subseteq U\subseteq R^m$.

In the case of a piecewise constant control function $u(t)$, the optimal control problem is to find the sequences t_i, $u(t_i)$, $i=0,1,...,n-1$ where $t_i<t_{i+1}$, $i=0,1,...,n-1$ which minimise the cost function:

$$\min\left\{\phi(x(t_f))+D\cdot\sum_{i=0}^{n-1}\max_j\left|u_j(t_{i+1})-u_j(t_i)\right|\right\} \tag{15}$$

where D is a constant matrix.

An individual within the algorithm has a variable length determined by the number of control changes (switching times). Given the cost on switching, an individual will not continue growing. Based on the algorithm representation the following evolutionary operators were proposed: *uniform-based crossover, mutation, blend,* and *insertion mutation.*

The combination of a genetic algorithm using individuals with a floating-point representation and a shooting method was proposed recently to solve optimal control problems with terminal state constraints and fixed or free final time [17]. In the latter case time-scaling is applied. Basically, the GA was applied to seek optimal initial values of the co-states. The fitness function consisted of the Hamiltonian of the optimal control problem. The genetic algorithm only computes a few generations to obtain an estimate of the final time and the initial values of the co-states needed to initialise the shooting method with. If the solution is not satisfactory the procedure is repeated.

An evolutionary approach based on the Breeder Genetic Algorithm was applied to solve a continuous time optimal control concerning greenhouse climate [18]. An important property of the evolutionary algorithm was the use of several sub-populations instead of just one population. In a subsequent paper, a comparison of the performance of two Evolutionary Algorithms on this problem was presented [19]. Both algorithms were able to solve the problem. The best result was obtained by combining them. The main limitation of this approach is the use of the Breeder Genetic Algorithm, which is not very efficient, although the use of sub-populations seems to improve the efficiency considerably.

Recently, the GENOCOP algorithm was extended with filter operators in order to allow for the calculation of smoother optimal control trajectories [20]. With this algorithm some optimal control problems having singular instead of bang-bang solutions were solved. The following two operators realize the smoothing. Firstly given an individual

$$\vec{x}' = (x_1, \ldots, x_k, x_{k+1}, \ldots, x_q) \qquad (16)$$

two neighbouring genes (x_k and x_{k+1}) are selected randomly and both are replaced by the average value $(x_k + x_{k+1})/2$. Secondly a least squares estimation of a line through five successive points ($x_{k-2}, \ldots x_{k+2}$) is performed. Then, a new individual

$$\vec{x}' = (x_1, \ldots, x_{k-1}, x_k, x_{k+1}, \ldots, x_q) \qquad (17)$$

is generated were x_{k-1}, x_k, and x_{k+1} are replaced by the corresponding estimates on the line.

2.1.5 Evolution Strategies in optimal control problems

An Evolutionary Algorithm inspired by ES has been proposed recently [21]. It has been applied to solve continuous and discrete-time optimal control problems with fixed final time and control constraints. A piece-wise constant approximation for the controls was applied. Floating-point numbers represent the individuals in the population. Several specialised evolutionary operators were designed to improve the local search capabilities of this algorithm. The recombination operators were *crossover*, *interpolation* and *extrapolation*. The mutation operators were *mutation and creep*, which used a Gaussian distribution, and also *shift*, *smoothing* and *swap*. The algorithm uses small population sizes (2, 4 and 8 individuals), which account for the observed high efficiency. Although the algorithm has been used successfully to solve some practical problems from chemical engineering the number of evolutionary operators and the associated algorithm parameters that need to be tuned present a serious drawback.

Evolution Strategies $(\mu+\lambda)$-ES improved with an arithmetical crossover operator, having sub-populations, and a time-variant mutation operator, were used to solve two types of discrete-time optimal control problems [22]. These two types of problems are the linear quadratic problem and the push-cart discrete-time control problem. On these examples this algorithm outperformed a classical Evolution Strategy in terms of both convergence and accuracy of the solution.

2.1.6 Differential Evolution in optimal control problems

To generate initial starting points for an SQP algorithm Wang and Chiou [23] have applied the Differential Evolution algorithm *DE/best/2/bin* to solve an optimal control concerning a differential-algebraic system. This dynamic system is described by

$$E\frac{dx}{dt} = f(x(t), z(t), u(t), t) \qquad (18)$$

for $t \in [0, t_f]$, where $x \in R^{n_x}$, $z \in R^{n_y}$, $u \in R^m$, $f = [f_1^T f_2^T]$, and $E = \begin{bmatrix} I_{n_x \times n_x} & 0 \\ 0 & 0 \end{bmatrix}$.

The final time t_f can be fixed or free. Here the vectors x and z contain the dynamic states and steady-state variables respectively. A special feature of this optimal control problem is that two of the initial state variables $x_1(0), x_2(0)$ of the initial state vector $x(0) = x_0$ are optimised as well. The optimal control problem then is to find a control function $u(t) \in R^m$ and the two initial values $x_1(0), x_2(0)$ which minimises

$$J = \phi(x(t_f), p, t_f) \qquad (19)$$

subjected to the dynamic equations (17) and the additional constraints:

$$h(x(t), z(t), u(t), p, t) = 0 \qquad (20)$$

$$g(x(t), z(t), u(t), p, t) \le 0 \qquad (21)$$

In addition the control variables and $x_1(0), x_2(0)$ are bounded:

$$u_{\min} \le u(t) \le u_{\max} \qquad (22)$$

$$x_{1\min} \le x_1(0) \le x_{1\max},\ x_{2\min} \le x_2(0) \le x_{2\max} \qquad (23)$$

A piecewise constant control parameterisation is used and the time nodes t_k, $k = 0,1,..,N$, which determine the parameterisation, are optimised as well but there number N is fixed a-priori. Time-scaling is used to accommodate for a free final time. In order to deal with the constraints (20) and (21) penalty functions were used.

Lee et al. [24], proposed a modified Differential Evolution algorithm to solve the time-optimal control problem described in [5]. The DE applied is the one in which the mutated vector combines each target vector with the difference between two randomly selected vectors plus the difference between the best current solution in the population and the target vector. So the mutation operator is described by:

$$\vec{a}_i' = \vec{a}_i + F \times (\vec{a}_{best} + \vec{a}_{r_1} - \vec{a}_i - \vec{a}_{r_2});\ i = 1,...,\mu \qquad (24)$$

The modification introduced, basically adds to the main loop involved in DE algorithm a secondary loop, in such a way that after a trial vector $\vec{a}_i''$ was created by the mutation equation (24) and binomial crossover (see section 1.2.1.3 in chapter 1) if the condition $\Phi(\vec{a}_i'') < \Phi(\vec{a}_i)$ is satisfied then a 'local search' is implemented as follows:

```
do
    $\vec{a}_i''' = \vec{a}_i''$
    $F = \xi F$
    $\vec{a}_i' = \vec{a}_i + F \times (\vec{a}_{best} + \vec{a}_{r_1} - \vec{a}_i - \vec{a}_{r_2})$ ; $i = 1,\dots,\mu$
    $a_{ji}'' = \begin{matrix} a_{ji}' \text{ if } randb(j) \le CR \text{ or } j = rnbr(i) \\ a_{ji} \text{ if } randb(j) > CR \text{ and } j \ne rnbr(i) \end{matrix}, i = 1,2,\dots,\mu\}, j = 1,2,\dots,n\}$
while $\Phi(\vec{a}_i''') < \Phi(\vec{a}_i'')$
```

where ξ is a coefficient that allows to increase the value of the mutation parameter (F) and the vectors $\vec{a}_i$, $\vec{a}_{best}$, $\vec{a}_{r_1}$ and $\vec{a}_{r_2}$ are the same vectors used at the main loop. Although improvements regarding function evaluations were reported this use of the algorithm parameter (F) seems to contradict the role it is known to play in DE algorithms since by increasing the value of F the global search capability of DE is increased instead of local search potential.

A modified Differential Evolution algorithm was proposed lately [25] to resolve optimal control problems with decision parameters and general constraints. The dynamic system is described by:

$$\dot{x} = f(x(t), u(t), p, t),\ t \in [0, t_f] \tag{25}$$

where $x(t) \in R^n$ are the states, $u(t) \in R^m$ are the control inputs, and $p \in R^q$ is a vector of decision parameters. The initial conditions $x(0) = x_0$ are given or can be considered as decision parameters. The constraints are:

$$h_k(x(t), u(t), p, t) = 0,\ k = 1,\dots,n_e,\ n_e < n \tag{26}$$

$$g_k(x(t), u(t), p, t) \le 0,\ k = 1,\dots,n_d \tag{27}$$

Additionally control and decision parameter constraints may be included:

$$p_{i\min} \le p_i \le p_{i_{\max}},\ i = 1,2,..,q \tag{28}$$

A piecewise constant control parameterisation is used in which the time nodes t_k, $k = 0,1,..,N$ and their number N are a-priori fixed. Two new evolutionary operators that supposedly make DE algorithms more efficient were proposed and tested using the *DE/best/2/bin* algorithm. The *acceleration* operator was introduced to increase the speed of convergence of the algorithm. It uses the best current solution in the population to start a local search as follows:

$$\vec{a}_{best}'(g) = \begin{cases} \vec{a}_{best}(g) & \text{if } J(\vec{a}_{best}(g) < J(\vec{a}_{best}(g-1)) \\ \vec{a}_{best}(g) - \alpha \nabla J & \text{otherwise} \end{cases} \tag{29}$$

where ∇J is the gradient of the objective function calculated by finite differences, and g denotes the current generation. A *migration* operator is proposed to avoid convergence to local minima caused by the previous operator, which is applied only if the diversity of the population is less than a specified tolerance. In that situation perturbing the best current solution present in the population generates new individuals:

$$\vec{a}_i(g) = \vec{a}_{best}(g) + N(0,\sigma)\,,\ i = 1,\ldots,\mu\,,\ i \neq best \qquad (30)$$

where $N(0,\sigma)$ denotes a vector of independent random Gaussian numbers with mean zero and standard deviation σ.

Even though this approach worked well compared to a classical genetic algorithm a detailed study of its performance compared to DE algorithms has not been made yet. Furthermore this hybrid algorithm is considerably more complex, since it has more algorithm parameters than a classical DE. Nonetheless the proposed operator to accelerate the convergence of DE algorithms deserves additional investigation. This investigation should also include the influence of the common algorithm parameters in DE (population size, mutation parameter) on the convergence and premature convergence.

2.1.7 Conclusions.

From the previous review it is clear that mainly binary-coded genetic algorithms and real-valued evolutionary algorithms have been applied to calculate solutions of optimal control problems while Evolution Strategies and Differential Evolution algorithms only rarely have been applied. In several references using GA's the control parameterisation was not fixed a-priori but was itself part of the optimisation. In terms of digital control problems, where the control is piecewise constant, this can be interpreted as optimising the sampling scheme. Optimising the control parameterisation however highly complicates the optimal control problem, which enlarges the probability of finding undesired or local solutions, despite the global nature of GA's. Roughly speaking GA's allow optimising almost anything at the same time. This possibility carries the danger of not carefully selecting, and judging mathematically, the nature and meaningfulness of the problem.

To the best knowledge of the author, the issue of efficiency of Evolutionary Algorithms in solving optimal control problems has been analysed only superficially in the literature. The same holds for the potential advantages of Evolutionary Algorithms to solve multi-modal optimal control problems.

The aim of this thesis therefore is to investigate and search for efficient GA's to solve multi-modal optimal control problems. An important additional requirement for the practical application of these algorithms is that they should require no algorithm parameter tuning, or only a little. Algorithms with less algorithm parameters should therefore be preferred and guidelines for the choice algorithm parameters and the possible development of automatic algorithm parameter tuning strategies are important issues.

2.1.8 References.

[1] Van Willigenburg L. G. and De Koning W.L., The digital optimal regulator and tracker for stochastic time-varying systems, Int. J. Systems Sci. 1992, Vol. 23, No. 12, 2309-2322.

[2] Van Willigenburg L. G. and De Koning W.L., Derivation and computation of the digital LQG regulator and tracker for time varying systems in the case of asynchronous and aperiodic sampling, Control-Theory and Advanced Technology, Vol. 10, No. 4, Part. 5, 1995, 2083-2098.

[3] Seywald H. and Kumar R.R. Genetic Algorithm Approach for Optimal Control Problems with Linearly Appearing Controls, *Journal of Guidance, Control and Dynamics* Vol. 18, No. 1, January-February 1995, 177-182.

[4] Yamashita Y. and Shima M., Numerical computational method using genetic algorithm for the optimal control problem with terminal constraints and free parameters, *Nonlinear Analysis, Theory, Methods & Applications*, Vol. 30 No. 4, pp. 2285-2290, 1997

[5] Lee M.H., Han Ch., Chang, K.S., Hierarchical time-optimal control of a continuous co-polymerization reactor during start-up or grade change operation using genetic algorithms, *Computers Chem. Engng.* Vol 21, Suppl., pp S1037-S1042, 1997.

[6] Michalewicz Z., Janikov C.Z., Krawczyk J.B., A modified Genetic Algorithm for optimal control problems, *Computers, Math. Applic.* Vol. 23, No. 12, pp. 83-94, 1992.

[7] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin, 1996.

[8] Smith S., An Evolutionary Program for a Class of Continuous Optimal Control Problems, *Proceedings of the IEEE Conference on Evolutionary Computation,* Vol. 1 1995, IEEE, Piscataway, NJ, USA, 418-422.

[9] Smith S. and Stonier R., Applying Evolution Program Techniques to Constrained Continuous Optimal Control Problems, *Proceedings of the IEEE Conference on Evolutionary-Computation*, 1996, IEEE, Piscataway, NJ, USA, 285-290.

[10] Dakev N.V., Chipperfield A.J., Fleming P.J., A General approach for solving optimal control problems using optimization techniques, *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics*, Part 5 (of 5), Vancouver, BC, Can, 4503-4508.

[11] Dakev N.V., Chipperfield A.J., Whidborne J.F., Fleming P.J., An Evolutionary Algorithm Approach for Solving Optimal Control Problems, *Proceedings of the 13th Triennial IFAC World Congress*, San Francisco, USA, 1996, pp 321-326.

[12]. Chipperfield A., Flemming P, Pohlheim H., Fonseca, C., Genetic Algorithm Toolbox for use with MATLAB, Department of Automatic Control and Systems Engineering, University of Sheffield, User's guide, version 1.2, 1994.

[13] Mühlenbein, H. Schlierkamp-Voosen, D., Predictive models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization, *Evolutionary Computation 1*(1), 1993, 25-49.

[14] Salomon R., Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms, *BioSystems 39* (1996), 263-278

[15] de Andres-Toro B. Giron-Sierra J.M., Lopez-Orozco J.A., A genetic optimisation method for dynamic processes, *Proceedings of the 14th Triennial IFAC World Congress*, Beijing P.R. China, 1999, 373-378.

[16] Bobbin J. and Yao X., Solving Optimal Control Problems with a cost on Changing Control by Evolutionary Algorithms, *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation,* ICEC'97, Indianapolis, IN, USA, 331-336.

[17] Sim Y.C., Leng S.B., Subramaniam V., A combined genetic algorithms-shooting method approach to solving optimal control problems, *International Journal of Systems Science*, volume 31, number 1, 2000, 83-89.

[18] Pohlheim H., Heißner A., Optimal control of greenhouse climate using Genetic Algorithms, MENDEL'96 –2nd International Conference on Genetic Algorithms, Brno Czech Republik, 112-119, 1996.

[19] Pohlheim H., Heißner A., Optimal Control of Greenhouse Climate Using a Short Time Climate Model and Evolutionary Algorithms, Preprints Mathematical and control applications in agriculture and horticulture, Edited by A Munack and H.J. Tantau, IFAC Workshop, Hannover, Germany 28 september – 2 October, 1997, 113-118.

[20] Roubos J.A., Van Straten G., Van Boxtel A.J.B., An evolutionary strategy for fed-batch bioreactor optimization: concepts and performance, Journal of Biotechnology 67 (1999), 173-187.

[21] Pham Q. T., Dynamic optimization of chemical engineering processes by an evolutionary method, *Computers Chem. Engng.* Vol. 22, No. 7-8, 1089-1097, 1998.

[22] Hashem M.M.A., Watanabe K., Izumi K., A new Evolution Strategy and Its Application to Solving Optimal Control Problems, *JSME International Journal*, Series C, Vol. 41, No. 3 1998, 406-412.

[23] Wang F.S., Chiou J.P., Optimal control and optimal time location problems of differential-algebraic systems by differential evolution, *Ind. Eng. Chem. Res.* 1997, 36, 5348-5357.

[24] Lee, M.H., Han Ch., Chang, K S., Dynamic optimisation of continuous polymer reactor using a modified differential evolution algorithm, *Ind. Eng. Chem. Res.* 1999, 38, 4825-4831.

[25] Chiou J.P., Wang F.S. Hybrid method of evolutionary algorithms for static and

dynamic optimization problems with application to a fed-batch fermentation process, *Computers and Chemical Engineering* 23 (1999) 1277-1291.

2.2. Evolutionary algorithms for optimal control of chemical processes[†]

2.2.1 Abstract

Because many approaches to Optimal Control problems may find local solutions, global optimisation methods are of interest. In the past, global optimisation methods such as Iterative Dynamic Programming, Stochastic Algorithms and Genetic Algorithms have been used to solve dynamic optimisation problems in chemical engineering. Recently, Evolutionary Algorithms have been proposed to solve global optimisation problems and there is an increasing interest to use them in the solution of high-dimensional, non-linear, multivariable and multi-modal dynamic optimisation problems. In this work an Evolutionary Algorithm, which uses a real valued chromosomes representation and specialised evolutionary operators is studied by solving some dynamic optimisation problems taken from the chemical engineering literature. Results show that this approach is suitable and can have some advantages over other approaches used by researchers so far.

Keywords: Chemical processes, Evolutionary Algorithms, Optimal Control, Optimization.

2.2.2 Introduction

Numerical methods for the solution of optimal control problems can be roughly divided in two groups: indirect and direct methods [1, 2]. The first group is based on finding a solution that satisfies the Pontryagin's maximum principle or the related first-order necessary optimality conditions through solving a two-point boundary-value problem [3]. These methods may find local solutions. Direct methods are based on an approximation of the infinite dimensional optimal control problem by a finite dimensional optimisation problem, or non-linear programming (NLP) problem. This transformation can be done by either control and state parameterisation, or only control vector parameterisation [4]. The parameterisation may introduce additional local minima [1]. Gradient-based optimisation techniques may converge to a local optimum [5]. To surmount this problem, global optimisation algorithms can be used, for instance Iterative Dynamic Programming [6], Stochastic Algorithms [7], and Evolutionary Algorithms [5, 8].

In this research, the use of optimisation algorithms inspired by natural evolution is investigated and their performance is compared with solutions obtained using well-known approaches in chemical engineering. Firstly, a general description of a class of optimal control problems we are interested in is given. Next, a general description of an Evolutionary Algorithm is provided, and the properties of the approach proposed here are outlined. Finally, some examples of optimal control of chemical processes are solved to illustrate how this paradigm can be applied. The results are compared

[†] Published by I.L. Lopez Cruz, L.G. Van Willigenburg and G. Van Straten in Proceedings of the IASTED International Conference Control and Applications (CA'2000), May 24-27, Cancun, Mexico, pp.155-161.

with Sequential Quadratic Programming (SQP) algorithms, Iterative Dynamic Programming (IDP) and Stochastic Algorithms.

2.2.3 The optimal control problem

We consider the class of optimal control problems in which the plant is described by the non-linear time-varying dynamic equation

$$\dot{x}(t) = f(x(t), u(t), t) \qquad (1)$$

where $x(t) \in \Re^n$ is the state and $u(t) \in \Re^m$ is the control. The control inputs are constrained,

$$\alpha_i(t) \le u_i(t) \le \beta_i(t),\ i = 1,2,..,m, \qquad (2)$$

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

$$x(0) = x_0, \qquad (3)$$

is the initial condition. The performance index associated with the system is,

$$J(u(t)) = \phi(x(t_f), t_f) + \int_0^{t_f} L(x(t), u(t), t)dt, \qquad (4)$$

where [0, t$_f$] is the interval of interest. The final time t$_f$ is fixed. The optimal control problem is to find the input u*(t) on the time interval [0, t$_f$] that drives the plant along the trajectory x*(t) such that the cost function *J(u(t))*, (equation (4)) is minimised [3, 9]. The previous so called Bolza formulation of an optimal control problem can be, conveniently, rewritten as the Mayer formulation that yields a simpler expression that makes it more suitable for a numerical solution. By introducing a new state variable,

$$x_{n+1}(t) = L(x(t), u(t), t),\ x_{n+1}(0) = 0, \qquad (5)$$

equation (1) is re-written as,

$$\dot{x}'(t) = f(x'(t), u(t), t), \qquad (6)$$

where $x'(t) \in \Re^{n+1}$ and the performance index is converted to:

$$J(u(t)) = \phi'(x'(t_f), t_f) = \phi(x(t_f), t_f) + x_{n+1}(t_f), \qquad (7)$$

where,

$$\phi' : \Re^{n+1} \to \Re. \qquad (8)$$

In order to use direct methods to solve this class of optimal control problems, the continuous problem has to be transformed into a non-linear programming problem or parameter optimisation problem. Let the time interval *t* 0 [*t$_0$*, *t$_f$*] be discretized using N control time nodes t_i such that: $0 = t_0 < t_1 <,...,< t_N = t_f$. In each of these time intervals the control $u(t)$ is approximated by a polynomial in *t*. The control time nodes and the coefficients of the polynomials then uniquely define the control. For instance, using a piecewise constant approximation the unknown control input $u(t)$ is determined by,

$$u(t)=u(t_k)=u_k,\ t\in[t_k,t_{k+1}),\ k=0,1,..,N-1 \tag{9}$$

So the control is fully determined by the vector,

$$\tilde{u}=\left[u_0^T\ u_1^T\ ..\ u_{N-1}^T\right]^T \tag{10}$$

Denote by $\tilde{u}_i$ the ith element of $\tilde{u}$. Now, with a guess value for the control history $\tilde{u}$ and using the appropriate piecewise polynomial approximation it is possible to integrate the dynamic equation (eqn 6). A state trajectory $\tilde{x}'(\tilde{u},t)$ is obtained, from which in turn, the parameterised performance index $J(\tilde{u})$ (eqn 7) can be calculated. In this way, the Non-linear Programming problem approximation for the Optimal Control Problem defined before can be written as follows,

$$\min J(\tilde{u}) \tag{11}$$

subject to,

$$\alpha_{mk+i}\le\tilde{u}_{mk+i}\le\beta_{mk+i},\ k=0,1,..,N-1, i=1,2,..,m \tag{12}$$

where

$$\alpha_{mk+i}=\max(\alpha_i(t)),\ t\in[t_k,t_{k+1}),\ k=0,1,..,N-1,\ i=1,2,..,m \tag{13}$$

$$\beta_{mk+i}=\min(\beta_i(t)),\ t\in[t_k,t_{k+1}),k=0,1,..,N-1,\ i=1,2,..,m \tag{14}$$

2.2.4 Evolutionary Algorithms

A generic description of an Evolutionary Algorithm is presented in figure 1 [10, 11]. An Evolutionary Algorithm is a stochastic search method, which maintains a population $P(t):=\{\vec{a}_1(t),...,\vec{a}_\mu(t)\}$ of individuals (chromosomes) $\vec{a}_i\in I$, at generation *t*, where I is a space of individuals, and μ is the population size. Each individual represents a potential solution and it is implemented as some generic data structure. 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 better solutions.

Each individual is evaluated by a fitness function $\Phi: I\rightarrow\Re$, and real values are assigned to each potential solution that is a measure of how individuals perform in the problem domain. Next, an iterative process starts in which a set of evolutionary operators [12] is applied to the population in order to generate new individuals. From a set

```
Procedure Evolutionary Algorithm
t := 0;
Generate $P(0):=\{\vec{a}_1(0),...,\vec{a}_\mu(0)\}\in I^\mu$;
Evaluate $P(0):\{\Phi(\vec{a}_1(0)),...,\Phi(\vec{a}_\mu(0))\}$;
While ($\iota$(P(t))≠true) do
        Recombine $P'(t):=r\Theta_r(P(t))$;
        Mutate $P''(t):=m\Theta_m(P'(t))$;
        Evaluate $P''(t):\{\Phi(\vec{a}''_1(t)),...,\Phi(\vec{a}''_\lambda(t))\}$;
        Select $P(t+1):=s\Theta_s(P''(t)\cup Q)$;
        t:=t+1;
                End
```

Figure 1. Structure of an Evolutionary Algorithm.

$\{w\Theta_1,\ldots,w\Theta_z \mid w\Theta_i : I^{\lambda} \to I^{\lambda}\} \cup \{w\Theta_0 : I^{\mu} \to I^{\lambda}\}$ of probabilistic evolutionary $w\Theta_i$ operators (for instance: crossover, mutation), each one specified by parameters given in the sets $\Theta_i \subset \Re$, some operators are applied to the population and a new evaluation of its fitness is calculated. A recombination (crossover) operator $r\Theta_r : I^{\mu} \to I^{\lambda}$ and a mutation operator $m\Theta_m : I^{\lambda} \to I^{\lambda}$ are applied to the population. A selection operator $s\Theta_s : \left(I^{\lambda} \cup I^{\mu+\lambda}\right) \to I^{\mu}$ which may modify the number of individuals from λ to λ+μ, is applied as well, where $\lambda, \mu \in \mathrm{N}$, and μ denotes the number of parent individuals and λ the number of offspring. As before a set of parameters Θ_s may be used by the selection operator, and $Q \subset P(t)$ denotes an additional set of individuals. $\iota : I^{\mu} \to \{\mathit{true}, \mathit{false}\}$ is 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 solution.

Three main streams of evolutionary algorithms can be identified: Genetic Algorithms [13], Evolution Strategies [14], and Evolutionary Programming [15]. They have some differences but also they share a number of similarities. Moreover, it is clear that for the last several years the boundaries between these approaches have broken down to some extend [16]. Recently, researchers have started to apply this paradigm to solve optimal control problems in chemical engineering and in bioprocessing. A genetic algorithm was used to solve a hierarchical time-optimal control of a continuous co-polymerisation reactor during start-up or grade change operation [17]. Pham [8] investigated the performance of an evolutionary method for the dynamic optimisation of several Continuous Stirred Tank Reactor (CSTRs) problems. The application of a genetic dynamic optimisation method to solve several problems in bioprocessing is given in [18]. The performance of an evolutionary strategy for fed-batch bioreactor optimisation has been investigated in [19].

The evolutionary algorithms used during this research have the following specific properties: the chromosome representation is a real-valued vector where each real number represents the value of an unknown parameter ($\tilde{u}_i$) of the approximated control input *u(t)*. The starting values for the parameters are calculated randomly inside the boundaries of the control-input (eqn 2). In order to represent a multi-input control system, a long chromosome of real-values was created. This has several advantages over the classical binary code of a canonical genetic algorithm [11]. The evaluation of the population consists of the integration of the set of dynamic equations (eqn 6) in the specified time interval, using the values of the parameterised control inputs and the calculation of the performance index (eqn 7). Our solution uses a C-MEX file s-function and the ODE45 integration routine available in MATLAB to integrate the dynamic equations (eqn 6).

A brief description of the evolutionary operators used for this work is provided next. Additional information is available elsewhere [11, 20]. Three recombination (crossover) operators were used: arithmetic, heuristic and simple crossover. Let $\vec{A}(t)$ and $\vec{B}(t)$ be two n-dimensional row vectors denoting parents from the population P(t). The arithmetic crossover operator generates two linear combinations of the parent's solutions:

$$\vec{A}'(t) = r\vec{A}(t) + (1-r)\vec{B}(t) \tag{15}$$

$$\vec{B}'(t) = (1-r)A(t) + r\vec{B}'(t) \tag{16}$$

where r is a random number from a uniform distribution from 0 to 1.

The heuristic crossover operator uses the individual's fitness information to generate a new solution:

$$\vec{A}'(t) = \vec{A}(t) + r(\vec{A}(t) - \vec{B}(t)) \cdot if \cdot \Phi(\vec{A}(t)) \leq \Phi(\vec{B}(t)) \tag{17}$$

and $\vec{A}(t)$ is feasible according to:

$$feasibility \ \ \vec{A}'(t) = \begin{Bmatrix} 1, if \cdot a_i' \geq \alpha_i, a_i' \leq \beta_i \forall_i \\ 0, otherwise \end{Bmatrix} \tag{18}$$

If after a number of attempts no new feasible solution is found, the operator produces no offspring

$$\vec{A}'(t) = \vec{A}(t) \tag{19}$$

with α_i, β_i defining the lower and upper bounds respectively for each variable i.

The simple crossover operator generates a random number r from a uniform distribution from 1 to m and creates two new individuals according to:

$$\vec{A}'(t) = a_i' = \begin{Bmatrix} a_i, if \cdot i < r \\ b_i, otherwise \end{Bmatrix} \tag{20}$$

$$\vec{B}'(t) = b_i' = \begin{Bmatrix} b_i, if \cdot i < r \\ a_i, otherwise \end{Bmatrix} \tag{21}$$

Four mutation operators also were used, namely: boundary, multi-non-uniform, non-uniform and uniform mutation. Boundary mutation randomly selects one variable j and sets it equal to either its lower or upper bound, where r is a random number from a uniform distribution from 0 to 1.

$$\vec{A}'(t) = a_i' = \begin{Bmatrix} \alpha_i, if \cdot i = j, r < 0.5 \\ \beta_i, if \cdot i = j, r \geq 0.5 \\ a_i, otherwise \end{Bmatrix} \tag{22}$$

Uniform mutation randomly selects one variable j and sets it equal to an uniform random number $U(\alpha_i, \beta_i)$:

$$\vec{A}(t) = a_i' = \begin{Bmatrix} U(\alpha_i, \beta_i) \cdot if \cdot i = j \\ a_i, otherwise \end{Bmatrix} \tag{23}$$

Non-uniform mutation randomly select one variable j and sets it equal to an non-uniform random number:

$$\vec{A}(t) = a_i' = \begin{Bmatrix} a_i + (\beta_i - a_i)g(t) \cdot if \cdot r_1 < 0.5 \\ a_i - (a_i + \alpha_i)g(t) \cdot if \cdot r_1 \geq 0.5, \\ a_i, otherwise \end{Bmatrix} \tag{24}$$

where $g(t) = \left(r_2 (1 - \frac{t}{g_{\max}}) \right)^b$ (25)

r_1 and r_2 are uniform random numbers between (0,1), t is the current generation number of the evolutionary algorithm and $g_{\max}$ is the maximum number of generations, b is a shape parameter. The multi-non-uniform mutation operator applies a non-uniform mutation operator to all the variables in the parent $\vec{A}(t)$.

The normalised geometric ranking operator proposed in [21] was used for the selection of individuals for the next generation. It is a ranking selection method that calculates the probability of an individual to be selected as:

$$p(i) = q'(1-q)^{r-1}, \; q' = \frac{q}{1-(1-q)^{\mu}} \tag{26}$$

where q is the probability of selecting the best individual (generation gap), r is the rank of the individual, μ is the population size. The software implementation for the MATLAB environment, Genetic Algorithms for Optimization Toolbox (GAOT) [20] was used for all our investigations.

2.2.5 Case studies and results

2.2.5.1 High-dimensional non-linear continuous stirred tank reactor

In [6] Luus presents a high-dimensional and non-linear chemical process, which includes five simultaneous chemical reactions in an isothermal continuous stirred tank reactor. The system in Mayer form is described by eight differential equations:

$$\dot{x}_1 = u_4 - qx_1 - 17.6x_1x_2 - 23x_1x_6u_3 \tag{27}$$

$$\dot{x}_2 = u_1 - qx_2 - 17.6x_1x_2 - 146x_2x_3 \tag{28}$$

$$\dot{x}_3 = u_2 - qx_3 - 73x_2x_3 \tag{29}$$

$$\dot{x}_4 = -qx_4 + 35.2x_1x_2 - 51.3x_4x_5 \tag{30}$$

$$\dot{x}_5 = -qx_5 + 219x_2x_3 - 51.3x_4x_5 \tag{31}$$

$$\dot{x}_6 = -qx_6 + 102.6x_4x_5 - 23x_1x_6u_3 \tag{32}$$

$$\dot{x}_7 = -qx_7 + 46x_1x_6u_3 \tag{33}$$

$$\dot{x}_8 = 5.8(qx_1 - u_4) - 3.7u_1 - 4.1u_2 + q(23x_4 + 11x_5 + 28x_6 + 35x_7) - 5u_3^2 - 0.099 \quad (34)$$

where the states denote concentrations,

$$q(t) = u_1(t) + u_2(t) + u_4(t), \quad (35)$$

is the total feed rate and $u_3(t)$ is the squared root of the light intensity. The time interval is given: $0 \le t \le t_f = 0.2h$, and the initial conditions are:

$$\mathbf{x}(0) = [0.1883, 0.2507, 0.0467, 0.0899, 0.1804, 0.1394, 0.1046, 0]^T \quad (36)$$

With the constraints for the control inputs:

$$0 \le u_1(t) \le 20 \quad (37)$$
$$0 \le u_2(t) \le 6 \quad (38)$$
$$0 \le u_3(t) \le 4 \quad (39)$$
$$0 \le u_4(t) \le 20 \quad (40)$$

Thus, the optimal control problems consists in finding the control histories $u_1(t)$, $u_2(t)$, $u_3(t)$, $u_4(t)$ in the specified time interval such that the performance index:

$$J = x_8(t_f) \quad (41)$$

related with the economic profit is maximised.

In order to solve this problem by means of a direct approach, the time interval was divided in the same number of intervals used by Luus [6], N=10, when three control values are taken into account ($u_4(t)$=6.0) and N=11 in the case of four control values. The value of the performance index obtained by the evolutionary algorithm in the first case was J=20.0893 with a population size of 40 individuals and 2000 generations. No considerable improvement was obtained with more individuals and an increasing number of generations since with a population size of 60 chromosomes and 3000 generations a value of J=20.0896 was obtained. The solution obtained by Luus [6], using Iterative Dynamic Programming, was J=20.0895. The best value from several runs (ten) using a SQP-based algorithm (*fmincon.m* function from MATLAB's optimisation Toolbox) was 19.6234. The optimal controls are shown in figure 2. Shapes of the control functions obtained by the Evolutionary Algorithm are almost equal to those obtained by Luus [6]. For N=40, which is identical to Luus [6], J=20.0924 was obtained using a population size of 40 individuals and 2000 generations. However, with an increasing number of individuals (100) and more generations (3000) this value was J=20.0935, against a value of J=20.0953 obtained by Luus [6].

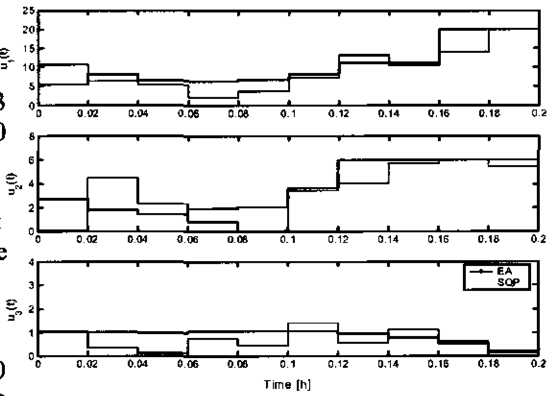


Figure 2. Optimal control of a CSTR by EAs and SQP.

In the case of four controls a value of J=21.7574 was obtained using a population size of 60 individuals and 3000 generations, against J=21.7572 reported using Iterative Dynamic Programming. As expected the SQP algorithm was unable to approximate the IDP solution reported previously by Luus [6]. From ten optimisations its best value was J=20.5105. Shapes of the controls calculated by Sequential Quadratic Programming, the Evolutionary Algorithm and those reported for IDP are presented in figure 3. We can see that for control u₃(t) and u₄(t) the plots of IDP and EAs completely overlap and tiny differences are shown in the case of controls u₁(t) and u₃(t). On the other hand the pattern of the solutions found by the SQP algorithm is completely different.

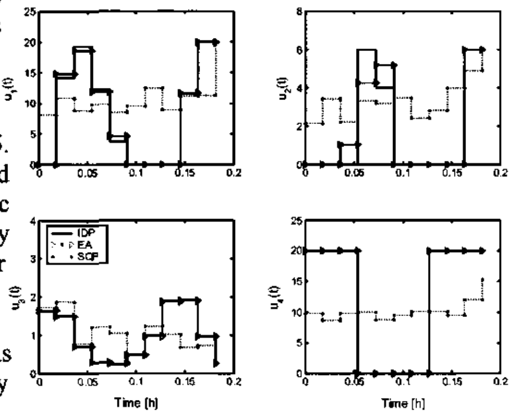


Figure 3. Optimal control of a CSTR with four controls by EAs.

2.2.5.2 Multimodal continuous stirred tank reactor (CSTR)

The optimal control of a non-linear CSTR with multiple solutions has been studied by a number of researchers [22, 23]. The CSTR model in Mayer form is given by three differential equations:

$$\dot{x}_1 = -(2+u)(x_1+0.25)+(x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right) \tag{42}$$

$$\dot{x}_2 = 0.5 - x_2 - (x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right) \tag{43}$$

$$\dot{x}_3 = x_1^2 + x_2^2 + 0.1u^2 \tag{44}$$

with the initial conditions $\mathbf{x}(0) = [0.09, 0.09, 0.0]^T$. The final time t_f is 0.78. The performance index to be minimised is defined as

$$J = x_3(t_f) \tag{45}$$

Luus and Galli [22], and Luus [23] reported a local minimum of J=0.244425 and a global minimum of J=0.133094 using the Pontryagin's maximum principle. Using IDP Luus and Galli [22] found the values J=0.24452 and J=0.13336, respectively. To solve this problem by Evolutionary Algorithms, the control time grid was divided in the same number of control intervals as used by Luus and Galli [22], N=40 and a piecewise constant approximation for the control was selected. The chromosomes of the evolutionary algorithm were codified as real-valued vectors with values inside the range $-10 \le u(t) \le 10$. The population size used was 60 individuals and the number of generations was 3000, to obtain a value of the performance index J=0.13336.

Whatever the starting values for the parameterised control were, the Evolutionary Algorithm never converged to the local minima. The Evolutionary Algorithm was run seven times. The same value for the performance index was obtained using a population of 100 individuals and 3000 generations. The best solution obtained by an SQP-based algorithm with the same boundaries for the control input was J=0.144028 using a multistart strategy (ten optimisations).

The Nelder-Mead Simplex algorithm (*fminsearch.m* function of MATLAB's Optimization Toolbox), which does not use derivatives got the solution J=0.133366 as a multistart optimisation (ten optimisations) was done. However, both SQP and Nelder-Mead's algorithms were very sensitive to the initial guesses since they often converged to the solution around the value J=0.2444, the local minimum. The 30% of the times the Nelder-Mead algorithm converged to the local minima and this percentage was even greater in the case of the SQP algorithm (80%). The control functions obtained by all three algorithms are presented in figure 4. It is almost impossible to distinguish between the plots of the EA and Nelder-Mead solutions because they overlap. The control history obtained when the SQP algorithm converges to the local minimum is shown as well. As done by Luus and Galli [22] this problem was also solved with less time intervals (N=20) by the Evolutionary approach. In this case the performance index value was J=0.134155 using 40, 60 and 100 number of individuals and 3000 generations. The value obtained previously by Luus and Galli was J=0.13416.

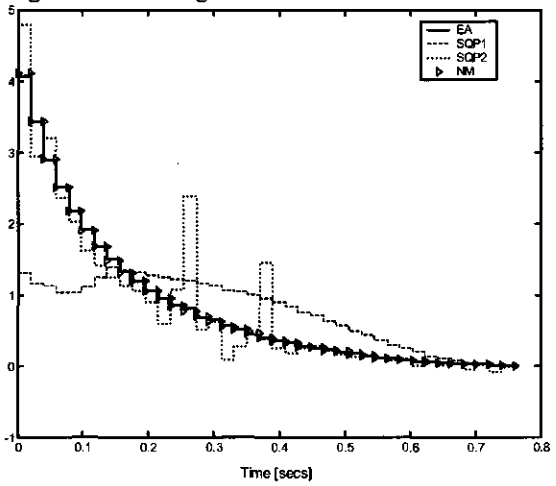


Figure 4. Optimal control of multimodal CSTR using EAs.

2.2.5.3 A non-differentiable system

The optimal control of a non-differentiable system has previously been analysed by several researchers [7, 23]. A system described by three differential equations is considered which a rectangular pulse signal is applied.

$$\dot{x}_1 = x_2 \tag{46}$$

$$\dot{x}_2 = -x_1 - x_2 + u + d \tag{47}$$

$$\dot{x}_3 = 5x_1^2 + 2.5x_2^2 + 0.5u^2 \tag{48}$$

where d=100[U(t-0.5)-U(t-0.6)] (49)

The final time considered is t_f=2.0 seconds and the initial conditions are:

$$x(0) = [0 \quad 0 \quad 0]^T \tag{50}$$

The control trajectory minimising the performance index

$$J = x_3(t_f) \tag{51}$$

has to be found.

To use the evolutionary approach for this problem, the time interval was divided in the same number of subintervals used before by Luus [23], N=40, and a piecewise constant control was used to approximate the control input. Because the control input does not have constraints we have to specify some boundaries for the variables of interest. The chromosomes were codified as real numbers inside the range $-20 \le u(t) \le 20$. The number of individuals for the evolutionary algorithm was 60, and the number of generations 1000 to obtain a value of the performance index of J=58.0927, which is less than those values reported previously (J=58.20) by Luus [23] using Iterative Dynamic Programming and Banga and Seider [7] (J=58.13) using their stochastic optimisation method.

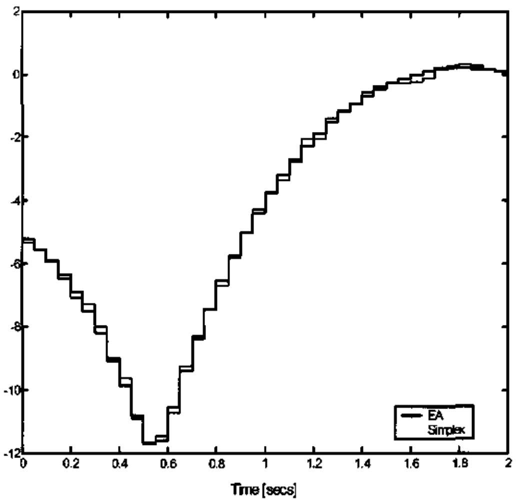


Figure 5. Optimal control of a non-differentiable system.

Moreover, a performance value of J=58.1194 is obtained with only 500 generations and a population size of 60 individuals. As expected, the SQP-based algorithms failed completely to solve this problem. The Nelder-Mead's simplex algorithm (function *fminsearch.m* of MATLAB's Optimisation Toolbox) which does not calculate derivatives was used as well and the best value obtained from several optimisations (four) was J=58.1057. The calculated control functions, which are very similar in shape to the ones calculated by Luus and Banga, are presented in figure 5. It is worthwhile to say that even in the case of a lower number of intervals (N=20) the Evolutionary Algorithm is able to converge to a very good solution (J=58.1050) but requiring in this case a population size of 60 individuals and 2000 generations.

2.2.6 Conclusions

The potential advantages of the use of Evolutionary approach to solve a class of optimal control problems were investigated by solving a high dimensional non-linear, a multi-modal non-linear and a non-differentiable, dynamic optimisation problem from chemical engineering literature. The results let us conclude that this approach is feasible to solve this class of optimal control problems. Moreover, Evolutionary

Algorithms seem to be able to always approximate the global optimum where other methods have difficulties in doing so. Nevertheless, further research is needed in order to investigate whether this approach is competitive computationally compared to other global optimisation methods used to solve this kind of problems. Future extensions of this work will be the solution of optimal control problems with final state constraints, problems with free final time and optimal control problems with general state constraints.

2.2.7 References

[1] O. von Stryk and R. Bulirsch, Direct and indirect methods for trajectory optimization, *Annals of Operations Research* 37, 1992, 357-373.

[2] S.K. Agrawal and B.C. Fabien, *Optimization of Dynamic Systems, Solid Mechanics and its applications*, (Dordrecht, The Netherlands, Kluwer Academic Publishers, 1999).

[3] A.E. Bryson Jr., *Dynamic Optimization*, (Menlo Park California: Addison-Wesley Longman Inc., 1999).

[4] C.J. Goh and K.L. Teo, Control Parametrization: a unified approach to optimal control problems with general constraints, *Automatica* Vol. 24, No. 1, 1988, 3-18.

[5] N.V. Dakev, A.J. Chipperfield, J.F. Whidborne, P.J. Flemming, An evolutionary algorithm approach for solving optimal control problems, *13th Triennial IFAC World Congress*, San Francisco, USA, 1996, 321-326.

[6] R. Luus, Application of dynamic programming to high-dimensional non-linear optimal control problems, *International Journal of Control,* 1990, 52, No. 1, 239-250.

[7] J.R. Banga and W.D. Seider, Global optimization of chemical processes using stochastic algorithms, in *State of the Art in Global Optimization,* Floudas C.A. and Pardalos PM (Eds), 563-583 (Dordrecht: Kluwer Academic Publishers, 1996).

[8] Q.T. Pham, Dynamic optimization of chemical engineering process by an evolutionary method, *Computers and Chemical Engineering*, Vol. 22, No. 7-8, , 1998, 1087-1097.

[9] F.L. Lewis and V.L. Syrmos, *Optimal Control*, (New York: John Wiley & Sons, Inc., 1995).

[10] T. Bäck, *Evolutionary Algorithms in Theory and Practice, Evolution Strategies, Evolutionary Programming, Genetic Algorithms* (New York: Oxford University Press, 1996).

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures=Evolution Programs*, (New York: Springer-Verlag , 1996).

[12] G. Rudolph, *Convergence Properties of Evolutionary Algorithms*, (Hamburg: Verlag Dr. Kovač, 1997).

[13] D.E. Goldberg, *Genetic Algorithms in search, optimization, and machine learning*, (Menlo Park, California: Addison-Wesley Publishing, 1989).

[14] Schwefel H.P. *Evolution and Optimum Seeking* (New York: John Wiley & Sons Inc, 1995).

[15] D.B. Fogel, *Evolutionary Computation: toward a new philosophy of machine learning* (Piscataway NJ: IEEE Press, 1995).

[16] M. Mitchell, *An introduction to genetic algorithms* (Cambridge, Massachusetts: The MIT Press, 1997).

[17] M.H. Lee, Ch. Han, K.S. Chang, Hierarchical time-optimal control of a continuous co-polymerization reactor during start-up or grade change operation using genetic algorithms, *Computers & Chemical Engineering*, Vol. 21, suppl., 1997, S1037-S1042.

[18] B. Andrés-Toro, J.M. Girón-Sierra, J.A. López-Orozco, J. Alvarez-Ruiz, P. Fernández-Blanco, A genetic optimization method for dynamic processes, *14th Triennial World Congress IFAC*, Beijing P.R. China, 1999, 373-378.

[19] J.A. Roubos G. van Straten, A.J.B. van Boxtel, An evolutionary strategy for fed-batch bioreactor optimization: concepts and performance, *Journal of Biotechnology* 67, 1999, 173-187.

[20] C. Houck, J.A. Joines, M.G. Kay, A genetic algorithm to function optimization: A MATLAB implementation, NCSU-IE TR 95-09, 1995.

[21] J. Joines and C.Houck, On the use of non-stationary penalty functions to solve constrained optimization problems with genetic algorithms. *1994 IEEE International Symposium Evolutionary Computation*, Orlando, FL, 579-584.

[22] R. Luus and M. Galli, Multiplicity of solutions in using dynamic programming for optimal control, *Hungarian Journal of Industrial Chemistry* 19, 1991, 55-61.

[23] R. Luus, Piecewise Linear Continuous Optimal Control by using Iterative Dynamic Programming, *Industrial and Engineering Chemistry Research* 32, 1993, 859-865.

PART II

EFFICIENT DIFFERENTIAL EVOLUTION ALGORITHMS IN OPTIMAL CONTROL

3 Efficient Differential Evolution algorithms for multimodal optimal control problems[†]

3.1 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 on the selection of the algorithm parameter values that steer the optimisation process when DE as 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 parameters that are easily determined such that multimodal optimal control problems are solved effectively and efficiently

KEY WORDS: Evolutionary Algorithms, Differential Evolution, Optimal Control, Optimization, Multimodal

3.2 Introduction

Indirect numerical methods for optimal control based on Pontryagin's Minimum Principle 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 parameterise 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.

In this study, Evolutionary Algorithms (EAs) are used to solve two optimal control problems that are known to have several local minima. Firstly, a First Order gradient

[†] A slightly modified version was submitted to the Journal Applied Soft Computing by I.L. Lopez Cruz, L. G. Van Willigenburg and G. Van Straten

algorithm from optimal control 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 several evolutionary algorithms is compared with that of a direct global method known as Iterative Dynamic Programming, which in the literature is reported as a very reliable method for the location of the global optimum in optimal control problems. It is well known that many Evolutionary 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 [1, 2] 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. Previous work on the application of some DE to solve optimal control problems can be found in references [3, 4, 5]. To our best knowledge there are no previous studies on the performance of DE to solve multimodal optimal control problems.

The paper is organised as follows: in section two a general description of the class of optimal control problems we are interested in is given. In section three 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 four 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 five 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.

3.3 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:

$$\dot{x} = f(x(t),u(t),t) \qquad (1)$$

where $x(t) \in \Re^n$ is the state and $u(t) \in \Re^m$ is the control. The control inputs are constrained,

$$\alpha_i(t) \le u_i(t) \le \beta_i(t)\,,\ i = 1,2,..,m\,, \qquad (2)$$

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

$$x(0) = x_0\,, \qquad (3)$$

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

$$J(u(t)) = \phi(x(t_f),t_f) + \int_0^{t_f} L(x(t),u(t),t)dt, \tag{4}$$

is minimised where the final time t_f is fixed [6]. There are two general approaches to solve these problems: indirect and direct methods [7]. The first group is based on the solution of a calculus of variations problem through the use of the Pontryagin's minimum principle (PMP) [8].

In using an indirect approach the necessary conditions for a stationary solution are derived adding the dynamic equations (1) to the performance index (4) by using a time-varying Lagrange multiplier vector $\lambda(t) \in R^n$

$$J' = \phi(x(t_f),t_f) + \int_0^{t_f} \{L(x(t),u(t),t) + \lambda^T(t) f(x(t),u(t),t) - \lambda^T(t)\dot{x}\}dt \tag{5}$$

Integrating by parts the last term below the integral and defining the Hamiltonian function as:

$$H(x(t),u(t),\lambda(t),t) = L(x(t),u(t),t) + \lambda^T(t) f(x(t),u(t),t) \tag{6}$$

the extended performance index is obtained:

$$J' = \phi(x(t_f,t_f)) - \lambda^T(t_f)x(t_f) + \lambda^T(0)x(0) + \int_0^{t_f} \{H(x(t),u(t),\lambda(t),t) + \dot{\lambda}^T(t)x(t)\}dt \tag{7}$$

An infinitesimal variation $\delta u(t)$ will produce the variations $\delta x(t)$ and $\delta J'$ in the states and the performance index respectively:

$$\delta J' = [\frac{\partial \phi}{\partial x}(t_f) - \lambda^T(t_f)\delta x] + [\lambda^T(0)\delta x] + \int_0^{t_f} [(\frac{\partial H(x,u,\lambda,t)}{\partial x} + \dot{\lambda}^T(t))\delta x + \frac{\partial H}{\partial u}\delta u(t)]dt \tag{8}$$

Hence, by choosing the multiplier functions $\lambda(t)$ in such a way that the coefficients of the state's variations vanish from (8) we obtained the Euler-Lagrange equations in the Calculus of variations [8].

$$\dot{\lambda} = -\frac{\partial H(x,u,\lambda,t)}{\partial x} = -\frac{\partial L(x(t),u(t),t)}{\partial x} - \lambda^T(t)\frac{\partial f(x(t),u(t),t)}{\partial x} \tag{9}$$

$$\lambda^T(t_f) = \frac{\partial \phi(t_f)}{\partial x(t_f)} \tag{10}$$

$$\frac{\partial H(x,u,\lambda,t)}{\partial u} = \frac{\partial L(x(t),u(t),t)}{\partial u} + \lambda^T(t)\frac{\partial f(x(t),u(t),t)}{\partial u} = 0,\ 0 \le t \le t_f \tag{11}$$

If the controls are bounded according to Pontryagin's minimum principle, equation (8) becomes:

$$H(x^*(t),u^*(t),\lambda^*(t),t) \le H(x^*(t),u(t),\lambda^*(t),t) \tag{12}$$

for all admissible $u(t)$. Hence, to find a control vector function $u(t)$ that minimises the performance index (4) a two-point boundary value problem with split boundary conditions has to be solved.

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 [9, 10]. This is achieved through control parameterisation. In our case the control $u(t)$ is assumed to be piecewise constant

$u(t_k) = u_k,\ t \in [t_k, t_{k+1}],\ k = 0,1,\ldots,N-1,\ t_0 = 0,\ t_N = t_f$ (13)

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 $\tilde{u} \in R^{m \times N}$ defined by $\tilde{u} = [u_0^T, .u_1^T, \ldots, u_{N-1}^T] = [\tilde{u}_1, \ldots, \tilde{u}_{m \times N}]$, where $\tilde{u}_i$, $i = 1,2,\ldots,m \times N$ are scalar parameters that minimise the cost function (4).

3.4 Two classes of Evolutionary Algorithms: Breeder Genetic Algorithms and Differential Evolution

Following Bäck [11] a generic description of an Evolutionary Algorithm is presented in figure 1. An Evolutionary Algorithm is a stochastic search method, which maintains a population $P(g) := \{\vec{a}_1(g), \ldots, \vec{a}_\mu(g)\}$ of individuals (chromosomes) $\vec{a}_i \in I; i = 1,\ldots,\mu$, at generation g, where I is a space of individuals, μ is the parent population size. Each individual represents a potential solution of the problem and it 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.

Figure 1. Procedure Evolutionary Algorithm

```
$g := 0;$
Generate $P(0) := \{\vec{a}_1(0), \ldots, \vec{a}_\mu(0)\} \in I^\mu$;
Evaluate $P(0) : \{\Phi(\vec{a}_1(0)), \ldots, \Phi(\vec{a}_\mu(0))\}$;
While $(\iota(P(g)) \neq \text{true})$ do
        Recombine $P'(g) := r\Theta_r(P(g))$;
        Mutate $P''(g) := m\Theta_m(P'(g))$;
        Evaluate
        $P''(g) : \{\Phi(\vec{a}_1''(g)), \ldots, \Phi(\vec{a}_\lambda''(g))\}$;
        Select $P(g+1) := s\Theta_s(P''(g) \cup Q)$;
        $g := g+1;$
End
```

Each individual is evaluated by a fitness function $\Phi : I \to R$, and 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 [12] is applied to the population in order to generate new individuals. From a set $\{w\Theta_1, \ldots, w\Theta_z \mid w\Theta_i : I^\lambda \to I^\lambda\} \cup \{w\Theta_0 : I^\mu \to I^\lambda\}$ of probabilistic evolutionary $w\Theta_i$ operators (for instance: crossover, mutation), each one specified by parameters given in the sets $\Theta_i \subset \Re$, some operators are applied to the population and a new evaluation of its fitness is calculated. The main evolutionary operators applied to the population $P(g)$ are: recombination (crossover) $r\Theta_r : I^\mu \to I^\lambda$ and mutation $m\Theta_m : I^\lambda \to I^\lambda$ are. A selection operator $s\Theta_s : \left(I^\lambda \cup I^{\mu+\lambda}\right) \to I^\mu$ which may modify the number of individuals from λ or $\lambda + \mu$ to μ, is applied as well, where $\lambda,\ \mu \in \mathrm{N}$, and μ denotes the number of parent individuals and λ the number of offspring. As before, the selection operator may be governed by a set of parameters defined in Θ_s. And $Q \subset P(g)$ denotes an additional set of individuals. The function

$\iota : I^{\mu} \rightarrow \{true, false\}$ represents the termination criterion for the evolutionary algorithm. After a number of generations it is expected that the best individual of the population represent a near-optimum solution.

3.4.1 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 [13]. 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: $\vec{a} = \vec{x} = (x_1,...,x_d) \in R^d$. Each potential solution in the evolutionary framework consists of the vector $\widetilde{u} = [\widetilde{u}_1, \widetilde{u}_2, ..., \widetilde{u}_{m \times N}]$ 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: $\vec{a} = \widetilde{u}$, so $d = m \times N$.

The genetic operators that have been chosen to make up the Evolutionary Algorithm are (i) cross-over by Discrete Recombination, (ii) mutation by the operator of the Breeder Genetic Algorithm [13], 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 [13], the discrete recombination $r_d : I^2 \rightarrow I$ (crossover operator) is defined as follows: let $\vec{a}_1 = (a_1^1,...,a_d^1)$ and $\vec{a}_2 = (a_1^2,...,a_d^2)$ be two parent chromosomes. Then each element of the offspring $\vec{a}_3 = (a_1^3,...,a_d^3)$ is computed by

$$a_i^3 = \begin{cases} a_i^1 & if \quad rand() < 0.5 \\ a_i^2 & otherwise \end{cases}, \; i = 1,...,d \tag{14}$$

where *rand()* is a uniform random number from [0,1]. This operator is applied μ times by picking up parents randomly in order to create an offspring population.

The mutation operator of the Breeder Genetic Algorithm $m_{\{p_m, r_s\}} : I \rightarrow I$ is defined as follows: let $\vec{a} = (a_1,...,a_d)$ be a parent solution. Then, for a given probability of mutation $p_m \in [0,1]$ and a specified 'shrinking mutation range' $r_s \in [0,1]$, a gene (variable) a_i' is selected and modified to generate a new variable according to:

$$a_i' = \begin{cases} a_i + m_r \cdot range_i \cdot \delta & if \quad rand() < 0.5 \\ a_i - m_r \cdot range_i \cdot \delta & otherwise \end{cases}, i = 1,...,d \tag{15}$$

where $range_i = \frac{1}{2} r_s (\beta_i - \alpha_i)$, $m_r = \begin{cases} 1 & if \quad rand() < p_m \\ 0 & otherwise \end{cases}$, $\delta = \sum_{j=0}^{19} \gamma_j 2^{-j}$,

rand() is a uniform random number from [0,1], $\gamma_j \in [0,1]$ with probability 0.05, $p_m = 1/d$ normally, and r_s are algorithm parameters that have to be specified, α_i and β_i denote the lower and upper boundaries of the variable a_i. With the given settings for δ the mutation operator is able to locate the optimum up to a precision of $range_i \cdot r_s \cdot 2^{-19}$.

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

$$\Phi(\vec{a}_i) = 2 - s_p + 2(s_p - 1)\frac{f'(\vec{a}_i) - 1}{\mu - 1}; i = 1,\ldots,\mu \qquad (16)$$

with the selection pressure coefficient $s_p = 2.0$ and $f'(\vec{a}_i)$ 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 $\vec{a}_i$ being picked equals $p_s(\vec{a}_i)$. $p_s(\vec{a}_i)$ are calculated according to:

$$p_s(\vec{a}_i) = \frac{\Phi(\vec{a}_i)}{s_\Phi} \quad , s_\Phi = \sum_{j=1}^{\mu} \Phi(\vec{a}_j) \qquad (17)$$

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

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 that shows sub-populations help evolutionary algorithms to locate the global optimum [14]. The computer implementation of these Evolutionary Algorithms is given in the Genetic Algorithm Toolbox for use with MATLAB [15]. The integration of the dynamic equations was implemented by using a C-MEX file routine in order to speed up the simulations.

3.4.2 Differential Evolution Algorithms

$$a'_{ji}(g)=\begin{cases}\beta_j & if\ a'_{ji}(g)>\beta_j\\ \alpha_j & if\ a'_{ji}(g)<\alpha_j\end{cases}\quad j=1,2,...,d; i=1,2,...,\mu \tag{25}$$

where α and β 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 computes a Simulink model programmed as a C-MEX s-function, which contains the dynamic equation of the system and the objective function.

3.5 The first order gradient algorithm and the Iterative Dynamic Programming algorithm

3.5.1 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 [8]. However, a line search procedure was introduced in order to calculate the value of the step size parameter (k), which in Bryson's algorithm [8] is constant. The gradient algorithm is described next and applies to a Mayer formulation of the optimal control problem [8].

i) Guess $u(t)$ at N+1 points $t-t_0=0,...,N\Delta T$, $\Delta T=(t_f-t_0)/N$. N is an even number.

ii) Integrate the state equations forward. Store $x(t)$ at $t-t_0=\Delta T,...,N\Delta T$.

iii) Evaluate $\phi[x(t_f)]$ and $\lambda^T(t_f)=\frac{\partial\phi}{\partial x}(t_f)$.

iv) Compute and store $\lambda(t)$ and the function $\frac{\partial H(x,u,\lambda,t)}{\partial u}$ at $t-t_0=\Delta T,...,N\Delta T$, by integrating backward in time $\dot{\lambda}=-\lambda^T\frac{\partial f(x(t),u(t),t)}{\partial x}$, starting at $\lambda(t_f)$, where $\frac{\partial H(x,u,\lambda,t)}{\partial u}=\lambda^T(t)\frac{\partial f(x(t),u(t),t)}{\partial u}$.

v) Apply a line search algorithm to determine the step size parameter (k).

vi) Compute $\delta u(t)$ and the new $u(t)$ according to:

$$\delta u(t)=-k\frac{\partial H^T}{\partial u}(t),\ u(t):=u(t)+\delta u(t)$$

vii) Clip the controls if necessary in accordance with:

$$u(t)=\begin{cases}\alpha(t)\ if\ u(t)<\alpha(t)\\ \beta(t)\ if\ u(t)>\beta(t)\end{cases}$$

viii) If $|\delta u_{avg}|<\varepsilon_g$ stop. Otherwise go to step ii).

where $\varepsilon_g>0$ is a desired precision and $\delta u_{avg}=\sqrt{\frac{1}{t_f}\int_0^{t_f}\delta u^T(t)\delta u(t)dt}$

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

3.5.2 Iterative Dynamic Programming algorithm

An iterative version of the Dynamic Programming Algorithm has been proposed by Luus [21] 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 [21].

Step 0. Initialisation. The time interval $[0, t_f]$ is divided into N time intervals, each of length L. The control is approximated by the piecewise constant control policy $u(t) = u(t_k) \in [t_k, t_{k+1}),\ k = 0,...,N-1,\ t_{k+1} - t_k = L,\ t_0 = 0,\ t_N = t_f$

Choose $u_0(0),...,u_0(N-1)$ and $r_0(0),...,r_0(N-1)$ where $r_0(k)$, $k = 0,1,...,N-1$ specifies the range $u_0(k) \pm r_0(k)$ of allowable values of control for the next iteration $u_1(k)$, $k = 0,...,N-1$. Select the number of allowable values for control $R > 1$ to be tried at each stage $k = 0,...,N-1$. Choose the region contraction factor $0.5 \le \gamma < 1.0$ 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) $u^*_{i-1}(0),...,u^*_{i-1}(N-1)$, and generate $M-1$ other control policies within the region $u^*_{i-1}(k) \pm r_{i-1}(k)$, $k = 0,...,N-1$. Integrate the dynamic equation (eqn 1) from $t = 0$ to t_f for all M control policies. The M values of $x_m(k)$, $k = 0,...,N-1$, $m = 1,2,...,M$ at the beginning of each time stage are stored.

Step 3. a) At stage N, for each of the M stored values for $\mathrm{x}_m(N-1)$, integrate the dynamic equation (eq. 1) from $t_f - L$ to t_f, with each of the R allowable values for the control, which are generated by:

$$\mathrm{u}_i(N-1) = \mathrm{u}^*_{i-1}(N-1) + \mathrm{D} \cdot \mathrm{r}_{i-1}(N-1) \qquad (26)$$

where $\mathrm{u}^*_{i-1}(N-1)$ 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:

$$u_i(N-1) = \begin{cases} \alpha(t)\ if\ u_i(N-1) < \alpha(t) \\ \beta(t)\ if\ u_i(N-1) > \beta(t) \end{cases} \qquad (27)$$

b) From the R values of the control choose the one $\mathrm{u}^*_i(N-1)$ 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 $t_f - L$ to t_f using the stored value for $\mathrm{u}^*_i(N-1)$ corresponding to the state grid point closest to the value of the calculated state vector at time $t_f - L$. From the R values of the control select $\mathrm{u}^*_i(N-2)$ 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 (eqn. 3). Store the trajectories $u_i^*(k)$, $x^*(k)$, $k = 0,1,...,N-1$.

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

$$r_{i+1}(k) = \gamma \cdot r_i(k), k = 0,1,...,N-1 \qquad (28)$$

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 consists in to restore the value of the region contraction factor γ 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 [21]. In that case another parameter called region restoration factor $0.5 \le \eta \le 1.0$ 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 (γ), number of allowable values for control (R), number of grid points (N), initial region size values ($r_0(k)$), and restoration factor (η) 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 [21] that with too small values of the region contraction factor (γ) premature collapse of the region $r(k)$, $k = 0,1,...,N-1$ 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 γ work properly with sufficiently large values of the allowable values for control (R). Conversely, when small values of R are used, high values of γ 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 s-function to speed up the simulation of the dynamic equations.

3.6 Benchmark problems solved and results

3.6.1 The optimal control of a non-linear stirred tank reactor

A multimodal optimal control problem has been used by Luus [21, 22] to evaluate his Iterative Dynamic Programming algorithm. Ali et al. [24] 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 [25]. 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.

$$\dot{x}_1 = -(2+u)(x_1+0.25)+(x_2+0.5)\exp(\frac{25x_1}{x_1+2}) \tag{29}$$

$$\dot{x}_2 = 0.5 - x_2 - (x_2+0.5)\exp(\frac{25x_1}{x_1+2}) \tag{30}$$

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:

$$J = \int_0^{t_f} (x_1^2 + x_2^2 + 0.1u^2)dt \tag{31}$$

where $t_f = 0.78$. The initial conditions are $\mathbf{x}(0) = [0.09 \quad 0.09]^T$. 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 1e-8. The initial guesses for the controls of the different algorithms were selected from the interval $0 \le u(t) \le 5.0$.

For the solution of this problem by the gradient method the step size parameter (*k=0.12*) was kept 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 [8] 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. As can be seen from table 1, 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, $u_0(t) = c$, $0 \le t \le t_f$ the gradient algorithm always converged to the local optimum ($J^* = 0.2444$) if $u_0(t) \le 1.8$; otherwise it converges to the global optimum ($J^* = 0.1330$). The associated control trajectories have completely different shapes.

Table 1. Optimal Control of a multimodal CSTR by a first order gradient algorithm (constant u_0 (t))

$u^0(t)$	J*	Iterations	CPU time
5.0	0.1330984	244	411.03
4.0	0.1330977	243	406.18
3.0	0.1330979	228	366.16
2.5	0.1330976	224	398.11
2.0	0.1330980	211	351.84
1.8	0.2444349	329	755.95
1.0	0.2444351	215	483.69
0.5	0.2444347	228	509.63
0.25	0.2444346	241	660.69
0.0	0.2444346	249	657.46

secs. Measured on a Pentium III 700 MHZ PC

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 [18, 19]. 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 [21] 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. $u_0(t_k) = c$, $k = 0,1,\dots,N-1$ with c a value randomly taken from the control interval $0 \le u(t) \le 5$. A similar procedure was followed for the selection of the initial region value $r_0(t_k)$, $k = 0,1,\dots,N-1$ which was selected from the interval $0 \le r_0(t_k) \le 5$. Since the control has no constrains equation 16 was not used.

Table 2 shows the results obtained by the IDP algorithm. From the values presented in the first four columns it is evident that IDP still can convergence to the local optimum ($J^* = 0.2446$), which in this case occurs, when both the initial value for the control and the initial region size are too small. Otherwise IDP converges to the global optimum ($J^* = 0.1356$). However, Luus and Bojkov [23] 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 $r_0(t_k) \ge 4$, convergence to the global optimum is always obtained regardless the initial value for control $u_0(t_k)$. This is shown in table 2 columns five to seven. Repeated optimisation is necessary since the IDP algorithm (see section 4.2.) generates randomly the allowable values for control (eqn. 15). So it is very likely may converge to a different value each run.

Table 2. Optimal control of a multimodal CSTR by Iterative Dynamic Programming

$u_0(t_k)$	$r_0(t_k)$	J*	CPU time#	$u_0(t_k), r_0(t_k)$=4	J*	CPU time#
3.4990	2.2999	0.1355869	600.79	1.0000	0.1355852	700.50
2.3744	3.1557	0.1356138	620.46	4.2823	0.1356766	686.30
0.9718	4.1646	0.1355876	600.16	2.0389	0.1356422	668.74
1.1568	1.2498	0.2446122	589.43	3.9007	0.1356262	665.34
3.8732	4.4433	0.1356089	590.77	0.7900	0.1355806	764.05
4.0504	3.4722	0.1355970	635.90	0.0108	0.1355933	701.57
0.7557	1.2997	0.2446122	599.58	2.3551	0.1355905	872.46
4.7105	3.7785	0.1355856	602.31	2.7851	0.1355866	849.92
4.6422	2.2919	0.1355816	599.64	1.0293	0.1356085	851.20
4.0494	0.8660	0.1355828	588.51	2.9137	0.1355811	683.33
				Mean	0.1356070	744.34

secs. Measured on a Pentium III 700 MHZ PC, Function Evaluations=2100.

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 the degree at which solutions in the population are close together seems to be a good criterion [26]. 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 (x), control (u), and also J are dimensionless in this problem it is a good option to select an absolute convergence criterion. This was defined as follows:

let J_b be the best objective function value in the population $J_b = \min J(\tilde{u}_i)$, $i = 1,...,\mu$, and J_w the worst function value $J_w = \max J(\tilde{u}_i)$, $i = 1,...,\mu$, then an absolute convergence criterion can be defined by $J_w - J_b < \varepsilon_c$. In the current application ε_c was selected to be $\varepsilon_c = 0.00001$, which guarantees good accuracy of the solution. Since this optimal control problem is unconstrained equation 14 was not used.

Seven variants of evolutionary algorithms were implemented and evaluated. Two of them are based on the Breeder Genetic Algorithm and five are Differential Evolution algorithms. Tables 3 to 5 present the main results for various values of the population size μ. The results reported in the tables are averages from 10 runs. EA_1 means the Breeder Genetic Algorithm with only one population, EA_2 denotes the Breeder Genetic Algorithm with sub-populations, EA_3 denotes the DE algorithm ***DE/rand/1/bin*** and EA_4 stands for the DE algorithm ***DE/best/1/bin***. EA_5 denotes the DE algorithm ***DE/best/2/bin***. EA_6 means the ***DE/rand/1/exp***, and **EA_7** denotes the ***DE/trial/2,best, trial/bin***. These algorithms are evaluated on the basis of four criteria: i) the number of function evaluations (F.E.), where each evaluation involves the integration of the dynamic equations (eq. 1) from 0 to t_f, 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 percentual number of times that the algorithm found the global solution.

A parameter tuning approach was applied in order to determine what combination of algorithm parameters gives the best performance index with the less number of function evaluations. Storn and Price [1] have suggested values for the population size from the interval $5 \cdot d \leq \mu \leq 10 \cdot d$ for static optimisation problems, where d is the dimension of the problem. Price [2] proposed selecting the population size from the interval $2 \cdot d \leq \mu \leq 20 \cdot d$. 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 $d = m \cdot N$ 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 less number of function evaluations. The values reported in the tables are the ultimate values obtained this way.

Several remarks need to be made. Firstly, all EAs 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* (EA_5).

Table 3. Optimal Control of a multimodal CSTR by evolutionary algorithms (population size $\mu = 20$)

Algorithm	EA_1	EA_2	EA_3	EA_4	EA_5	EA_6	EA_7
FE	7401.80	7143.4	3494	2240	2270	3578	2492
CPU time#	430.02	451.10	242.51	177.06	160.44	277.35	194.92
J*	0.1358249	0.1355985	0.1355966	0.1356143	0.1355850	0.1355929	0.1355847
CE (%)	100	100	100	100	100	100	100
Iterations	410.1	397.3	174.7	112	113.5	177.9	124.6
Parameters	$p_m = 0.09$	$p_m = 0.09$ $m_r = 0.8$	CR=0.5	CR=0.5	CR=0.5	CR=0.15	CR=0.5
	g_{gap}=0.9	g_{gap}=0.9, subpop=2	F=0.4	F=0.5	F=0.4	F=0.5	F=0.6

F.E.=Function evaluations, C.E.=Convergence effectiveness. # On a Pentium III, 700 MHZ PC

Table 4. Optimal Control of a multimodal CSTR by evolutionary algorithms (population size $\mu = 15$)

Algorithm	EA_1	EA_2	EA_3	EA_4	EA_5	EA_6	EA_7
FE	6985.60	5272.6	3535.5	2100	1783.50	3018	2083.5
CPU time#	406.85	314.93	251.44	154.79	134.26	237.95	144.24
J*	0.1362200	0.1362479	0.1355920	0.1356205	0.1355970	0.1357880	0.1355867
EC (%)	100	100	100	100	100	100	100
Iterations	497.9	438.8	235.7	140	118.9	200.2	138.9
Parameters	$p_m = 0.09$	$p_m = 0.09$, $m_r = 0.8$	CR=0.5	CR=0.35	CR=0.5	CR=0.1	CR=0.6
	g_{gap}=0.9	g_{gap}=0.9, subpop=2	F=0.5	F=0.5	F=0.4	F=0.5	F=0.7

F.E.=Function evaluations, C.E.=Convergence effectiveness. # On a Pentium III, 700 MHZ PC

Table 5. Optimal Control of a multimodal CSTR by evolutionary algorithms (population size $\mu = 10$)

Algorithm	EA_1	EA_2	EA_3	EA_4	EA_5	EA_6	EA_7
FE	2225.8	7925	2097	2294	1719	2843	1918
CPU time#	135.57	449.50	200.89	157.14	134.80	206.74	130.23
J*	0.1449189	0.1365219	0.1356905	0.1356040	0.1356052	0.1367922	0.1355913
CE (%)	100	100	100	100	100	100	100
Iterations	246.20	792	290.7	229.40	171.9	283.3	191.8
Parameters	$p_m = 0.09$	$p_m = 0.09$, $m_r = 0.8$	CR=0.5	CR=0.25	CR=0.5	CR=0.1	CR=0.4
	g_{gap}=0.9	g_{gap}=0.9, subpop=2	F=0.6	F=0.6	F=0.5	F=0.62	F=0.7

F.E.=Function evaluations, C.E.= Convergence effectiveness. # On a Pentium III, 700 MHZ PC

In general the parameter settings are different among DE algorithms due to the different effect of the mutation operator. In both algorithms EA_1 and EA_2 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 give rise to a higher number of function evaluations. In case of EA_2 it was not possible to obtain a better solution by increasing the number of subpopulations by more than two. For EA_2 the migration rate (m_r) between populations was allowed each 20 generations.

That actually, Differential Evolution algorithms are efficient in solving this problem can clearly be demonstrated comparing our results against those reported previously by using Controlled Random Search methods for global optimisation [27, 28] which hinge, like Evolutionary Algorithms, on the manipulation of a population of potential solutions. Ali et al. [24] have reported that from a total of four evaluated controlled random search algorithms a modified controlled random search algorithm identified as CRS4 [27] obtained the best solution. The number of function evaluations required was 8997 to obtain a performance index value of $J^* = 0.136$ using the same number of number of time intervals (13) we have used. Clearly, this number of function evaluations is greater than the values reported in tables 3 to 5. Moreover, the Evolutionary Algorithm that performed worst on this problem (EA_6) is more efficient than CRS4. From our own MATLAB implementation of the controlled random search algorithm (CRS2) [27] the average from ten runs was a performance index value of $J^*=0.1384931$, with the corresponding function evaluations (10675) and 863.64 seconds of computation time. Again the number of function evaluations is greater that those values reported on tables 3 to 5.

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 $F = 0.5$, $CR = 0.1$ according to Storn and Price [1], and then if 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 $0.4 \le F \le 0.7$ were sufficient to obtain a C.E. of 100%. Greater values of CR resulted in solutions with worst (with less accuracy) performance but not necessarily to local minima. 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 $5*(m*N) \le \mu \le 10*(m*N)$ 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 of the evaluated algorithms the *DE/best/2/bin* was able to solve this problem with a smaller number of function evaluations[&]. This shows that DE algorithms are actually very efficient evolutionary algorithms. To find 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 than guarantee CE=100% are easily obtained for DE algorithms considered here. Therefore, taking into consideration the whole preliminary work that the application of the IDP algorithm demands it turns out deceptively inefficient.

[&] cf. table 2

3.6.2 The bifunctional catalyst blend optimal control problem

A very challenging multimodal optimal control problem has been posed by Luus [21, 23]. This problem is also proposed as benchmark in the Handbook of Test Problems in Local and Global Optimization [25]. A chemical process converting methylcyclopentane to benzene in a tubular reactor is modelled by a set of seven differential equations:

$$\dot{x}_1 = -k_1 x_1 \tag{32}$$
$$\dot{x}_2 = k_1 x_1 - (k_2 + k_3) x_2 + k_4 x_5 \tag{33}$$
$$\dot{x}_3 = k_2 x_2 \tag{34}$$
$$\dot{x}_4 = -k_6 x_4 + k_5 x_5 \tag{35}$$
$$\dot{x}_5 = k_3 x_2 + k_6 x_4 - (k_4 + k_5 + k_8 + k_9) x_5 + k_7 x_6 + k_{10} x_7 \tag{36}$$
$$\dot{x}_6 = k_8 x_5 - k_7 x_6 \tag{37}$$
$$\dot{x}_7 = k_9 x_5 - k_{10} x_7 \tag{38}$$

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

$$k_i = c_{i1} + c_{i2}u + c_{i3}u^2 + c_{i4}u^3,\ i = 1,...,10. \tag{39}$$

The values of the coefficients c_{ij} are given in [21]. The upper and lower bounds on the mass fraction of the hydrogenation catalyst are: $0.6 \le u(t) \le 0.9$, and the initial vector of mole fraction is $\mathbf{x}[0] = [1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^T$. This is a continuous process operated in steady state, so that 'time' in equations (32)-(39) 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 $0 \le t \le t_f$ where the final effective residence time $t_f = 2000 g \cdot h / mol$ such that the concentration in the reactor is maximised: $J = x_7(t_f) \times 10^3$. Luus [21] has shown that this problem has a lot of local optima (25).

Table 6. Optimal control of bifunctional catalyst blend by a first order gradient algorithm (N=10)

Constant $u^0(t)$				Random $u^0(t)$		
$u^0(t)$	J*	iterations	CPU time#	J*	iterations	CPU time#
0.80	9.6419	16	166.66	8.7627	22	2052.40
0.70	8.1215	20	3689.00	8.0054	37	6811.80
0.85	9.6419	8	97.86	8.4409	24	5450.10
0.65	8.1214	23	4593.30	8.1691	46	6076.10
0.90	9.6419	6	36.03	8.5083	30	6235.30
0.79	**9.7577**	38	494.68	8.4300	21	1931.10
0.60	8.1214	27	6928.60	**9.3883**	17	1344.30
0.72	8.1223	49	9072.10	8.2718	22	3860.80
0.78	9.6574	31	3355.40	9.1816	38	3396.10
0.82	9.6419	9	131.05	9.0628	89	13156.00

# On a Pentium III, 700 MHZ PC

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 $\varepsilon_g = 0.000001$. Despite both enhancements the classical method failed to locate the global optimum as can be seen in table 6 that shows the results of twenty optimisations. The best solutions (emphasized in table 6) 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 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 optimisation 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 exactly seems a good approach in solving multi-modal optimal control problems.

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 $u_0(t_k)$; $k = 0,\dots,N-1$ was chosen constant with a value randomly chosen from the control interval $0.6 \le u(t) \le 0.9$. A similar procedure was followed to select the initial region value $r_0(t_k)$, $k = 0,\dots,N-1$ which was chosen, from the interval $0.6 \le r_0(t_k) \le 0.9$.

Table 7. Optimal control of bifunctional catalyst by Iterative Dynamic Programming

$u_0(t_k)$	$r_0(t_k)$	FE	J*	Iterations	CPU time#
0.6658	0.7814	1500	10.0942	20	174.36
0.8727	0.6677	1350	10.0942	18	165.58
0.6353	0.6330	1125	10.0942	15	126.59
0.8574	0.8600	600	10.0942	8	72.41
0.7537	0.7033	1125	10.0942	15	121.19
0.6842	0.7090	1500	10.0942	20	137.52
0.6357	0.6099	825	10.0942	11	92.58
0.8515	0.7711	1425	10.0942	19	152.20
0.8035	0.8417	1575	10.0942	21	169.65
0.7915	0.6154	1050	10.0942	14	138.23
Mean		1207.5	10.0942	16.1	135.03

FE= function evaluations. # On a Pentium III, 700 MHZ PC

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 [21]. The maximum number of iterations was ($I = 30$), but the optimisation was stopped when it reached the condition $J^* - J \le 0.00005$. Table 7 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 8a shows that with a value of $r_0(t_k) = 0.27$ the IDP algorithm 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%.

Table 8.a. Optimal control of bifunctional catalyst by Iterative Dynamic Programming

$r_0(t_k) = 0.27$					$r_0(t_k) = 0.30$				
$u_0(t_k)$	FE	J*	iterations	CPU time#	$u_0(t_k)$	FE	J*	iterations	CPU time#
0.75	2250	10.0395	30	293.78	0.65	2250	10.0528	30	263.18
0.70	1125	10.0942	15	370.76	0.68	675	10.0942	9	88.81
0.80	2250	9.94429	30	295.76	0.70	450	10.0942	6	61.44
0.65	2250	10.0395	30	299.92	0.72	975	10.0942	13	121.72
0.85	2250	10.0395	30	286.77	0.75	2250	10.0395	30	320.65
0.68	2250	10.0395	30	297.81	0.78	450	10.0942	6	57.42
0.72	1050	10.0942	14	145.40	0.80	2250	10.0395	30	306.41
0.78	2250	10.0395	30	300.68	0.82	2250	10.0395	30	328.19
0.82	2250	9.8978	30	321.38	0.85	900	10.0942	12	514.37
0.75	2250	10.0395	30	293.78	0.88	1275	10.0942	17	385.43

FE= function evaluations. # On a Pentium III, 700 MHZ PC

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 is shown in table 8b. Table 8b shows the average of ten optimisations with different random allowable values for control.

Table 8.b. optimal control of bifunctional catalyst by Iterative Dynamic Programming

$r_0(t_k)$	0.40	0.50
Function evaluations	938.70	1177.5
Iterations	13.70	15.70
CPU time (secs.)#	132.33	150.64
J*	10.0942	10.0942

FE= function evaluations. # On a Pentium III, 700 MHZ PC

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:

$$\frac{\mu}{\varepsilon_d}(J_w - J_b) \leq \left|\sum_{i=1}^{\mu} J(\tilde{u}_i)\right| \quad (40)$$

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 9 to 11 show the averages of 10 runs for 3 values of the population size ($\mu = 15, 20, 25$). Reported values of the performance (J^*) are averages over successful optimisations.

Table 9. Optimal control of the bifunctional catalyst blend problem by EAs (population size $\mu = 25$)

Algorithm	EA_1	EA_2	EA_3	EA_4	EA_5	EA_6	EA_7
FE	7007.10	4890	3172.50	2657.5	3607.5	2930	2897.5
CPU time#	1186.60	515.14	549.75	460.11	632.84	542.24	478.24
J*	10.0942	10.0929	10.0941	10.0942	10.0941	10.0941	10.0941
CE (%)	70	80	100	100	100	100	100
Iterations	278.28	202.50	126.9	106.3	144.3	116.2	115.9
Parameters	$g_{gap}=1$	$g_{gap}=1, m_r = 0.2$	F=0.9	F=0.9	F=0.9	F=0.9	F=1.0
	$p_m = 0.18$	$p_m = 0.2$, subpop=4	CR=0.0	CR=0.0	CR=0.0	CR=0.0	CR=0.0

F.E.=Function evaluations, C.E.= Convergence effectiveness, # On a Pentium III , 700 MHZ PC

Table 10. Optimal control of the bifunctional catalyst blend problem (population size $\mu = 20$)

Algorithm	EA_1	EA_2	EA_3	EA_4	EA_5	EA_6	EA_7
FE	11493	18227.5	2496	1900	2736	2260	2104.4
CPU time#	1896.90	3237.20	453.92	361.07	537.62	413.77	387.72
J*	10.0934	10.0916	10.0941	10.0941	10.0940	10.0940	10.0942
EC	60	80	100	100	100	100	90
Iterations	572.66	910.12	124.8	95	136.8	112	105.22
Parameters	$g_{gap}=1$	$g_{gap}=1, m_r = 0.2$	F=0.9	F=0.9	F=0.9	F=0.9	F=1.0
	$p_m = 0.28$	$p_m = 0.45$,subpop=4	CR=0.0	CR=0.0	CR=0.0	CR=0.0	CR=0.0

F.E.=Function evaluations, E.C.=efficiency of convergence. # On a Pentium III, 700 MHZ PC

Table 11. Optimal control of the bifunctional catalyst blend problem (population size $\mu = 15$)

Algorithm	EA_1	EA_2	EA_3	EA_4	EA_5	EA_6	EA_7
FE	6552.90	12718	1752	1644	2268	1803	1518.3
CPU time#	578.99	2433.40	341.28	297.98	385.44	355.07	273.79
J*	10.0937	10.0854	10.0940	10.0941	10.0938	10.0922	10.0928
EC	70	50	100	100	100	100	90
Iterations	434.85	793.60	116.8	109.6	151.2	119.2	101.22
Parameters	$g_{gap}=1$	$g_{gap}=1, m_r = 0.2$	F=0.9	F=0.9	F=1.0	F=0.9	F=1.0
	$p_m = 0.29$	$p_m = 0.56$,subpop=4	CR=0.0	CR=0.0	CR=0.0	CR=0.0	CR=0.0

F.E.=Function evaluations, E.C.=efficiency of convergence. # On a Pentium III, 700 MHZ PC

The algorithms EA_1 and EA_2 did not reach $CE = 100\%$ with small population sizes. However, it was found that by increasing adequately the population size, EA_2 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 $m_r = 0.2$, mutation rate $p_m = 0.1$ and an elapsed time of 10 generations between migrations, EA_2 converged always to a value J*=10.0942. The average required number of function evaluations was 7048.6. EA_1 converged only

80% of the times to the global optimum with population size $\mu = 40$ and $\mu = 50$. These results illustrate the benefits of using sub-populations. Anyway, the best result of EA_2 regarding the number of function evaluations is considerably inferior to those results obtained by DE algorithms. Except for EA_7 all the Differential Evolution algorithms always reached $CE = 100\%$ with the tested population sizes. Moreover, Differential Evolution algorithms were considerably more efficient than BGAs.

By comparing tables 9 through 11 it can be seen that both EA_1 and EA_2 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 algorithms require less function evaluations, and usually have better convergence to the global optimum. As for the number of function evaluations the best algorithm in this case is EA_4. But, the others are not significantly less efficient than EA_4. EA_7 required a greater value for F in order to improved convergence to the global optimum. However, when the population is too small it has some problems to avoid be trapped by local minima.

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 expenses 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, by comparing the efficiency achieve by EA_4 or EA_3 against that of IDP (tables 7, 8b and 11) the difference is relatively small. Moreover, one could argue that this difference is not significant 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.

3.7 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 compare the Breeder Genetic algorithm. In contrast to the majority of Evolutionary Algorithms, where many algorithm parameters have to be tuned, in DE only three parameter values (the population size, the crossover constant and the differential variation coefficient) are required to steer the search of the algorithm. 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 becomes higher. 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 Algorithm 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 [24]. 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 ($\mu \leq 2 * (N \times m)$) 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.

3.8 References

[1] R. Storn , K. Price, Differential Evolution- a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization 11*, 1997, 341-359.

[2] Price K. V. An Introduction to Differential Evolution in Corne D., Dorigo, M. and Glover F., New Ideas in Optimization, Mc GrawHill, 1999.

[3] F.S. Wang & J.P Chiou, Optimal control and optimal time location problems of differential-algebraic systems by differential evolution, *Industrial & Engineering Chemistry Research*, *36* (1997), 5348-5357.

[4] J.P. Chiou, F.S. Wang, Hybrid method of evolutionary algorithms for static and dynamic optimisation problems with applications to a fed-batch fermentation process, *Computers and Chemical Engineering 23* (1999), 1277-1291.

[5] M.H. Lee, Ch. Han, K. S. Chang, Dynamic optimisation of continuous polymer reactor using a modified differential evolution algorithm, *Ind. Eng. Chem. Res.* 38 (1999), 4825-4831.

[6] F.L. Lewis, V.L. Syrmos, *Optimal Control* (John Wiley & Sons, INC, NY:1995).

[7] O. von Stryk, R. Bulirsch, Direct and indirect methods for trajectory optimization, *Annals of Operations Research* 37, 1992, 357-373.

[8] A.E. Bryson, *Dynamic Optimization* (Addison Wesley, Menlo Park NY: 1999).

[9] S.K. Agrawal, B.C. Fabien, *Optimization of Dynamic Systems, Solid Mechanics and its applications*, (Dordrecht, The Netherlands, Kluwer Academic Publishers, 1999).

[10] C.J. Goh, K.L. Teo, Control Parametrization: a unified approach to optimal control problems with general constraints, *Automatica 24*(1), 1988, 3-18.

[11] T. Bäck, *Evolutionary Algorithms in Theory and Practice, Evolution Strategies, Evolutionary Programming, Genetic Algorithms* (New York: Oxford University Press, 1996).

[12] G. Rudolph, *Convergence Properties of Evolutionary Algorithms*, (Hamburg: Verlag Dr. Kovač, 1997).

[13] H. Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization, *Evolutionary Computation 1*(1), 1993, 25-49.

[14] E. Cantú-Paz, E. Designing Efficient and Accurate Parallel Genetic Algorithms, PhD thesis. University of Illinois at Urbana-Champaign, 1999.

[15] A. Chipperfield, P. Flemming, H. Pohlheim, C. Fonseca, *Genetic Algorithm Toolbox for use with MATLAB*, Department of Automatic Control and Systems Engineering, University of Sheffield, User's guide, version 1.2, 1994.

[16] R. Storn, On the usage of differential evolution for function optimisation, NAFIPS 1996, Berkeley, pp. 519 – 523.

[17] Storn, R. and Price, K., Minimizing the real functions of the ICEC'96 contest by Differential Evolution, IEEE Conference on Evolutionary Computation, Nagoya, 1996, pp. 842 – 844.

[18] Storn R., System Design by Constraint Adaptation and Differential Evolution, *IEEE Transactions on Evolutionary Computation 3*(1), 1999, 22-34.

[19] M.M. Fisher, K. Hlaváčková-Schindler, M. Reismann, A global search procedure for parameter estimation in neural spatial interaction modelling, *Papers in Regional Science 78*, 1999, 119-134.

[20] R. Storn, Differential Evolution design of an IIR-filter with requirements for magnitude and group delay, International Computer Science Institute, 1947 Center Street, Berkeley, Technical Report TR-95-026, ICSI, May 1995.

[21] R. Luus, *Iterative Dynamic Programming* (Boca Raton, FL: Chapman & Hall/CRC, 2000).

[22] R. Luus and M. Galli, Multiplicity of solutions in using dynamic programming for optimal control, *Hung. J. Ind. Chem* 19 (1991), 55-62.

[23] R. Luus and B. Bojkov, Global optimization of the bifunctional catalyst problem, Can. J. Chem. Eng. 72 (1994), 160-163.

[24] M.M. Ali, C. Storey, A. Törn, Application of stochastic global optimisation algorithms to practical problems, *Journal of optimization theory and applications 95*, 1997, 545-563.

[25] Ch. A. Floudas, P.M. Pardalos, C. S. Adjiman, W. R. Esposito, Z. H. Gumus, S.T. Harding, J.L. Klepeis, C. A. Meyer, C. A. Schweiger, *Handbook of test problems in local and global optimization* (Dordrecht, The Netherlands: Kluwer Academic Publishers, 1996).

[26] H. P. Schwefel, *Evolution and Optimum Seeking* (NY: John Wiley & Sons, INC, 1995).

[27]. M.M. Ali, C. Storey, Modified controlled random search algorithms, *International Journal of Computer Mathematics*, 53, 1994, 229-235.

[28] W. L. Price, Global Optimization by Controlled Random Search, *Journal of Optimization Theory and Applications 40*(3), 1983, 333-348.

4 Parameter control strategy in differential evolution algorithm for optimal control[†]

4.1 Abstract

Most optimal control algorithms are not capable of finding the global solution among local ones. Because of this we recently proposed the use of a Differential Evolution algorithm to solve multimodal optimal control problems. The DE algorithm is efficient compared to most other evolutionary algorithms. Still, when applied to optimal control problems, the algorithm is significantly less efficient than other, non-global, optimal control algorithms. In this paper the efficiency of the DE algorithm for optimal control is improved significantly through parameter control. In the DE algorithm three main parameters have to be set by the user. The parameter values all constitute a compromise between the efficiency of the algorithm and the capability of finding the global minimum. Instead of keeping these parameters constant, which is common practice, these parameters are changed during the optimization. Roughly speaking in the beginning of the optimization it is important to search the whole space, while after some time, to improve the efficiency, the search must be more local. Based on the diversity of intermediate computations, our algorithm makes this transition, i.e. the change of the parameters, more quickly or slowly. The algorithm is illustrated through numerical solutions of two multimodal optimal control problems.

KEY WORDS: Evolutionary Algorithms, Differential Evolution, Optimal Control, Optimization.

4.2 Introduction

Evolutionary Algorithms are appealing because of their capability to locate the global optimum of optimal control problems. However, it is well known that the majority of those algorithms demand an excessive number of function evaluations. Instead of an 'ordinary' function evaluation in optimal control problems this evaluation constitutes the numerical integration of a set of differential equations. The recently proposed Differential Evolution (DE) [1, 2] algorithm seems to be a good candidate to surpass these drawbacks, since it was shown to converge efficiently to the global optimum in some benchmark optimal control problems [3]. In spite of its simplicity, in all differential evolution algorithms there are three parameters, namely, the population size (μ), the amplification of the differential variation (F) and the crossover constant (CR) that have to be set by the user. Little is known on the choice of the values of these parameters, when applying differential evolution to solve multimodal optimal control problems. Following the guidance of Storn and Price [1], which applies to static optimization problems, a reasonable set of parameters can be selected a-priori. To further improve the performance of the algorithm parameter tuning is proposed. This tuning however requires an excessive number of experiments. Several limitations

[†] Published by I.L. Lopez Cruz, L.G. Van Willigenburg and G. Van Straten in Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing, May 21-24, 2001, Cancun, Mexico, pp. 211-216.

of parameter tuning vis-à-vis the potential advantages of parameter control in evolutionary algorithms have been discussed recently [4].

In this paper we propose a parameter control strategy for the DE algorithm. The proposed algorithm is based on a heuristic rule that takes some feedback information from the actual population, a measure of the diversity of the population, in order to modify the values of the amplification parameter (F) and also the crossover constant (CR). Broadly speaking the strategy modifies the effect of the parameters amplification variation and crossover at different stages of the search trying to do a search across the whole space at early stages and speeding up the search process when it gets close to convergence. That this strategy improves the search of DE is demonstrated by implementing it in the differential evolution algorithm in which the vector to be mutated is selected randomly. Only one difference vector is used for the mutation and the crossover is given by binomial experiments. To evaluate the performance of this strategy against a classical parameter tuning approach two difficult multimodal benchmark optimal control problems are solved. The paper is organized as follows. We first briefly describe the differential evolution algorithm in section two. In section three we summarize the parameter tuning approach and we describe our parameter control strategy and, finally, in section four the results obtained from the solution of two multimodal optimal control problems are discussed.

4.3 The differential evolution algorithm

Differential Evolution algorithms (figure 1) comprise a class of evolutionary algorithms recently proposed in the literature [1, 2]. Just like Evolution Strategies they use chromosomes based on floating-point numbers to represent candidate solutions. Each individual is defined as a d-dimensional vector $\vec{a} \in R^d$ and the whole population of potential solutions at generation g, is given by $P(g) = \{\vec{a}_i, ..., \vec{a}_\mu\}$. Here the population size μ does not change during the search. The main evolutionary operator in DE is completely different from other evolutionary algorithms since mutation neither is based on the alteration of genes by using a mutation probability nor rest on the use of a defined probability distribution function. In DE the mutation operator mutates μ vectors through the weighted difference of two (or four) others vectors according to:

Figure 1. Differential Evolution algorithm.

```
Generate random solutions that cover the given space.
Evaluate each solution.
g=1;
while (convergence is not reached)
    for i=1 to Population size
        Apply differential mutation.
        Execute differential crossover.
        Clip the solutions if necessary.
        Evaluate the new solution.
        Apply differential selection.
    end
    g=g+1;
```

$$\vec{v}_i = \vec{a}_{r_1} + F \times (\vec{a}_{r_2} - \vec{a}_{r_3}) \qquad (1)$$

where $i = 1,2,...,\mu$, and the random indexes $r_1, r_2, r_3 \in [1,2,...,\mu]$ are mutually different and also distinct from the index i. $F \in [0,2]$ is a real constant which affects the differential variation between two vectors. As in other evolutionary algorithms, the crossover operator is introduced in order to increase the diversity of the population.

The crossover operator combines the previously mutated vector $\vec{v}_i = [v_{1i}, v_{2i}, ..., v_{di}]$ with a so-called target vector $\vec{a}_i = [a_{1i}, a_{2i}, ..., a_{di}]$ to generate a named trial vector $\vec{a}'_i = [a'_{1i}, a'_{2i}, ..., a'_{di}]$ according to:

$$a'_{ji} = \begin{cases} v_{ji} \text{ } if \text{ } (randb(j) \le CR) \text{ } or \text{ } j = rnbr(i) \\ a_{ji} \text{ } if \text{ } (randb(j) > CR) \text{ } and \text{ } j \ne rnbr(i) \end{cases} j = 1,2,...,d; i = 1,2,...,\mu \qquad (2)$$

where $randb(j) \in [0,1]$ is the j-th evaluation of a uniform random number generator. $CR \in [0,1]$ is the crossover parameter. $rnbr(i) \in [1,2,...,d]$ is a randomly chosen index. Each member (i) of the population plays once the role of a target vector, thus, there are μ competitions at each generation.

The population size (μ), the differential variation parameter (F) and the crossover constant (CR) are parameters in the algorithm that have to be set the user. The selection operator only compares the cost function value of both competing vectors (target and trial vectors) and the better individual becomes a member of the population for the next generation. That means:

$$\text{if}\, \Phi(\vec{a}'_i(g)) < \Phi(\vec{a}_i(g)) \text{ then } \vec{a}_i(g+1) := \vec{a}'_i(g)$$

$$\text{else } \vec{a}_i(g+1) := \vec{a}_i(g); i = 1,2,...,\mu \qquad (3)$$

where Φ is the performance function that has to be minimized. According to the notation proposed by Storn and Price [1] the previous algorithm can be shortly described by: *DE/rand/1/bin*. This stands for a differential evolution algorithm in which the vector to be mutated ($\vec{a}_{r_1}$ in (1)) is selected randomly from the population. Only the difference of two vectors is considered by the mutation operation in (1). The crossover is due to the binomial scheme (eq. 2). It is apparent that there are more differential evolution algorithms. We have studied the behavior of in total five different DEs somewhere else [3].

When we want to use DE to solve optimal control problems an extension is required in order to deal with constraints for the controls. A clipping technique has been introduced to guarantee that only feasible trial vectors are generated after the application of mutation and crossover operators, as follows:

$$a'_{ji}(g) = \begin{cases} \beta_j \text{ } if \text{ } a'_{ji}(g) > \beta_j \\ \alpha_j \text{ } if \text{ } a'_{ji}(g) < \alpha_j \end{cases} j = 1,...,d; i = 1,...,\mu \qquad (4)$$

where α_j and β_j are the lower and upper bounds on a_j.

4.4 Parameter tuning and control in differential evolution algorithm

The values of population size (μ), the amplification variation (F) and crossover constant (CR) greatly determine whether DE will find an optimum solution and whether it will find an acceptable solution efficiently. Roughly speaking, by using greater values of population size, DE has more chances to converge to the global optimum at the expense of computation time. Greater values of the differential variation coefficient make it possible to explore the whole search space and to prevent premature convergence of DE. A value for the crossover constant close to one increases the speed of convergence. The values of these parameters can be determined either by parameter tuning (PT) or parameter control (PC).

Parameter tuning (PT) in evolutionary algorithms amounts to finding parameter's values that do not change during the optimization. Therefore, a number of experiments are needed in order to find out the best combination of those parameters that give the solution of the problem accurately and efficiently. An experimental design would require a huge amount of experiments. Therefore, in order to solve optimal control problems by DE applying a parameter tuning approach, we have proceed as follows. Since the rule of thumb of Storn and Price [1] does not give enough information about the selection of the population size, we have followed three heuristic rules in applying DE to solve dynamic optimization problems.

Heuristic one: Seeing that in solving optimal control problems a greater value of population size entails an increased number of function evaluations and computation time, some values for μ near the dimension of the optimization problem are tried. If premature convergence is observed the population size can be increased. Otherwise it can be decreased.

Heuristic two: Following the rule of thumb suggested by Storn and Price [1] we have chosen values of F=0.5 for the amplification variation parameter, and CR=0.1 for the crossover constant as starting candidates for each selected population size. Our examples show that, indeed, sometimes this value of F works very well (see problem 1 below), but also sometimes it is necessary to increase that value in order to avoid premature convergence (see example 2 below).

Heuristic three: According to another suggestion of Storn and Price, since changing the values of crossover constant CR from 0.1 to 0.9 or 1.0 speeds up convergence, possibly at the expense of the quality of the solution, we have tried these higher values for CR always. Sometimes increasing the value of the parameter CR is very convenient (see example 1 below), but in other cases values different than zero might deteriorate the quality of the solution (see example 2 below) due to convergence to a local optimum.

Parameter control (PC) in evolutionary algorithms, on the other hand, means that the values of the design parameters are changed during the optimization. There are two general ways to do this. Either by using a heuristic rule, which takes feedback from the current state of the search and modifies the parameter values accordingly or by incorporating the parameters into the chromosomes [4]. Following the first approach we have devised a parameter control strategy for the Differential Evolution algorithm. We have tested its behaviour on the differential evolution algorithm DE*/rand/1/bin*. Yet, it can be applied to other DE algorithms as well. Our proposed parameter control strategy is based on the assumption that for multimodal problems we want to explore the search space as much as possible at the beginning of the search. Assuming that the population size does not change during a run, this means that the parameter F, controlling the differential variation in DE, should take greater values at the beginning of the optimization than at later stages as the algorithm is approaching a solution. Conversely, the crossover parameter CR at early stages should be smaller than at stages close to convergence. On the other hand, as soon as the algorithm is approaching a solution we should reduce the effect of the mutation parameter and because we are close to a solution it would be advisable to speed up the convergence by enlarging the effect of the crossover parameter. The difficulty, however, is how to determine the status of the current search. There are several possibilities to do this, for instance, a measure of the population diversity, relative improvements or absolute solution quality [4]. Here we have chosen a measure of population diversity [5] around the current best solution, which is defined as follows. First a diversity index η_{ji} is required:

$$\eta_{ji} = \begin{cases} 1 & \text{if} \quad \left| \dfrac{a_{ji} - a_{j,best}}{a_{j,best}} \right| > \varepsilon_2 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $a_{j,best}$ is the *j*th gene of the individual with the best performance (i.e.: $\Phi(\bar{a}_{best}) \le \Phi(\bar{a}_i), \forall_i \ne best$) in the population, and ε_2 is an assigned tolerance for the gene diversity. A value of $\eta = 1$ means a diversified gene. Next the degree of population diversity ($0 \le \rho \le 1$) is calculated as follows:

$$\rho = \sum_{\substack{i=1 \\ i \ne best}}^{\mu} \sum_{j=1}^{d} \eta_{ji} / (d \times (\mu - 1)) \tag{6}$$

Then, by using this measure of diversity in the population (ρ) a heuristic rule that allows us to modify two parameters in DE can be written as follows:

if $\rho > \varepsilon_1$ then

F=0.9;CR=0.1

else

F=0.5;CR=0.5 (7)

where ε_1 is a desired tolerance.

4.5 Results from two-benchmark multimodal optimal control problems

4.5.1 *The optimal control of a multimodal CSTR*

We have solved by means of differential evolution a well-known multimodal optimal control problem that consists in determining the optimal control *u(t)* in the time interval $0 \le t \le t_f$ that minimizes the performance index:

$$J = \int_0^{t_f} (x_1^2 + x_2^2 + 0.1u^2) dt \tag{8}$$

subjected to the dynamic equations:

$$\dot{x}_1 = -(2+u)(x_1 + 0.25) + (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) \tag{9}$$

$$\dot{x}_2 = 0.5 - x_2 - (x_2 + 0.5)\exp(\frac{25x_1}{x_1 + 2}) \qquad (10)$$

from chemical reaction engineering [6]. Herein x_1 represents deviation from dimensionless steady-state temperature and x_2 stands for deviation from dimensionless steady-state concentration. Both state variables model a first-order irreversible chemical reaction carried out in a continuous stirred tank reactor (CSTR). The control *u(t)* represents the manipulation of the flow-rate of the cooling fluid through a coil inserted in the reactor. The final time $t_f = 0.78$ is fixed. The initial conditions are: $x(0) = [0.09 \quad 0.09]^T$. It can be shown that according to classical methods in optimal control this problem has two solutions. The global optimum is associated with a performance index value of J*=0.13309 and the local optimum has a cost J*=0.24443 [3].

In order to solve this optimal control problem by a direct optimization method like Differential Evolution it is necessary to approximate the continuous-time optimal control problem to a non-linear programming problem (NLP). This can be done by control parameterization [7]. The time interval $t \in [0, t_f]$is discretized using N time intervals $0 = t_0 < t_1 <,...,< t_N = t_f$. Next the control is assumed to be piecewise constant at each time interval:

$$u(t) \cong u(t_k) = u_k, t \in [t_k, t_{k+1}), k = 1,..., N \qquad (11)$$

This parameterization of the controls is used to solve the dynamic equations (eqns 9 and 10) and also to evaluate the performance index (J) given by equation (8).

When a piece-wise constant parameterization of the control with 13 intervals (parameters) is adopted, the global optimum for the CSTR optimal control has a cost function around J*=0.1356. The local optimum presents a performance index value of J*=0.2446. So as to solve this dynamic optimization problem by differential evolution it is assumed that the control is inside the bounds $0 \le u(t) \le 5$. The initial population of the DE algorithm was generated uniformly from that domain. To evaluate the integral (J) accurately the Mayer formulation of this optimal control problem was adopted and three differential equations were solved. A variable step size integration routine (function *ode45.m* of MATLAB) was used to integrate the differential equations and the relative tolerance for the integration was set to 1e-8. In order to speed up the simulation a C-MEX file s-function containing the dynamic differential equations was written.

An absolute convergence criterion for DE algorithm was chosen as $J_w - J_b < \varepsilon_c$ which is reasonable seeing that both the states (x) and the control (u) are normalized in this problem. Where J_w and J_b are the worst and the best performance index values from the population. Values of the tolerances in the parameter control strategy were $\varepsilon_1 = 0.5$and $\varepsilon_2 = 0.1$. The tolerance $\varepsilon_c$constant was $\varepsilon_c = 0.00001$. Table 1 shows main the main results obtained with the two algorithms described before where the averages from ten runs are reported. All the time the DE algorithm converged to the global optimum solution.

Table 1. Comparison of parameter tuning and parameter control in DE/rand/1/bin from solving the optimal control of a CSTR

DE strategy	PT	PC	PT	PC	PT	PC
μ	20		15		10	
F.E.	4388	4292	4402.50	2913	3116	1884
CPU time#[secs]	308.13	250.91	313.08	198.41	222.94	138.33
J*	0.1355943	0.1355846	0.1356408	0.1355878	0.1382629	0.1360762
CE(%)	100	100	100	100	100	100
Generations	219.4	214.6	293.5	194.2	311.6	188.4
Parameters	F=0.5,CR=0.5	eq. (7)	F=0.5,CR=0.2	eq. (7)	F=0.5,CR=0.2	eq. (7)

F.E.=function evaluations. CE=percentage of hitting the global optimum. #On a Pentium III at 700 MHZ PC. Averages of ten runs.

Values for parameters F and CR were, in the case of parameter tuning, obtained through the heuristic rules described before (section 3). Clearly the number of function evaluations is always less in case of the parameter control strategy and also the quality of the solution is slightly better.

4.5.2 *The optimal control of the bifunctional catalyst blend problem.*

A highly multimodal optimal control problem has been posed by Luus [6]. A chemical process involves converting methylcyclopentane to benzene in a tubular reactor is modeled by a set of seven differential equations (12)-(18):

$$\dot{x}_1 = -k_1 x_1 \tag{12}$$

$$\dot{x}_2 = k_1 x_1 - (k_2 + k_3) x_2 + k_4 x_5 \tag{13}$$

$$\dot{x}_3 = k_2 x_2 \tag{14}$$

$$\dot{x}_4 = -k_6 x_4 + k_5 x_5 \tag{15}$$

$$\dot{x}_5 = k_3 x_2 + k_6 x_4 - (k_4 + k_5 + k_8 + k_9) x_5 + k_7 x_6 + k_{10} x_7 \tag{16}$$

$$\dot{x}_6 = k_8 x_5 - k_7 x_6 \tag{17}$$

$$\dot{x}_7 = k_9 x_5 - k_{10} x_7 \tag{18}$$

where the states $x_i; i = 1,\ldots,7$ are the mole fractions of the chemical species and the rate coefficients (k_i) are cubic functions of the catalyst blend (control u(t)), according to:

$$k_i = c_{i1} + c_{i2} u + c_{i3} u^2 + c_{i4} u^3; i = 1,\ldots,10 \tag{19}$$

where all the values of the constants c_{ij} are given in reference [6]. 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 values for the states are:

$x(0) = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$

The optimal control problem consist in to find the catalyst blend along the length of the reactor in the interval $0 \leq t \leq t_f$, where $t_f = 2000 g \cdot h / mol$ such that the concentration of benzene is maximized. Thus the performance index to be maximized is given by:

$$J = x_7(t_f) \times 10^3 \qquad (20)$$

where 10^3 is a scaling factor. This problem has been studied by Luus [6] who by using Iterative Dynamic Programming has shown that there are a lot of local optima. The global optimum is around the performance index value of J*=10.0942 when a piecewise constant control parameterization with ten intervals is adopted.

In order to solve this problem by differential evolution, the initial population was generated randomly from the parameterized control space. As for the first example, a variable step size integration routine (function *ode45.m* of MATLAB) was used to integrate the dynamic equations (12-18). A relative tolerance of 1e-8 was set for the integration. Also a C-MEX file s-function with the differential equations was written so as to speed up the simulations. Again, a criterion of convergence based on the population was selected. But, in contrast to the first example here a relative convergence criterion was chosen as follows:

$$\frac{\mu}{\varepsilon_d}(J_w - J_b) \leq \left| \sum_{i=1}^{\mu} J(\tilde{u}_i) \right| \qquad (21)$$

where J_w and J_b are defined as earlier and $\varepsilon_d = 0.001$. The values of the tolerances for the parameter control strategy were: $\varepsilon_1 = 0.02$ and $\varepsilon_2 = 0.02$.

Table 2. Comparison of parameter tuning and parameter control in DE/rand/1/bin from solving the optimal control of the blend of a bifunctional catalyst

DE strategy	PT	PC	PT	PC	PT	PC
μ	25		20		15	
F.E.	3117.5	1635	2604	1240	1858.3	937.50
CPU time#[secs]	390.07	224.18	339.58	171.70	246.90	127.70
J*	10.0942	10.0940	10.0942	10.0906	10.0942	10.0919
CE(%)	*100*	*100*	*100*	*100*	90	80
Generations	124.7	65.40	130.2	62	123.88	62.50
Parameters	F=0.9,CR=0.0	eq. (7)	F=0.9,CR=0.0	eq. (7)	F=0.9,CR=0.0	eq. (7)

F.E.=function evaluations. CE=percentage of hitting the global optimum. # On a Pentium III at 700 MHZ PC. Averages of ten runs.

Table 2 shows main results when DE was used to solve this problem. It can be seen from table 2 that the DE method mostly converges to the global optimum. However, as the population size is smaller the algorithm is affected by premature convergence. On the other hand, it is apparent that the number of function evaluations was significantly smaller for the parameter control strategy than in the case of parameter tuning. It seems possible to improve the quality of the solution obtained by means of parameter control by increasing the population size without augmenting too much the number of function evaluations.

4.6 Discussion

The parameter control strategy introduces two new parameters, namely, the tolerances ε_1 and ε_2. However, in contrast to the original parameters these tolerances have a clear meaning and, in addition, there are less reasons why they themselves should be changed during the optimization. In any case, they have led to larger efficiency without any tuning.

4.7 Conclusions

Through the solution of two multi-modal optimal control problems we have demonstrated that implementation of a parameter control strategy in the Differential Evolution algorithm has significant advantages over widely practiced parameter tuning approach. The parameter control strategy, which takes into account the diversity in the population in order to modify the values of the differential variation (F) and crossover (CR) parameters during the optimization is able to significantly improve the efficiency of the algorithm, and also the quality of the solution as compared to those obtained by parameter tuning. In future work we plan to apply and study the behavior of the proposed parameter control strategy in other Differential Evolution algorithms. Within the parameter control strategy the design of measures to determine the current convergence status of the search is an important issue that will be addressed.

4.8 References

[1] R. Storn and K. Price, Differential Evolution-a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization, 11*, 1997, 341-359.

[2] R. Storn, System design by Constraint Adaptation and Differential Evolution, *IEEE Transactions on Evolutionary Computation, 3*(1), 1999, 22-34.

[3] I.L. Lopez-Cruz, L.G. Van Willigenburg, G. Van Straten, Evolutionary Algorithms for multimodal optimal control problems: a comparative study, In preparation.

[4] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter Control in Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, 3(2), 1999, 124-141.

[5] J.P. Chiou and F.S. Wang, Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process, Computers and Chemical Engineering, 23, 1999, 1277-1291.

[6] R. Luus, Iterative Dynamic Programming (Chapman & Hall/CRC, Boca Raton, Florida, 2000).

[7] C.J. Goh and K.L. Teo, Control parametrization: a Unified Approach to Optimal Control Problems with General Constraints, *Automatica, 24*(1), 1988, 3-18.

For a complete description refer to the paper by Seginer et al. 1998 [5]. In Figure 1 a relational diagram, the backbone of the model is presented. The nitrate balance is not shown there. A summary of the main equations is presented in Appendix 1. A brief description of the model is as follows. The model has two state variables: non-structural carbon content (S_{Cv}) and structural carbon content (S_{Cs}) measured in moles [C] per unit surface area. No distinction between shoot and root is made in the model. Also long-term storage of assimilates is neglected. Some additional assumptions are that the volume of the vacuoles is a fixed fraction of the total volume of the plant; carbon-to-nitrogen ratio in the structure is also fixed, whereas the ratio in the vacuoles is variable, but constrained by the need to maintain a constant turgor pressure. It is assumed that there are no limitations in the flow of nitrate into the vacuoles to support growth and to maintain turgor. The plant grows by building new cells with exactly the same proportions as the already existing cells.

The model assumes that photosynthesis depends on light and CO_2 only, whereas respiration and growth are assumed to depend on temperature. Both photosynthesis and growth depend on the size of the crop as well. In this way, the non-structural carbon content is a result of the photosynthetic activity of the plant (F_{Cav}, eqn. 3) driven by light (I) and CO_2 (C_{Ca}). The maintenance (F_{Cm}, eqn. 5) and growth (F_{Cg}, eqn. 6) of the plant draw upon the produced carbohydrates under the influence of temperature (T). The photosynthetic process is described by a rectangular hyperbola function (eqn. 8). Both photosynthesis and growth functions include an exponential canopy closure function (eqn. 10) to relate these processes to light-intercepting leaf area. And also both functions have an inhibition function as a mechanism to restrict the growth of the plant and the carbohydrates production, as explained below.

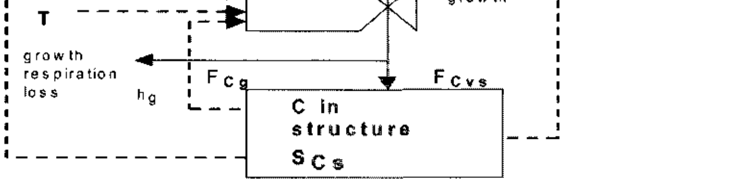


Figure 1. Dynamic model of lettuce crop. **I** is global solar radiation, **Cca** is the carbon dioxide concentration and T is the air temperature inside the greenhouse.

The growth inhibition function ($h_g\{C_{Cv}\}$, eqn. 12) and photosynthesis inhibition function ($h_p\{ C_{Cv}\}$, eqn. 9) both depend on the non-structural carbon concentration (C_{Cv}) which is calculated from the state variables of the model (eqn. 7). If as a result of the activity of light and CO_2 the non-structural carbon concentration (the assimilates stock) approaches zero then the growth of the plant is reduced. That means that the growth switching function $h_g\{C_{Cv}\}$ decreases to zero. On the other hand, when the carbon assimilates in the vacuoles is too high the photosynthesis inhibition function $h_p\{C_{Cv}\}$ approaches zero and brings the photosynthetic activity to a halt. From the basic model it is possible to derive some observable variables or output variables, for instance, plant dry weight matter (eqn. 19), which can be used to several purposes as the calibration of the model. A simulation program of this model was implemented in MATLAB-Simulink environment. The program uses a C-MEX file S-function so as to increase the speed of simulations.

5.1.4 Genetic Algorithms

The use of optimisation algorithms to calibrate a dynamic model is nowadays a common activity [8],[4]. We use a method inspired by the theory of evolution and natural genetics to calibrate the model aforementioned. A genetic algorithm (GA) [2] is an example of an Evolutionary Algorithm [1]. EAs are a relatively novel group of probabilistic search methods with robust properties as global optimisation procedures. A genetic algorithm has at least four properties: a population of chromosomes P(t) which contains candidate solutions to the problem, selection according to the fitness of each solution, crossover to produce new offspring, and random mutation [6]. Even though a GA is conceptually simple it performs well on many different types of problems. In figure 2 the general view of a GA is presented [5].

Figure 2. Pseudo-code of a Genetic Algorithm

```
Procedure Genetic Algorithm
begin
        t =0;
        initialise P(t);
        evaluate P(t);
while (not termination-condition) do
        t=t+1;
        select P(t) from P(t-1);
        alter P(t);
        evaluate P(t);
end
end
```

A genetic algorithm starts with a randomly or knowledge-based generated population P (t) of *n* candidate solutions to a problem. In a classic GA a binary representation is commonly used. For more complex GAs other structures (like vectors of integer or real numbers) can be used. Each chromosome of the population is evaluated to obtain a measure of its fitness. Then, a new population is formed, by selecting the more fit individuals. Some members of the new population are altered by means of crossover and mutation operators. The role of crossover is to combine features of the two solutions to form two offspring by swapping segments of the parents. We expect that through the information exchange of both solutions it can be possible to generate better solutions. The mutation operator stochastically modifies some genes of a selected chromosome with a probability equal to the mutation rate. The mutation operator introduces variability in the population.

Table 1. Genetic operators used in the calibration of the model.

Type of chromosome	Selection	Crossover	Mutation
Binary	Normalised geometric ranking. p_b =0.08	Simple (p_c=0.6)	Binary mutation (p_m=0.05)
Real	Normalised geometric ranking. p_b =0.08	Arithmetic, Heuristic, simple	Boundary, multi-non-uniform, non-uniform, uniform mutation

In this research both a GA with binary chromosome representation as well as a GA with real representation were investigated. Each chromosome of the initial population was generated in a random way but they were within a range given by expert-knowledge. That means that reasonable ranges were defined for all the parameters under study. The fitness function was a simple sum of squares of the error between observed values of the dry weight matter of the plant and the simulated value. The number of generations was five hundred for all the simulations. And the number of chromosomes in the population was fifty. Because the genetic operators depend on the kind of chromosome representation used they are presented in table 1. The probability of selecting the best individual (p_b), probability of crossover (p_c) and probability of mutation (p_m) used for the binary GA are presented there. The real GA used the same selection operator. We applied the genetic operators proposed by Michalewicz [5] for the case of real GA. The values for all their parameters were taken from Houck et al. 1995 [3]. The Genetic Algorithms Optimization Toolbox (GAOT) software available for MATLAB [3] was used for all the simulations because it is easy to link with a simulation model built in MATLAB-Simulink.

5.1.5 Calibration results

The calibration was done using original parameterisation done by Seginer et al. [7]. Only the parameters that were known from previous experience to have a strong effect in the model were incorporated in the calibration. Thus, the growth (m), the apparent light efficiency parameter (ε) and the CO_2 transport coefficient (σ) parameters were considered. Two sets of environmental data collected by Van Henten [10] were used as inputs of the model. The calibration results using both GA implementations

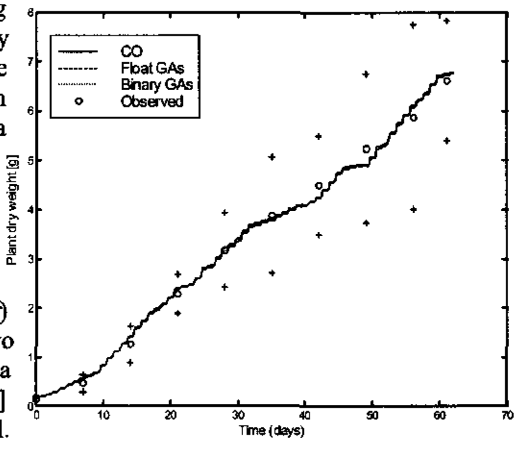


Figure 3. Model calibration results for experiment 1.

and a local optimisation method are shown in Figure 3 and Figure 4 respectively. The calculated values of the calibrated parameters for both experiments are presented in table 2. The results of both GAs are compared with a classic optimisation algorithm based on Sequential Quadratic Programming algorithm. Since in many cases the local optimisation algorithm converged to a local minimum, a kind of global optimisation for it was implemented. The best value reached after forty optimisations using random guessed initial parameter values, are presented here. From this table we can conclude that the GA implementations were always superior. For the case of experiment 1 the solution found by both genetic algorithms was essentially the same. Figure 3 shows minor differences between all implementations.

Table 2. Results of parameters calibration.

	m		ε		σ		Error	
	exp. 1	exp. 2	exp. 1	exp. 2	exp. 1	exp. 2	exp. 1	exp. 2
LO	8.0214	32.512	0.0751	0.0717	0.0011	0.0011	0.2955	0.1967
B GA	7.5532	30.728	0.0670	0.0717	0.0020	0.0011	0.2849	0.1968
F GA	7.5661	32.995	0.0670	0.0708	0.0020	0.0011	0.2849	0.1963

For experiment 2 the genetic algorithm with real number code chromosomes got the lowest error value but the differences are very small. This result confirms some advantages of the use of this GA instead of the binary version [5]. We could observe that for several runs the floating point GA converged to an error of 0.1985 and values for the parameters m= 20.8843, ε=0.0733, and σ=0.0011. Even though the difference between both error values is small we can see the value of the parameter *m* changes considerably. This can be explained by the difficulties to determine the growth parameter (m) using information of growth experiments [9]. Figure 4 shows the simulation results using the parameters calculated for all calibrations. Minor differences can be observed between both GAs.

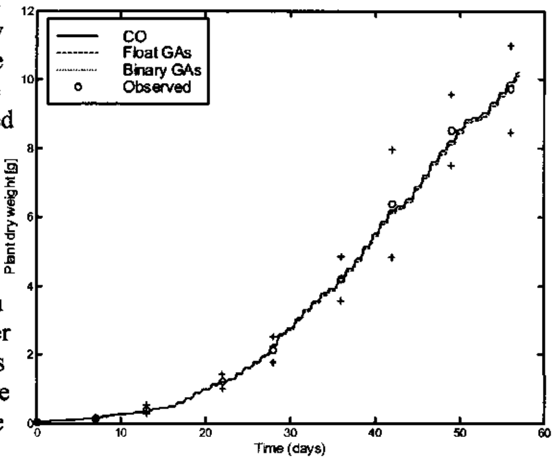


Figure 4. Model calibration results for experiment 2.

5.1.6 Sensitivity analysis

In a previous work [9] the sensitivity of the states of this model to its parameters was presented. In this study we show the results of the calculation of the sensitivity of plant dry weight (Y_{dw}) model output to changes of the parameters in the model. The relative sensitivity was calculated using $S_i = \frac{\partial Y_{dw}(t,\theta^o)}{\partial \theta_i}\frac{\theta_i}{Y_{dw}(t,\theta^o)}$, and solving the algebraic equation

$S_i^Y = \frac{\partial Y_{dw}(t,\theta^o)}{\partial x_1}\frac{\partial x_1}{\partial \theta_i} + \frac{\partial Y_{dw}(t,\theta^o)}{\partial x_2}\frac{\partial x_2}{\partial \theta_i} + \frac{\partial Y_{dw}(t,\theta^o)}{\partial \theta_i}$ along with the solution of the sensitivity equations. The results are presented in table 3.

From this table we can conclude that the most important parameter in the model that act upon the plant dry weight is the apparent light use efficiency (ε). The CO_2 transport coefficient (σ), closure parameter (a) and growth parameter (m) have less effect. The sensitivity results help to understand the large differences in parameter values for m and σ between the two experiments. They cannot be determined accurately in cases where the sensitivity is low. The differences between the role of m and σ in both experiments, could be explained by the different environmental conditions of both experiments as was discussed before in [9].

Table 3. Relative sensitivity of plant dry weight to changes in the parameters in decreasing order of importance for both experiments.

	Experiment 1		Experiment 2
ε	0.84	ε	0.86
a	0.49	σ	0.48
m	0.34	a	0.45
κ	-0.23	θ	-0.25
θ	-0.17	k	-0.17
k	0.14	κ	-0.14
σ	0.13	c	0.08
b_p	0.10	b_g	-0.05
γ	-0.09	m	0.02
b_g	-0.04	γ	0.01
c	-0.01	s_g	-0.01
s_p	0.01	s_p	0.00
s_g	0.01	b_p	0.00

5.1.7 Conclusions

This study showed that Genetic Algorithms are suitable to do the calibration of dynamic models. However, a high computation time was required. This can be attributed to the quantity of function evaluations, a complete simulation of the dynamic model, in this case, executed by a GA to calculate the fitness of each potential solution. Another explanation can be the inherent slow properties of the current interpreted MATLAB GA implementation. Although the computation time could not be a limitation to use this approach, additional work is needed to know the performance of GAs with more complex dynamic models and also to improve the efficiency of the present implementation.

5.1.8 References

[1] Back T. & Schwefel H-P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation* 1(1), pp. 1-23, 1993

[2] Goldberg, D.E. Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading MA,1989.

[3] Houck, Ch.,R., Joines, J., A., Kay, M.,G., A genetic algorithm to function optimization: A Matlab implementation, NCSU-IE TR 95-09, 1995.

[4] Metselaar, K. Auditing predictive models: a case of study in crop growth. PhD Thesis Wageningen Agricultural University. Wageningen, The Netherlands, 1999.

[5] Michalewicz, Z. Genetic Algorithms + data structures = evolution programs, Springer-Verlag, Berlin, 1992.

[6] Mitchell, M. An introduction to genetic algorithms, MIT Press, Cambridge, Massachusetts, 1996.

[7] Seginer, I. , Buwalda , F. & Van Straten G. Nitrate concentration in greenhouse lettuce: a modelling study, *Acta Horticulturae* 456 (1998) 189-197.

[8] Solomatine, D.P. The use of global random search methods for models calibration. Proc. XXVIth Congress of the International Association for Hydraulic Research, London September 1995.

[9] Van Straten, G., Lopez-Cruz I., Seginer, I. , Buwalda, F. Dynamic model for control of nitrate in lettuce. 3rd International workshop ISHS, Models for plant growth and control of the shoot and root environments in greenhouses. The Volcani Center, Bet Dagan, Israel, February 21-25, 1999.

[10] Van Henten, E.J. Greenhouse climate management: an optimal control approach, PhD Thesis, Wageningen Agricultural University. Wageningen, The Netherlands, 1994.

Appendix 1. Basic lettuce model (NICOLET) equations

State equations

Carbon in vacuoles $$\frac{dS_{Cv}}{dt} = F_{Cav} - F_{Cm} - F_{Cg} - F_{Cvs} \qquad [1]$$

Carbon in structure $$\frac{dS_{Cs}}{dt} = F_{Cvs} \qquad [2]$$

C-fluxes

photosynthetic assimilation $$F_{Cav} = p\{I, C_a\}\, h_p\{S_{Cv}, S_{Cs}\}\, f\{S_{Cs}\} \qquad [3]$$

growth $$F_{Cvs} = g\{T\}\, h_g\{S_{Cv}, S_{Cs}\}\, f\{S_{Cs}\} \qquad [4]$$

maintenance respiration $$F_{Cm} = e\{T\}\, S_{Cs} \qquad [5]$$

growth respiration	$F_{Cg} = \theta F_{Cvs}$	[6]

Additional relations

carbon concentration in the vacuoles	$C_{Cv} = \kappa \dfrac{S_{Cv}}{S_{Cs}}$	[7]
uninhibited photosynthesis rate	$p\{I, C_{Ca}\} = \dfrac{\varepsilon I \, \sigma(C_{Ca} - C_{C^*})}{\varepsilon I + \sigma(C_{Ca} - C_{C^*})} \qquad C_{Ca} > C_{C^*}$	[8]
photosynthesis inhibition function	$h_p\{C_{Cv}\} = 1 - \left(1 + \exp\left\{- s_p\left(\dfrac{C_{Cv}\gamma}{\Pi_v} - b_p\right)\right\}\right)^{-1}$	[9]
canopy closure reduction function	$f\{S_{Cs}\} = 1 - \exp\{-aS_{Cs}\}$	[10]
maximum growth rate	$g\{T\} = mk\exp\{c(T - T^*)\}$	[11]
source depletion switching (inhibition) function	$h_g\{C_{Cv}\} = \left(1 + \exp\left\{- s_g\left(\dfrac{C_{Cv}\gamma}{\Pi_v} - b_g\right)\right\}\right)^{-1}$	[12]
specific maintenance respiration	$e\{T\} = k\exp\{c(T - T^*)\}$	[13]
osmotic pressure in vacuoles (Pa)	$\Pi_v = P_v + \Pi_r$	[14]
nitrate concentration in the vacuoles	$C_{Nv} = \dfrac{\Pi_v - \gamma C_{Cv}}{\beta}$	[15]

Outputs

dry matter (g m^{-2})	$DM = \eta_{OM/C}(S_{Cs} + S_{Cv}) + \eta_{MM/N} S_{Nv}$	[16]
	with $S_{Nv} = C_{Nv} \dfrac{S_{Cs}}{\kappa}$	[17]
fresh matter (g m^{-2})	$FM = \dfrac{DM}{\eta_{DM/FM}}$	[18]
dry weight per head (g)	$y_{dw} = DM/N_p$	[19]

5.2 Optimal control of nitrate in lettuce by gradient and differential evolution algorithms[†]

5.2.1 Abstract

Since high concentration levels of nitrate in lettuce are undesirable, its control is currently an important problem in the context of European Union regulations. Using a dynamic model that predicts the amount of nitrate at harvest time, an optimal control problem is formulated and solved through an enhanced classical gradient method and a Differential Evolution algorithm. This work shows that in order to avoid local minima an efficient evolutionary algorithm may be applied to solve optimal control problems or to provide a good initial guess for a classical method, which solves smooth continuous-time optimal control problems more accurately and efficiently.

Keywords: Optimal control, Artificial Intelligence, Genetic Algorithms, Global optimization, Gradient methods

5.2.2 Introduction

High concentration levels of nitrate in lettuce crop and other leafy vegetables are undesirable because they have a negative effect on human health. Therefore, methods are sought to control the nitrate levels of a greenhouse lettuce crop. As a first step a model of lettuce growth, which predicts the amount of nitrate content at harvest time, has been proposed (Seginer et al., 1998). An optimal control problem has been formulated and some properties of its solution have been analyzed using a simplified lettuce model (Ioslovich and Seginer, 2000). Also the full two-state nonlinear lettuce model has been used to get a numerical solution to another optimal control problem that uses light and temperature as control inputs by means of a first order gradient algorithm (Stigter and Van Straten, 2000). The aim of this paper is to solve a new optimal control problem that includes light, temperature and also carbon dioxide concentration as control inputs, by an evolutionary algorithm and to compare the results with those obtained by the Adjustable Control-variation Weight (ACW) gradient algorithm (Bryson, 1999, Weinreb, 1985). The evolutionary algorithm selected is the recently proposed Differential Evolution (DE) algorithm (Storn and Price, 1997), since DE is an evolutionary algorithm that can approximate the global optimum and is also very efficient computationally compared to other evolutionary algorithms. The paper is organized as follows: first a brief description of the optimal control problem is given; especially some properties of the dynamic lettuce model are emphasized. Then, the main characteristics of the Differential Evolution algorithm are outlined. Next, the results are described, compared and discussed.

5.2.3 Optimal control of nitrate in lettuce

[†] Published by I.L. Lopez Cruz, L.G. Van Willigenburg and G. Van Straten in Artificial Intelligence in Agriculture 2001, a proceedings volume from the 4th IFAC workshop Budapest, Hungary, 6-8 June 2001, pp. 123-128.

The lettuce model is based on carbon balances of the vacuoles and the structure that prevail in the plant cells. The so-called NICOLET model (Seginer et al., 1998) has two state variables: carbon content in the vacuoles (M_{Cv} mol[C] m^{-2} [ground]) and carbon content in the structure (M_{Cs} mol[C] m^{-2} [ground]) that represent a carbon source-sink relation in the plant driven by sunlight, temperature and carbon dioxide concentration. Photosynthesis and growth can proceed uninhibited as long as the non-structural carbon concentration in the vacuoles remains within certain limits. However, when affected by environmental conditions the non-structural carbon concentration approaches zero, growth will be reduced. In the model, this transition is implemented by introducing a smooth switching function which is one for non-inhibiting levels, but falls off to zero rapidly when the assimilate stock becomes empty. When carbon assimilates in the vacuoles are too high a similar switching function brings photosynthesis to a halt.

The core of the model is given by two differential equations:

$$\dot{M}_{Cv} = F_{Cav} - h_g F_{Cm} - F_{Cg} - F_{Cvs} \qquad (1)$$

$$\dot{M}_{Cs} = F_{Cvs} - (1 - h_g) F_{Cm} \qquad (2)$$

which represent the main carbon balances. The F's in equation (1) denote the rates of photosynthesis, maintenance, growth respiration and uninhibited growth, respectively, which are functions of the states and the inputs light (I [mol PAR $m^{-2}s^{-1}$]), carbon dioxide (C_{Ca} [mol m^{-3}]) and temperature (T °C). h_g denotes the inhibition function for growth.

The description of all model equations and parameter values is given in Yarkoni and McKenna, (2000). Here it is worthwhile to outline that in the version used here (NICOLET B3) two important modifications were made as compared to the original NICOLET model (Seginer et al., 1998). In NICOLET version B3 the inhibition functions for photosynthesis and growth were changed in such a way that they take a value of zero when vacuolar carbon concentrations reach the appropriate bound. And also, as seen in Eqns (1) and (2), the new model incorporates the depletion of structural matter to meet the requirements of maintenance respiration when the carbon content in the vacuoles is low. This situation may occur when the model is exposed to a long darkness period. It turned out that the original model predicts negative values for the carbon content in the vacuoles when it was used in the solution of an optimal control problem by the differential evolution algorithm with three control inputs. From the states of the lettuce model several outputs such as dry and fresh matter, sugars and nitrate concentration are calculated. The nitrate concentration follows from M_{Cv} by a negative algebraic correlation, which expresses the plants policy to maintain its turgor pressure.

One formulation of the optimal control of nitrate in lettuce is as follows. While minimizing the integral of light

$$J = \int_0^{t_f} I(t)dt \qquad (3)$$

calculate the control trajectories of light, carbon dioxide and temperature such that a desired fresh head weight of lettuce (y_{dfm} [gr]) and a specified amount of nitrate (y_{dNO_3} [ppm]) are obtained at a specified harvest time (t_f) i.e.,

$$y_{fm}(t_f) = y_{dfm} \tag{4}$$

$$y_{NO_3}(t_f) = y_{dNO_3} \tag{5}$$

where $y_{fm}(t_f)$ and $y_{NO_3}(t_f)$ are the corresponding outputs of the model. The control inputs are bounded because it is apparent that the light intensity cannot be negative and the same is true for carbon dioxide. Also the temperature must lie within a domain tolerated by the lettuce crop

$$I_{\min} \le I(t) \le I_{\max},\ C_{\min} \le C_{Ca}(t) \le C_{\max},\ T_{\min} \le T(t) \le T_{\max} \text{ for } 0 \le t \le t_f \tag{6}$$

5.2.4 **Extended Differential evolution algorithms**

A numerical solution to the specified optimal control problem can be obtained by indirect methods of optimal control like a first order gradient method, but also by direct methods. In this work a direct method based on evolutionary algorithms is used to approximate the global optimum solution. In the direct method the control trajectory is parameterised as a sequence of piece-wise constant values, which have to be found by the optimisation. The Differential Evolution algorithm is a kind of evolutionary algorithm that has recently been proposed for the solution of static parameter optimisation problems (Storn and Price, 1997). This algorithm has several nice properties over other evolutionary algorithms because it is easy to understand and very efficient computationally. An outline of this algorithm is presented in figure 1. As in other evolutionary algorithms main operators in DE are mutation, crossover and selection. Yet, in contrast to archetype genetic algorithms, they use a floating-point representation for the solutions. Also the main operator in DE is rather different than in other evolutionary algorithms. Similarly to Evolution Strategies here each chromosome is represented as a real parameter vector $\mathbf{a} = [a_1, \ldots, a_q]$, and it is required at generation g a population containing μ individuals $\mathbf{a}_i; i = 1, \ldots, \mu$. The essential feature of differential evolution algorithm rests on the mutation operator. A so-called mutant vector ($\mathbf{v}_i$), is generated by adding the weighted difference between two or four selected population vectors to another vector:

Figure 1. ***Differential Evolution algorithm***

```
Generate random solutions that cover the
given space.
Evaluate each solution.
g=1;
while (convergence is not reached)
   for i=1 to Population Size
         Apply differential mutation.
         Execute differential crossover.
         Clip the new solution if
         necessary.
         Evaluate the new solution.
         Apply differential selection.
   end
   g=g+1;
end
```

$$\mathbf{v}_i = \mathbf{a}_{r1} + F \cdot (\mathbf{a}_{r2} - \mathbf{a}_{r3}) \tag{7}$$

where ($\mathbf{a}_{r_1}$) is either a randomly selected vector or it represents the best solution from the current population. $F \in [0,2]$ is a factor that controls the amplification of the differential variation. The mutation operator implemented in this work was: $\mathbf{v}_i = \mathbf{a}_{best} + F \cdot (\mathbf{a}_{r1} + \mathbf{a}_{r2} - \mathbf{a}_{r3} - \mathbf{a}_{r4})$; $i = 1,\ldots,\mu$ (8)

The crossover operator increases the diversity of the mutated vector by means of the combination of two solutions:

$$a'_{ji} = \begin{cases} v_{ji} \ \text{if} \ randb \le CR \ \text{or} \ j = randr \\ a_{ji} \ \text{if} \ randb > CR \ \text{and} \ j \ne randr \end{cases}; \ i = 1,\ldots,\mu; j = 1,\ldots,q \qquad (9)$$

where $\mathbf{v}_{ji}$ is the j-th element of the mutated vector $\mathbf{v}_i$, $\mathbf{a}_i$ is a so-called target vector, against which each new solution is compared to, and *randb* is a uniform random number from [0,1]. $randr \in [1,2,\ldots,q]$ is a randomly chosen index, and *CR* is a parameter [0,1], which specifies the crossover constant. Selection is implemented only on the basis of a comparison between the cost function value of the new solution $\mathbf{a}'_i$ and that of the target vector $\mathbf{a}_i$. This means that if $J(\mathbf{a}'_i) \le J(\mathbf{a}_i)$ the new solution becomes a member of the population otherwise the old solution is retained. The inner loop in figure 1 implies that there are μ competitions at each generation since each member of the population plays the role of a target vector once.

Two extensions have been introduced in the original differential evolution algorithm. The first one is related to the fact that the controls are bounded so a clipping technique is introduced to prevent inadmissible solutions.

$$u_j = \begin{cases} u_j = \alpha_j \ \text{if} \ u_j < \alpha_j \\ u_j = \beta_j \ \text{if} \ u_j > \beta_j \end{cases}; \ j = 1,\ldots,q \qquad (10)$$

where α and β represent the lower and upper boundary of the control variables, respectively. The second modification is due to the fact that the optimal control of nitrate in lettuce presents constraints at harvest time (final time). The solution computed is based on the use of penalty functions. The augmented performance index is given by

$$J' = J + \lambda(g) dist(\mathbf{x}(t_f)) \qquad (11)$$

where $\lambda(g)$ is either a penalty factor depending on the current generation or a constant penalty factor, J is given by equation (3), and two options for *dist* are:

$$dist(\mathbf{x}(t_f)) = \begin{cases} \left|\mathbf{x}_d - \mathbf{x}(t_f)\right| \\ [\mathbf{x}_d - \mathbf{x}(t_f)]^T [\mathbf{x}_d - \mathbf{x}(t_f)] \end{cases} \qquad (12)$$

5.2. 5 Numerical results

5.2.5.1 A Solution obtained by a gradient algorithm

A solution of the optimal control of nitrate in lettuce was obtained by a classical first order gradient method. However, in contrast to the solution reported previously (Stigter and Van Straten, 2000) here the Adjustable Control Weight (ACW) gradient algorithm from Weinreb (1985) was implemented in order to deal with the constraints of the controls properly. This method uses an adjustable weighting matrix to modify the variation of the controls in the neighbourhood of the hard control bounds. The ACW gradient method solves the continuous-time optimal control problem according to the Pontryagin's Minimum Principle. The required gradients were calculated analytically. The final time was specified at $t_f = 60$ days. The constraints at the final time were $y_{dfm} = 400$ grams of head fresh weight, and $y_{dNO_3} = 3500$ ppm of nitrate concentration. The bounds on the controls were as follows: $0 \le I(t) \le 17.5$, $0.01 \le C_{Ca}(t) \le 0.04$, and $10 \le T(t) \le 30$. From several optimizations which were initialized with constant values for the controls a solution with a performance J*=306.3420 mol PAR m^{-2} was obtained. For one of the optimizations figure 2 shows the calculated optimal trajectories of fresh matter and nitrate content obtained after 3000 iterations of the ACW gradient method. The optimal solution satisfies almost exactly the two constraints at harvest time. The error for fresh head weight was 0.0012 grams, and 0.1057 ppm in case of nitrate concentration.

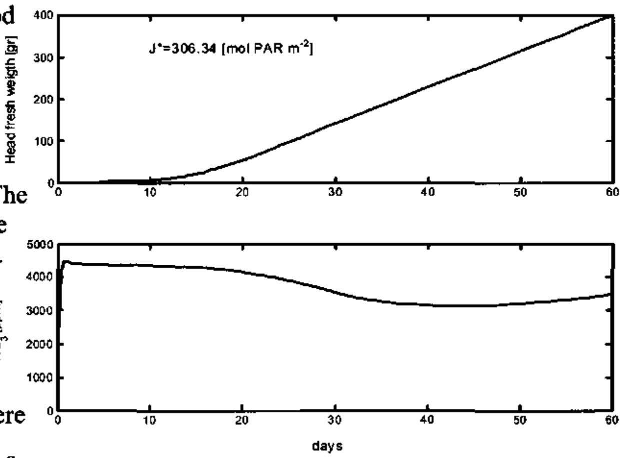


Figure 2. Optimal trajectories of head fresh weight and nitrate concentration calculated by the ACW gradient algorithm.

Figure 3 presents the optimal control: light, carbon dioxide and temperature. Since one goal of the optimal control problem formulation was the minimization of the integral of light, the calculated optimal solution shows that it is possible to control the nitrate level at harvest time by increasing the supply of carbon dioxide and decreasing the temperature. This result confirms that with artificial light it is possible to control nitrate levels in lettuce through the control of the shoot environment (Seginer et al., 1998). The optimal trajectory of carbon dioxide supply is at the upper specified limit (0.04 mol m^{-3}), which is consistent with the fact that no

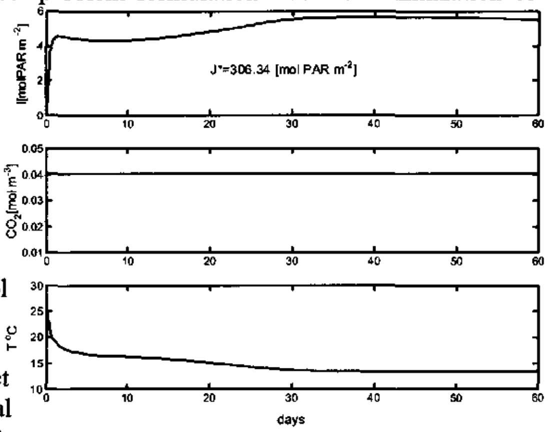


Figure 3. Optimal control inputs of light, carbon dioxide and temperature calculated by the ACW gradient algorithm.

cost was associated to it in the formulation of the optimal control problem.

The optimal trajectory of temperature presents a trend that goes down across the cultivation period. On the other hand, the optimal trajectory of light presents a sharp increase at earliest days, which can be explained by the demand of photosynthetic activity required to produce the desired fresh weight. Next, for the rest of the growing period, the amount of light is increased but not too much in order to meet the specification of the performance index and also to come up to the desired fresh head weight and the nitrate content at harvest time.

5.2.5.2 A solution obtained by a Differential Evolution algorithm

In order to solve the previous optimal control problem by the differential evolution algorithm, the first step is the selection of a reasonable parameterization for the control inputs. For the sake of keeping the number of parameters to be optimized as small as possible, only twenty time intervals (*N=20*) were selected. Then a piece-wise constant approximation for the three controls (*m=3*) was chosen:

$$\mathbf{u}(t) \cong \mathbf{u}(\mathrm{t_k}) = u_k^i \; t \in [t_k, t_{k+1}) \; k = 1,\ldots,N \,, i = 1,\ldots,m \qquad (13)$$

Therefore the optimization problem has 60 parameters. It was observed that the DE algorithm works much better solving the optimal control problem with state constraints instead of working with the original outputs head fresh weight and nitrate concentration. For that reason, using the desired values of both head fresh weight (y_{dfm}) and nitrate concentration (y_{dNO_3}), the desired values of the states at harvest time ($\mathbf{x}_d$) were calculated. The next step consists of the selection of the design parameters (population size, mutation and crossover constant) in the differential evolution algorithm. Roughly speaking, using a greater population size (μ) the algorithm has more chance to convergence to a global optimum at the expense of more computation time. A population size around the dimension of the optimization problem is a good starting point but, sometimes, smaller values are enough to get good results. Greater values of the mutation constant (F) make it possible to explore the whole search space and to prevent premature convergence. Crossover constant (CR) values around one speed up the convergence of the algorithm. So, a compromise has to be established among these three parameter values in the DE algorithm. With respect to the penalty functions, several options were tested. However, better results were obtained by using varying penalty coefficients ($\lambda(g)$), which were changed exponentially according to the generation number. They were used together with the absolute difference between the desired and calculated state values for the function $dist(x(t_f)$. This approach has been applied in a similar manner by Smith and Stonier (1996) in other evolutionary algorithms.

After some experiments with several values of the population size it was observed that even with a relatively small value for the population size ($\mu = 20$), F=0.5, and CR=0.2 a good solution with a performance index of J*=317.0987 mol PAR $m^{-2}s^{-1}$ was obtained. The penalty coefficients changed exponentially from $\lambda(0) = 50$ to $\lambda(g_{\max}) = 100$. The number of generations was 5000. The deviations from the desired outputs values were 0.2130 grams for fresh weight, and 0.4272 ppm for nitrate concentration. Thus, the final constraints are almost satisfied using relatively small values for the penalty coefficients. Figure 4 presents the optimal trajectories of head fresh weight and nitrate content calculated where J*=314.6248. The deviations from the final desired outputs were 1.2418 grams and 9.9060 ppm. The number of generations was 10000. Figure 5 presents the corresponding optimal controls. The shape of the sub-optimal trajectories calculated by the DE algorithm are different from those obtained by a gradient method but clearly they show a trend that resembles the controls presented in figure 3.

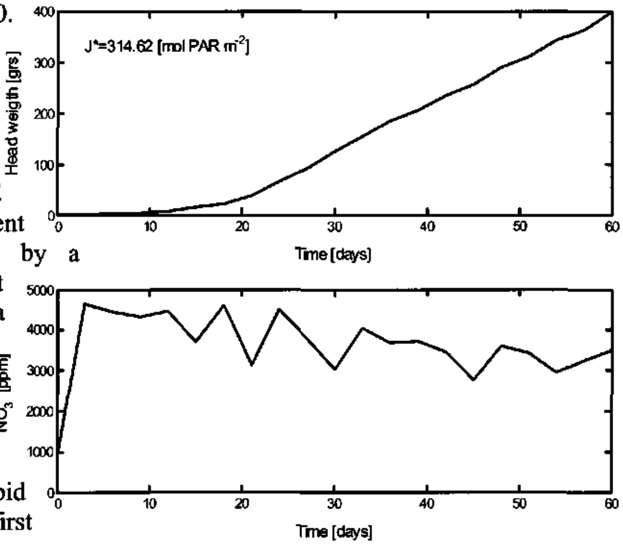


Figure 4. Near-optimal trajectories of head fresh weight and nitrate concentration calculated by the DE algorithm.

Looking at the rapid change during the first days of the optimal trajectory of light and temperature, calculated by the gradient method, it is clear that the piecewise constant control parameterization used by the DE algorithm is not able to approximate the continuous-time solution accurately. As a result the optimal performance found by the classical method is better than that found by the DE algorithm, which is only, near optimal.

On the other hand, as opposed to the classical algorithm, the differential evolution algorithm potentially finds the global solution. Therefore, efficient evolutionary algorithms as DE can be used to come up with an initial guess for classical algorithms, to prevent them from finding local minima. By increasing the number of time intervals or by specifying them as variable-length the solution of the DE algorithm will more closely approximate the continuous-time solution obtained by the classical algorithm. Finally, the rapid speed of convergence of the

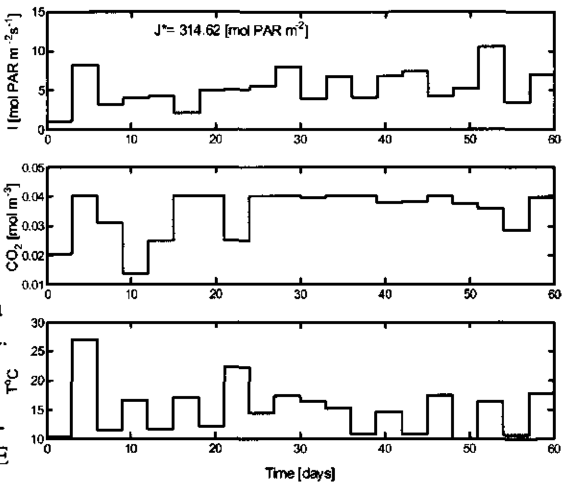


Figure 5. Near-optimal control inputs of light, carbon dioxide and temperature calculated by the DE algorithm.

differential evolution algorithm near the optimal solution is illustrated in figure 6. This appealing characteristic could be exploited to generate the initial guess for a local optimization method.

5.2.6 Conclusions

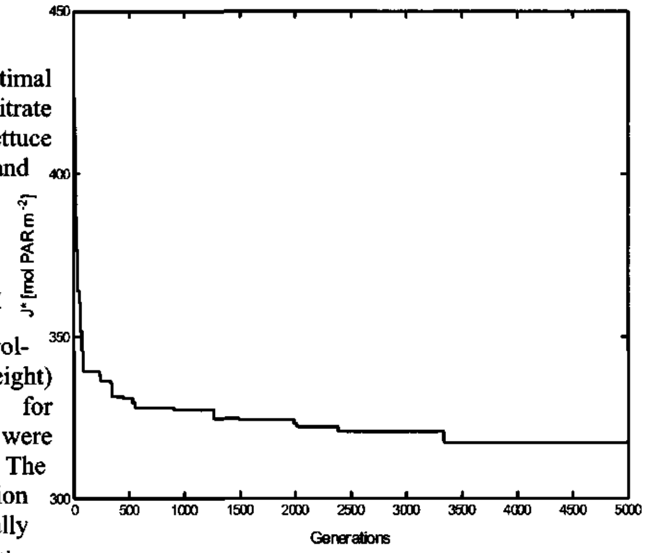


Figure 6. Convergence of the Differential Evolution algorithm.

Through the optimal control of nitrate concentration in lettuce the benefits and drawbacks of a Differential Evolution algorithm (DE algorithm) and a classical ACW (Adjustable Control-variation Weight) gradient algorithm for optimal control were demonstrated. The Differential Evolution algorithm potentially finds the global solution whereas the classical algorithm does not. On the other hand, the classical algorithm is able to find the continuous-time solution and is more efficient, even though compared to many other evolutionary algorithms, the DE algorithm is highly efficient. Therefore, taken together, i.e. using the DE algorithm to compute an initial guess for the classical algorithm, an algorithm can be obtained that combines the advantages of both approaches.

5.2.7 References

Bryson A. E. Jr., *Dynamic Optimization*, Addison Wesley, Menlo Park, 1999.

Ioslovich I, Seginer I., Acceptable nitrate concentration of greenhouse lettuce and optimal control policy for temperature plant spacing and nitrate supply, Preprints Agricontrol 2000, International conference on Modelling and control in agriculture, horticulture and post-harvested processing, July 10-12, 2000, Wageningen, The Netherlands, 89-94.

Seginer, I., Buwalda, F., Van Straten G., Nitrate concentration in greenhouse lettuce: A modeling study, *Acta Horticulturae* 456: 189-197, 1998.

Smith S., Stonier R., Applying Evolution Program Techniques to Constrained Continuous Optimal Control Problems, *Proceedings of the IEEE Conference on Evolutionary-Computation*, 1996, IEEE, Piscataway, NJ, USA, 285-290.

Stigter, J.D., Van Straten, G. Nitrate control of leafy vegetables: a classical dynamic optimization approach, Preprints Agricontrol 2000, International conference on Modelling and control in agriculture, horticulture and post-harvested processing, July 10-12, 2000, Wageningen, The Netherlands, 95-99.

Storn R. and Price K., Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, *Journal of Global Optimization* 11:341-359, 1997.

Weinreb A., Optimal control with multiple bounded inputs, PhD Thesis, Stanford University, 1985.

Yarkoni, N., McKenna, P. NICOLET Simulation model B3, Agricultural Engineering Department, Technion-Israel Institute of Technology, July, 2000.

5.3 Optimal control of nitrate in lettuce by a hybrid approach: differential evolution and ACW gradient algorithms[†]

5.3.1 Abstract

Since high concentration levels of nitrate in lettuce and other leafy vegetables are undesirable, cultivation of lettuce according to specified governmental regulations is currently an important issue. Therefore, methods are sought in order to produce a lettuce crop that allow maximization of the profits of the grower while at the same time insuring the quality of the crops. Using a two-state dynamic lettuce model that predicts the amount of nitrate at harvest time, an optimal control problem with terminal constraints is formulated. The situation considered may be relevant in a plant factory where a fixed head weight should be reached in fixed time while minimizing light input. First, optimal trajectories of light, CO_2 and temperature are calculated using the Adjustable Control Weight (ACW) gradient method. Subsequently, novel, efficient and modified Differential Evolution (DE) algorithms are used to obtain an approximate solution to the same optimal control problem. While the gradient method yields a more accurate result, the optimum may be local. In order to exploit the salient characteristics of a Differential Evolution algorithm as a global direct search method a hybrid-combined approach is proposed. An approximate solution obtained with a Differential Evolution algorithm is used to initialize the ACW gradient method. Although local minima did not seem to occur in this particular case, the results show the feasibility of this approach.

Keywords: Optimal control, Artificial Intelligence, Differential Evolution, Global optimization, Gradient method, Lettuce growth, Nitrate content

5.3.2 Introduction

High concentration levels of nitrate in a lettuce crop and other leafy vegetables are undesirable because they have a negative effect on human health. Therefore, methods are needed to control nitrate levels of a greenhouse lettuce crop as long as a profitable amount of head fresh weight is produced. In work associated to the European project NICOLET (Nitrate Control in Lettuce and other leafy vegetables), a model of lettuce growth, which predicts the amount of nitrate content at harvest time, has been proposed (Seginer et al., 1998, 1999). Also, an optimal control problem that considers temperature, nitrate supply and plant density as control variables, has been formulated and some properties of its solution have been analyzed using a reduced (one state variable) lettuce model (Ioslovich and Seginer, 2000). A two-state nonlinear lettuce model (original NICOLET model) has been used to calculate a numerical solution to another optimal control problem that uses light and temperature as control inputs by means of a first order gradient algorithm (Stigter and Van Straten, 2000). In the present study, a solution is presented for a new optimal control problem where the purpose is to produce a fixed final head weight in fixed final time, with a fixed final

[†] Accepted as publication by the Journal Computers and Electronics in Agriculture, special issue on Artificial Intelligence in Agriculture

nitrate level according to the standard, against minimal cost for artificial lighting. Control inputs are light, temperature and carbon dioxide concentration. A two-state lettuce model is used. Combinations of two different algorithms are applied in order to calculate a numerical solution. The Adjustable Control Weight (ACW) gradient algorithm (Weinreb 1985, Weinreb and Bryson 1985) is used first to obtain a solution. The recently proposed Differential Evolution (DE) algorithms (Storn and Price, 1997, Storn, 1999) are extended and applied to solve the optimal control problem of nitrates in lettuce.

The reason to investigate the potential application of evolutionary algorithms to dynamic optimization problems is that Differential Evolution algorithms constitute global direct search methods which have shown appealing convergence characteristics to solve multimodal optimal control problems (Lopez Cruz et al., 2002). Since gradient methods (like ACW algorithm) are mainly local dynamic optimization methods, in this work a hybrid method is proposed and applied in which a Differential Evolution algorithm provides an approximate solution by which the ACW gradient algorithm is initialized. A comparison of the solutions reached by the ACW algorithm and the hybrid approach is presented. This paper is organized as follows. In section 5.3.3 the main properties of the NICOLET (Nitrate Concentration in Lettuce) model are explained. Section 5.3.4 presents a description of the optimal control problem of nitrates in lettuce. Sections 5.3.5 and 5.3.6 describe the ACW and DE algorithms. Finally in section 5.3.7 the results are presented and discussed.

5.3.3 A dynamic model to predict nitrate concentration in lettuce crop

A dynamic model that accurately predicts, the nitrate content of a lettuce crop has recently been developed (Seginer et al., 1998, 1999). The so-called NICOLET model (Seginer et al., 1998) has two state variables: carbon content in the vacuoles (M_{Cv} mol[C] m^{-2} [ground]) and carbon content in the structure (M_{Cs} mol[C] m^{-2} [ground]). It represents a carbon source-sink relationship in the plant driven by sunlight, temperature and carbon dioxide concentration. The core of the model is given by the following two differential equations:

$$\dot{M}_{Cv} = F_{Cav} - h_g F_{Cm} - F_{Cg} - F_{Cvs} \tag{1}$$

$$\dot{M}_{Cs} = F_{Cvs} - (1 - h_g) F_{Cm} \tag{2}$$

which represent carbon balances of the vacuoles and structure that prevail in the plant cells. The terms F_{Cav}, F_{Cm}, F_{Cg} and F_{Cvs} (eqns. A1-A4 in appendix A) denote the rates of photosynthesis, maintenance respiration, growth respiration and growth, respectively. They are functions of the states (M_{Cv}, M_{Cs}) and the driving variables light (I [mol PAR m^{-2}s^{-1}]), carbon dioxide (C_{Ca} [mol m^{-3}]) and temperature (T °C). It is assumed that photosynthesis depends on light and carbon dioxide concentration but not on temperature, whereas growth and respiration hinges on temperature only. All the functions are given in appendix A. Appendix C shows all the values of the parameters of NICOLET B3 model.

According to the model, both photosynthesis and growth can proceed uninhibited as long as the non-structural carbon concentration in the vacuoles (C_{Cv}, eqn. A11) remains within certain limits. However, when due to environmental conditions the non-structural carbon concentration approaches zero growth will be reduced. This is described by a smooth switching function ($h_g(C_{Cv})$, eqn. A10) which is equal to one for non-inhibiting levels, but falls off to zero rapidly when the assimilate stock in the vacuoles becomes empty. When carbon assimilates in the vacuoles are too high a similar switching function ($h_p(C_{Cv})$, eqn. A9) brings photosynthesis to a halt.

The original NICOLET model has been modified slightly to deal more properly with the inhibition of growth and photosynthesis (McKenna, 2000). The original growth and photosynthesis inhibition functions did not completely avoid growth when the level of carbon material in the vacuoles is very small, thus leading to negative values of carbon content in the vacuoles when the lettuce is exposed to a long period of darkness. The new growth inhibition function $h_g(C_{Cv})$ (eqn. A10) complies with the condition $\lim_{C_{Cv}\to 0} h_g(C_{Cv}) = 0$. Likewise, the new photosynthesis inhibition function $h_p(C_{Cv})$ (eqn. A9) fulfills the condition $\lim_{C_{Cv}\to \Pi v/\gamma} h_p(C_{Cv}) = 0$. As can be seen from the term h_g in equations (1) and (2), the lettuce model used in this study includes the depletion of structural matter in order to meet the requirements of maintenance respiration when the carbon concentration in the vacuoles is low ($h_g(C_{Cv}) < 1$).

An important characteristic of the NICOLET model is that there exists a negative correlation between nitrate and sugar content in the crop. This means that when carbohydrates are low the plant responds by accumulating more nitrates and vice versa. By this mechanism the plant can maintain its turgor since both nitrates as well as carbohydrates are osmotically active components. Accordingly, the nitrate concentration in the vacuoles is calculated from an algebraic relationship with the carbon concentration and osmotic pressure in the vacuoles (eqn. A12). The difference between osmotic pressure in the vacuoles and rhizosphere is defined as the desired turgor pressure (eqn. A13). Also it is assumed that there is no limitation of nutrients supply, especially of nitrates to the crop. In a further modification of the original model this condition has been relaxed (Seginer et al. 1999). Calculation of other output variables such as dry and fresh matter and sugars is done by additional algebraic relationships that involve the states and additional parameters (eqn. A14-A19).

5.3.4 **Optimal control of levels of nitrates in lettuce crop**

An important consequence of the turgor maintenance hypothesis behind the NICOLET model is that the control of nitrate concentration can be done by manipulation of the shoot environment, namely, by increasing the carbon dioxide concentration (C_{Ca} [mol m^{-3}]) or the light and/or by lowering temperature (T °C). In this paper a situation is studied where the market demands fixed desired head fresh weight at a fixed harvest date. Consequently, one can formulate an optimal control problem with fixed final time and terminal constraints (Bryson 1999, Stigter and Van

Straten, 2000). Details concerning this optimal control problem formulation are presented in Appendix B.

Since the supply of artificial light to the greenhouse would be associated with an increasing cost in supplied energy, it is reasonable to try to produce a desired lettuce head with the lowest possible light. Therefore, a reasonable performance index would be the integral of artificial light

$$J = \int_0^{t_f} I(t)dt \tag{3}$$

which has to be minimized by a suitable control in such a way that a desired fresh head weight of lettuce (y_{dfm} [grams]) and a specified amount of nitrate (y_{dNO_3} [ppm]) are obtained at the given harvest time (t_f [days]) i.e.,

$$y_{fm}(t_f) = y_{dfm} \tag{4}$$

$$y_{NO_3}(t_f) = y_{dNO_3} \tag{5}$$

where $y_{fm}(t_f)$ and $y_{NO_3}(t_f)$ are the corresponding predicted fresh head weight and the predicted amount of nitrate from the model. Equations (4) and (5) represent here terminal state constraints. Furthermore, the control inputs are bounded because it is apparent that the light intensity cannot take negative values and the same is true for carbon dioxide. Also the temperature must lie within a domain in which no harm is done to the lettuce crop. Therefore the optimal control problem has some hard bounds according to:

$$I_{\min} \le I(t) \le I_{\max},\ C_{\min} \le C_{Ca}(t) \le C_{\max},\ T_{\min} \le T(t) \le T_{\max} \text{ for } 0 \le t \le t_f \tag{6}$$

where *min* and *max* indicate the lower and upper limits for the control inputs. This optimal control problem corresponds to a situation one may find in a plant factory environment, which is based on artificial lighting (Nakayama, 1991). Also it is easy to see that the previous optimal control problem can be changed in one in which we would desire to minimize *additional* artificial light. That would correspond to a plant factory situation in which sunlight is incorporated (Takatsuji, 1991). In contrast to the formulation of the optimal control of nitrate in lettuce given by Ioslovich and Seginer (2000), in our study neither nitrate supply nor plant density are considered as control variables. Plant density was kept constant in our calculations.

5.3.5 The Adjustable Control Weight gradient algorithm

Basically, the Adjustable Control-variation Weight (Weinreb, 1985, Weinreb and Bryson, 1985) gradient method modifies a classical first order gradient algorithm for solving optimal control problems with fixed final time and terminal constraints, by making K_u in the equation

$$\delta u(t) = -K_u \cdot H_u^{*T}(t) \tag{7}$$

dependent on the control, in order to properly deal with the control bounds $-1 \le u_i \le 1$, $i = 1,...,m$. Here K_u is a diagonal matrix with elements $k_u(u_i) \ge 0$. $H_u^*(t)$ is the derivative of the Hamiltonian to the controls $u(t)$ and $\delta u(t)$ is the adjustment of the control at each step of the iteration. In order to avoid loss of controllability $k_u(u_i)$ is decreased in the vicinity of both bounds independently of the sign of $H^*_{u_i}$ by means of,

$$k_u(u_i) = 1 - |u_i| \tag{8}$$

during the backward integration of the co-state equations. To prevent that the algorithm generates no optimal control at the bounds during the forward integration the following function is used

$$k_u(u_i) = \begin{cases} 1 - u_{th} \ \ if \cdot |u_i| \geq u_{th}, \mathrm{sgn}(u_i H^*_{u_i}) = +1 \\ 1 - |u_i| \qquad otherwise \end{cases} \tag{9}$$

where u_{th} is a design value close to $+1$. Because of those modifications the ACW gradient algorithm convergences generally slow to an optimal solution. For a more extensive description of the ACW gradient algorithm the reader is referred to Weinreb (1985).

5.3.6 Differential Evolution algorithms

A direct method based on evolutionary algorithms is used to approximate the global optimal solution. In this direct method the control trajectory is parameterised as a sequence of piece-wise constant values, which have to be found by the optimisation algorithm. Differential Evolution algorithms are evolutionary algorithms that have recently been proposed for the solution of static parameter optimisation problems (Storn and Price, 1997, Storn, 1999). These algorithms have several advantages over other evolutionary algorithms because they are easy to understand and very efficient computationally. An outline of these algorithms is presented in figure 1. As in other evolutionary algorithms the main operators in Differential Evolution are mutation, crossover and selection. However, in contrast to archetype genetic algorithms, they use a floating-point representation for the solutions. Also the main operator in DE is rather different than in other evolutionary algorithms. Similar to Evolution Strategies here each chromosome is represented as a real parameter vector $\mathbf{a} = [a_1, ..., a_q]$, and it is required that at generation g there is a population containing μ individuals $\mathbf{a}_i; i = 1, ..., \mu$. The essential feature of differential evolution algorithms rests on the mutation operator. A so-called mutant vector ($\mathbf{v}_i$), is generated by adding the weighted difference of two or four randomly selected vectors from the population, to another (to be mutated, $\mathbf{a}_{r_1}$) vector:

$$\mathbf{v}_i = \mathbf{a}_{r_1} + F \times (\mathbf{a}_{r_2} - \mathbf{a}_{r_3}) \, ; i = 1, ..., \mu \tag{10}$$

where ($r_1 \neq r_2 \neq r_3 \neq i$) are mutually different indexes. $F \in [0,2]$ is a factor that controls the amplification of the differential variation. Other schemes for mutation found in literature (Fisher et al. 1999, Lee et al., 1999) are:

$$\mathbf{v}_i = \mathbf{a}_{best} + F \times (\mathbf{a}_{r_1} + \mathbf{a}_{r_2} - \mathbf{a}_{r_3} - \mathbf{a}_{r_4}) \tag{11}$$

$$\mathbf{v}_i = \mathbf{a}_{best} + F \times (\mathbf{a}_{r_1} - \mathbf{a}_{r_2}) \tag{12}$$

$$\mathbf{v}_i = \mathbf{a}_i + F \times (\mathbf{a}_{r_1} - \mathbf{a}_{r_2}) \tag{13}$$

$$\mathbf{v}_i = \mathbf{a}_i + F \times (\mathbf{a}_{best} + \mathbf{a}_{r_1} - \mathbf{a}_i - \mathbf{a}_{r_2}) \tag{14}$$

where $\mathbf{a}_{best}$ is the best individual in the current population and $\mathbf{a}_i$ denotes a target individual (see below).

The crossover operator increases the diversity of the mutated vector by means of the combination of two solutions (mutant ($\mathbf{v}_i$) and target ($\mathbf{a}'_i$) vectors):

$$a'_{ji} = \begin{cases} v_{ji} \text{ } if \text{ } randb \leq CR \text{ } or \text{ } j = randr \\ a_{ji} \text{ } if \text{ } randb > CR \text{ } and \text{ } j \neq randr \end{cases}; \; i = 1,\ldots,\mu; j = 1,\ldots,q \tag{15}$$

where $\mathbf{v}_{ji}$ is the j-th element of the mutated vector $\mathbf{v}_i$, $\mathbf{a}_i$ is a so-called target vector, against which each new solution is compared to, and *randb* is a uniform random number from [0,1]. $randr \in [1,2,\ldots,q]$ is a randomly chosen index, and *CR* is a parameter $\in [0,1]$, which specifies the crossover constant. Selection is implemented only on the basis of a comparison between the cost function value of the new solution $\mathbf{a}'_i$ and that of the target vector $\mathbf{a}_i$. This means that if $J(\mathbf{a}'_i) \leq J(\mathbf{a}_i)$ is true, the new solution ($\mathbf{a}'_i$) becomes a member of the population at generation $g+1$, otherwise the old solution ($\mathbf{a}_i$) is retained. The inner loop in figure 1 implies that there are μ competitions at each generation since each member of the population plays the role of a target vector once.

Figure 1. *Differential Evolution algorithm*

```
Generate random solutions covering the given space.
Evaluate each solution.
g=1;
while (convergence is not reached)
   for i=1 to Population Size
       Apply differential mutation.
       Execute differential crossover.
       Clip the new solution if necessary.
       Evaluate the new solution.
       Apply differential selection.
   end
   g=g+1;
end
```

As one wants to apply DE algorithms to optimal control problems with control bounds, it is necessary to introduce a modification in the original algorithm in order to avoid the generation of inadmissible solutions. So a clipping technique can be added as follows:

$$u_j = \begin{cases} u_j = \alpha_j \text{ } if \text{ } u_j < \alpha_j \\ u_j = \beta_j \text{ } if \text{ } u_j > \beta_j \end{cases}, j = 1,\ldots,q \tag{16}$$

where α and β represent the lower and upper boundary of the control variables, respectively. Also in order to solve optimal control problems with terminal constraints a second extension is required. A solution can be computed by using penalty functions that enforce the desired values for the states. An augmented performance index is given by

$$J' = J + \lambda(g) dist(\mathbf{x}(t_f)) \tag{17}$$

where $\lambda(g)$ is either a penalty factor depending on the current generation or a constant penalty factor, J is given by equation (3), and two options for *dist* are:

$$dist(\mathbf{x}(t_f)) = \begin{cases} |\mathbf{x}_d - \mathbf{x}(t_f)| \\ [\mathbf{x}_d - \mathbf{x}(t_f)]^T [\mathbf{x}_d - \mathbf{x}(t_f)] \end{cases} \tag{18}$$

5.3.7. Results

5.3.7.1 *A solution obtained by a gradient algorithm*

Our implementation of the ACW follows Bryson's algorithm for solving optimal control problems with fixed final time and terminal constraints (Bryson, 1999). Thus, a variable step-size solver was used (ode45.m in Matlab) to integrate the dynamic equations and to solve a continuous-time optimal control problem. The co-state equations were calculated from analytical expressions of $\partial f / \partial x$ and $\partial f / \partial u$. The final time was specified at $t_f = 60$ days. The constraints at the harvest time were $y_{dfm} = 400$ grams of head fresh weight, and $y_{dNO_3} = 3500$ ppm of nitrate concentration. The bounds on the controls were as follows: $0 \leq I(t) \leq 17.5$ [mol PAR m^{-2}], $0.01 \leq C_{Ca}(t) \leq 0.04$ [mol m^{-3}], and $10 \leq T(t) \leq 30$ [$^{\circ}$C], respectively. The upper limit for light is somewhat arbitrary but it is not relevant since we want to reduce the use of artificial light as much as possible. The upper limit for CO2 is determined by workability conditions in the greenhouse. The bounds for temperature reflect the range for growth of lettuce without any stress.

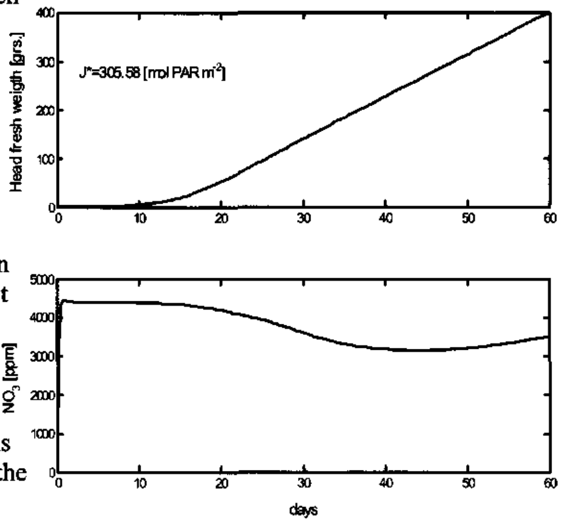


Figure 2. Optimal trajectories of head fresh weight and nitrate concentration, showing the terminal constraints are satisfied.

The ACW algorithm was run using several different constant initial values for the controls. Every time a similar cost function value was obtained which means probably that this problem is not multimodal. For one of the optimizations with a performance index value of $J^* = 305.58$ [mol PAR m^{-2}] figure 2 shows the calculated optimal trajectories of fresh head weight of lettuce as well as the optimal trajectory of nitrate content obtained after 3000 iterations of the ACW gradient algorithm. The optimal solution satisfies almost exactly the two constraints at harvest time. The error for fresh head weight was 0.01 grams, and 0.11 ppm in case of nitrate concentration.

Figure 3 presents the optimal controls: light, carbon dioxide and temperature. Since one goal of the optimal control problem formulation was the minimization of the integral of light, the calculated optimal solution shows that actually it is possible to control the nitrate levels at harvest time by increasing the supply of carbon dioxide and decreasing the temperature. This result confirms that with artificial light it is possible to control nitrate levels in lettuce through the control of the shoot environment (Seginer et al., 1998). The optimal trajectory of carbon dioxide supply is at the upper specified limit (0.04 mol m^{-3}), which is consistent with the fact that no cost was associated to it in the formulation of the optimal control problem.

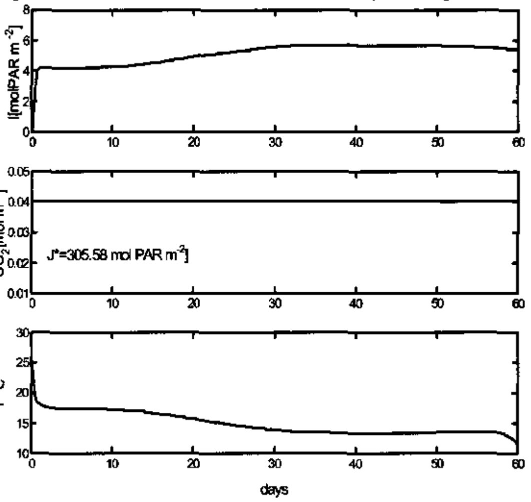


Figure 3. Optimal control inputs of light, carbon dioxide, and temperature.

Even though starting with a different initial constant temperature trajectory the algorithm calculates a slightly different optimal trajectory, a trend is fairly clear that temperature goes down across the cultivation period, as expected. This can be explained by the fact that a high temperature during the early stages of lettuce stimulates growth, especially as long as the ground is not completely covered. Directly at the start the optimal reaches fairly high levels, because obviously enough light must be supplied to produce photosynthetic activity. Next, for the rest of the growing period, the amount of light is increased but not too much in order to meet the specification of the performance index and also to come up to the desired fresh head weight and the nitrate content at harvest time. It can be noticed that during the last 30 days the amount of light is increased a bit in order to decrease the concentration of nitrate.

5.3.7.2 *Generating an initial solution to the ACW method by means of Differential Evolution algorithms*

In order to solve the previous optimal control problem by differential evolution algorithms, a reasonably accurate parameterization for the control inputs must be selected that keeps the number of parameters to be optimized as small as possible. A piece-wise constant approximation for the three controls (*m=3*) was chosen, over twenty time intervals (*N=20*).

$$\mathbf{u}(t) \cong \mathbf{u}(\mathrm{t_k}) = [u_k^1 \quad u_k^2 \quad \dots \quad u_k^m]^T \; t \in [t_k, t_{k+1})\,; k = 1,\dots,N\,, \qquad (19)$$

Therefore the optimization problem has 60 parameters. It was found that DE algorithms worked much better with state constraints instead of constraints on the original outputs head fresh weight and nitrate concentration. For that reason, using the desired values of both head fresh weight ($y_{dfm} = 400$ grams) and nitrate concentration ($y_{dNO_3} = 3500$ ppm), the desired values of the states at harvest time $x_1(t_f) = M_{Cv}(t_f) = 2.4866$ [mol C m^{-2} ground s^{-1}] and $x_2(t_f) = M_{Cs}(t_f) = 8.1840$ [mol C m^{-2} ground s^{-1}] were calculated. This was done by solving numerically a system of two simultaneous algebraic equations derived from equations A14 and A18 for M_{Cv} and M_{Cs}. The next step consists of the selection of the design parameters, population size (μ), mutation (F), and crossover (CR) constants in the differential evolution algorithms. Roughly speaking, using a greater population size DE has more chance to converge to a global optimum at the expense of more computation time. A population size around the dimension of the optimization problem is a good starting point but, sometimes, smaller values are enough to get good results. Greater values of the mutation constant (F) make it possible to explore the whole search space and to prevent premature convergence. Crossover constant (CR) values around one speed up the convergence of the algorithm. Therefore, a compromise has to be established among these three parameter values in the DE algorithms. With respect to the penalty functions, several options were tested. However, better results were obtained by using varying penalty coefficients ($\lambda(g)$), which were changed exponentially according to the generation number. They were used together with the absolute difference between the desired and calculated state values for the function $dist(x(t_f))$. Smith and Stonier (1996) have applied this approach in a similar manner in other evolutionary algorithms.

Although we have done experiments using in total five Differential Evolution algorithms described in section 5.3.6, from now on we report results obtained with the DE in which the mutation operator was given by equation (11). After some experiments with several values of the population size it was observed that even with a relatively small value for the population size ($\mu = 20$), and almost standard values for mutation constant F=0.5, and crossover constant CR=0.2 (Storn and Price 1997) a good solution with a performance index of J*=317.0987 mol PAR m^{-2} was obtained. The penalty coefficients changed exponentially from $\lambda(0) = 50$ to $\lambda(g_{\max}) = 100$. The number of generations was 5000. The deviations from the desired outputs values were 0.2130 grams for fresh weight, and 0.4272 ppm for nitrate concentration. Therefore, the final constraints are almost satisfied using relatively small values for the penalty coefficients. Increasing the number of generations to 10000 a performance index of J*=314.6248 mol PAR m^{-2} was obtained. The deviations from the final desired outputs were 1.2418 grams and 9.9060 ppm.

Looking at the optimal trajectory of light and temperature, calculated by the ACW gradient method (section 5.3.5.), it is clear that the piecewise constant control parameterization used by the DE algorithm is unable to approximate the continuous-time solution accurately. Consequently, the optimal performance found by the classical method is better than that found by the DE algorithm, which is only near optimal. Also the shapes of the optimal trajectories were different from those obtained by a classical approach (Lopez Cruz et al, 2001).

The use of a hybrid approach that uses the solution found by the DE to initialize ACW is motivated by its potential to locate the global optimum. First the DE algorithm is applied in order to increase the chances to get a near global optimum solution. The algorithm is stopped after a reasonable number of iterations, for instance when no considerable improvement is observed. Then the ACW algorithm is used to refine the solution provided by DE. Since by using DE the search space is reduced we would expect that the local method converges faster than when it is applied alone. Initializing ACW by the DE solution is possible since the implemented ACW gradient algorithm allows a start with even a small number of controls, because of the variable step size integration and interpolation (Bryson, 1999). The computation time of the hybrid DE/ACW approach exceeds that of ACW algorithm alone. However, for a multimodal optimal control problem the additional computational load would be justified by the increased chance of finding the global optimum of the problem.

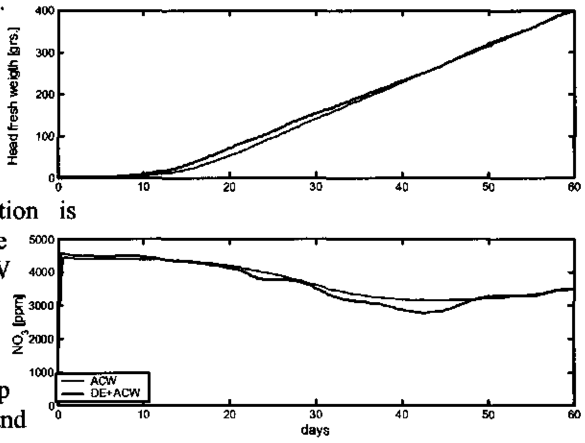
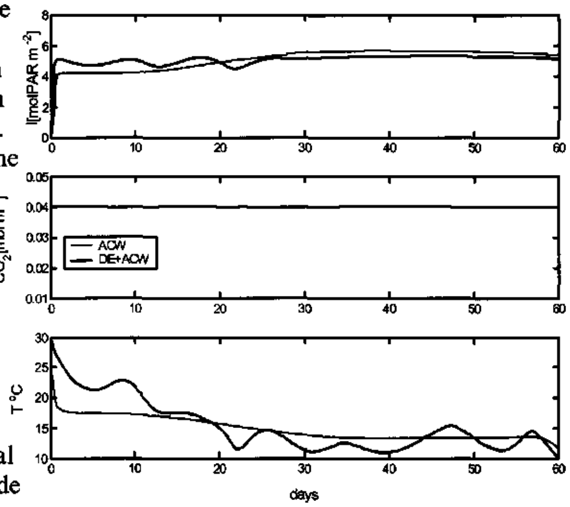


Figure 4. Optimal trajectories of head fresh weight and nitrate concentration calculated by a hybrid algorithm.

Figure 4 shows the optimal trajectories calculated by a hybrid method. The Differential Evolution algorithm was run 2000 iterations and then ACW was run 2000 more iterations. The trajectories obtained by the ACW algorithm alone with 3000 iterations are shown as well to make a comparison. Both terminal constraints were satisfied accurately, since deviations from the target were 0.01 grams for head fresh weight and 0.34 ppm in case of nitrate concentration. Only small differences can be observed between both optimal state trajectories. Also the values of the performance index were quite similar: J*=305.58 mol PAR m^{-2} for ACW and J*=305.65 mol PAR m^{-2} in case of the hybrid algorithm (DE+ACW).



The optimal control trajectories are shown in figure 5. The optimal trajectories of carbon dioxide are exactly the same, apart from some oscillations; the

Figure 5. Optimal trajectories of light, carbon dioxide and temperature calculated by a hybrid algorithm.

optimal trajectories of light are also similar. However, the trajectory for temperature calculated by the hybrid approach shows considerable oscillations even though the general decreasing trend is similar. These differences can be explained by a likely lack of sensitivity of the cost criterion to changes of the temperature trajectory as long as on average they are identical. Table 1 illustrates the idea that a global optimization method as a first step is useful to reduce the number of steps for a gradient method.

Table 1. Results of ACW and a hybrid (DE+ACW) method in solving OCP of nitrates in lettuce.

Method	Values	
ACW	J*	307.86
	Iterations	1000.00
	$y_{fm}(t_f)$	399.99
	$y_{nc}(t_f)$	3500.18
DE (1000 iterations) + ACW	J*	308.54
	Iterations	1000.00
	$y_{fm}(t_f)$	399.99
	$y_{nc}(t_f)$	3500.42
DE (2000 iterations) + ACW	J*	307.85
	Iterations	272.00
	$y_{fm}(t_f)$	399.99
	$y_{nc}(t_f)$	3500.56
DE (5000 iterations) + ACW	J*	307.83
	Iterations	250.00
	$y_{fm}(t_f)$	399.99
	$y_{nc}(t_f)$	3499.96

The ACW method was run 1000 iterations only and the value of the associated cost was used as reference for our comparisons. Next, the DE algorithm was run five times for three different numbers of generations (1000, 2000, and 5000). Using the best of the solutions (those that satisfied the terminal constraints best) to initialize the ACW algorithm the optimization problem was solved combining both approaches. The ACW algorithm was stopped when it reached the same goal function value. Table 1 shows, in fact, that by using 2000 and 5000 iterations of the DE and then ACW, there is a considerable reduction in the number of iterations of the local optimization method to calculate the same solution. These results clearly show that by using DE algorithm to initialize ACW is not only possible to achieve similar solutions than those attained by ACW alone but also that the initialization with DE significantly reduces the time ACW takes to converge.

5.3.8 Discussion

According to our results it is likely that the selected optimal control problem is not multimodal. Although the same costs were obtained when different initial values for the controls were chosen, some differences were observed in the shape of the optimal trajectory of temperature. These differences were greater when the ACW gradient algorithm was initialized using a non-smooth trajectory generated by the DE algorithm. Apparently, the gradient method 'sticks' to another smooth solution pattern after initializing with DE than in the original full problem. However, the optimal trajectories of light and CO_2 were always similar. This means that, the optimal trajectory of temperature and so the solution is not unique. Therefore, the optimal control problem is redundant. The redundancy constitutes a difficult situation for any search method. It would be preferable to obtain a problem formulation without redundancy but this formulation seems difficult to achieve without making further assumptions on the lettuce model, by simplifying the model (cf. Ioslovich and Seginer 2000) or by selecting a different cost function. In addition, the non-uniqueness feature of the solution is arguable from a mathematical point of view, but it is quite acceptable from an engineering viewpoint while the same minimal cost is found.

Most significant is the fact that for each of the patterns found, the optimal trajectory of temperature showed a decreasing trend over the cultivation period. This result was in accordance with one of the hypothesis of the lettuce model that states possibility of control levels of nitrates by lowering temperature. In general, the optimal control trajectories of light, carbon dioxide and temperature were in agreement with the predictions of the lettuce model and have confirmed some possibilities for control of nitrate concentration when the shoot environment and artificial lighting are considered. Nevertheless, further insight in the model is required to account for the behavior observed in case of optimal trajectories of temperature.

Generally, highly multimodal optimal control problems are very difficult to solve by gradient-based optimization methods since they require an initial control rather close to the global optimum otherwise are easily trapped by any local solution (Chalabi, 1994, Lopez Cruz et al, 2002). Although the advantages of using a Differential Evolution algorithm most clearly stand out in multimodal problems, the current case clearly demonstrates the feasibility of the application of DE algorithms in solving optimal control problems in agriculture.

5.3.9 Conclusions

Through the optimal control of nitrate concentration in lettuce the feasibility of using a Differential Evolution algorithm to provide an initial optimal control trajectory for a classical ACW (Adjustable Control-variation Weight) gradient algorithm was demonstrated. Although in the current case no local optima were found, the Differential Evolution algorithm potentially finds the global optimal solution whereas the classical algorithm alone does not. On the other hand, the classical algorithm is able to find an accurate solution of a continuous-time optimal control problem. Therefore, taken together, i.e. using the Differential Evolution algorithms to compute an initial feasible estimate of the optimal controls for the classical algorithm, an hybrid algorithm is obtained that combines the advantages of both approaches.

Acknowledgements

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

5.3.10 References

Bryson A.E. Jr., Ho Y.Ch., 1975. Applied Optimal Control, Hemisphere Publishing

Corporation, NY.

Bryson A. E. Jr., Dynamic Optimization, Addison Wesley, Menlo Park, 1999.

Chalabi Z. S., 1994. Optimal control methods for agricultural systems. In Chalabi Z. S. and DayW. (Eds.) Proceedings of the second IFAC/ISHS workshop on mathematical and control applications in agriculture and horticulture, Silsoe UK, 12-15, September 1994, pp. 221-227.

Fisher, M.M., Hlaváčková-Schindler K., Reismann M., A global search procedure for parameter estimation in neural spatial interaction modelling, *Papers in Regional Science 78*, 1999, 119-134.

Ioslovich I, Seginer I., Acceptable nitrate concentration of greenhouse lettuce and optimal control policy for temperature plant spacing and nitrate supply, Preprints Agricontrol 2000, International conference on Modelling and control in agriculture, horticulture and post-harvested processing, July 10-12, 2000, Wageningen, The Netherlands, 89-94.

Lee, M.H., Ch. Han, Ch., Chang, K.S., Dynamic Optimization of a Continuous Polymer Reactor Using a Modified Differential Evolution Algorithm, *Ind. Eng. Chem. Res. 38*, 1999, 4825-4831.

Lopez Cruz I.L., van Willigenburg L.G., van Straten G., Optimal control of nitrate in lettuce by gradient and differential evolution algorithms, IFAC/CIGR 4 International Workshop on Artificial Intelligence in Agriculture, June 6-8, 2001, Budapest, Hungary, 123-128.

Lopez Cruz I.L., van Willigenburg L.G., van Straten G., 2002. Efficient Differential Evolution algorithms for multimodal optimal control problems, Internal Report. Systems and Control Group, Wageningen University, The Netherlands.

Nakayama, S., Plant factory and its prospects, Mathematical and Control Applications in Agriculture and Horticulture (FAC Workshop Series No.1), 1991, 85-92.

McKenna, P., Summary of minor additions to original NICOLET model B3, Agricultural Engineering Department, Technion-Israel Institute of Technology, 2000.

Price, K., 1999, An Introduction to Differential Evolution. In Corne D., Dorigo, M. and Glover F., New Ideas in Optimization, Mc GrawHill, NY.

Seginer, I., Buwalda, F., Van Straten G., Nitrate concentration in greenhouse lettuce: A modeling study, Acta Horticulturae 456, 1998, 189-197.

Seginer, I., Van Straten, G., Buwalda F., Lettuce growth limited by nitrate supply, Acta Horticulturae 507, 1999, 141-148.

Smith S., Stonier R., Applying Evolution Program Techniques to Constrained Continuous Optimal Control Problems, Proceedings of the IEEE Conference on Evolutionary-Computation, IEEE, Piscataway, NJ, USA, 1996, 285-290.

Stigter, J.D., Van Straten, G., Nitrate control of leafy vegetables: a classical dynamic optimization approach, Preprints Agricontrol 2000, International conference on Modelling and control in agriculture, horticulture and post-harvested processing, July 10-12, 2000, Wageningen, The Netherlands, 95-99.

Storn R., System design by Constraint Adaptation and Differential Evolution, IEEE Transactions on Evolutionary Computation 3, (1), 1999, 22-34.

Storn R. and Price K., 1997. Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization 11, 341-359.

Takatsuji M. ,1991. Fundamental study of plant factories, Mathematical and Control Applications in Agriculture and Horticulture (FAC Workshop Series No.1), 81-84.

Weinreb A., 1985. Optimal control with multiple bounded inputs, PhD Thesis, Stanford University.

Weinreb A. Bryson A.E. 1985. Optimal control of systems with hard control bounds. IEEE Trans. Autom. Control, AC-30, 1135-1138.

Yarkoni, N., McKenna, P. 2000. NICOLET Simulation model B3, Agricultural Engineering Department, Technion-Israel Institute of Technology.

Appendix A. Nicolet B3 model equations.

$$\dot{M}_{Cv} = F_{Cav} - h_g F_{Cm} - F_{Cg} - F_{Cvs} \tag{1}$$

$$\dot{M}_{Cs} = F_{Cvs} - (1-h_g)F_{Cm} \tag{2}$$

$$F_{Cav} = p\{I, C_{Ca}\} f\{M_{Cs}\} h_p\{C_{Cv}\} \tag{A1}$$

$$F_{Cm} = M_{Cs} e\{T\} \tag{A2}$$

$$F_{Cg} = \theta F_{Cvs} \tag{A3}$$

$$F_{Cvs} = g\{T\} f\{M_{Cs}\} h_g\{C_{Cv}\} \tag{A4}$$

$$f\{M_{Cs}\} = 1 - e^{(-a \cdot M_{Cs})} \tag{A5}$$

$$p\{I, C_{Ca}\} = \frac{\varepsilon I \sigma (C_{Ca} - C_c^*)}{\varepsilon I + \sigma (C_{Ca} - C_c^*)} \tag{A6}$$

$$e\{T\} = k \cdot e^{c(T-T^*)} \tag{A7}$$

$$g\{T\} = v \cdot e\{T\} \tag{A8}$$

$$h_p\{C_{Cv}\} = \frac{1}{1 + ((1-b_p)\Pi_v / (\Pi_v - \gamma C_{Cv}))^{s_p}} \tag{A9}$$

$$h_g\{C_{Cv}\} = \frac{1}{1 + (b_g \Pi_v / (\gamma \cdot C_{Cv}))^{s_g}} \tag{A10}$$

$$C_{Cv} = \frac{M_{Cv}}{\lambda \cdot M_{Cs}} \tag{A11}$$

$$\beta C_{Nv} + \gamma C_{Cv} = \Pi_v \tag{A12}$$

$$\Pi_v - \Pi_r = P_v \tag{A13}$$

$$Y_{fm} = \frac{1000}{pldens} M_{fm} \tag{A14}$$

$$M_{fm} = 1000 \cdot \lambda \cdot M_{Cs} + M_{dm} \tag{A15}$$

$$M_{dm} = \eta_{OMC}(M_{Cv} + M_{Cs}) + \eta_{MMN}(\frac{\lambda \Pi_v}{\beta} M_{Cs} - \frac{\gamma}{\beta} M_{Cv}) \tag{A16}$$

$$C_{NO3} = 10^6 \cdot \eta_{NO3N} \cdot C_{NO3N} \tag{A17}$$

$$C_{NO3} = C_{Nv} \times (1 - DFR)/1000 \tag{A18}$$

$$DFR = M_{dm} / M_{fm} \tag{A19}$$

Appendix B. The Hamiltonian function and its derivative

Using a state space representation the optimal control problem presented in section 3 can be generically described as follows:
Let equations (1) and (2) be defined as

$$\dot{x} = f(x,u),\ x(t_0) \tag{B1}$$

where $x(t)$ and $u(t)$ represent an n-dimensional state vector and m-dimensional control vector respectively. The cost function (3) is given by

$$J = \phi(x(t_f)) + \int_{t_0}^{t_f} L(x,u)dt \tag{B2}$$

where $L(x,u) = I(t)$ and $\phi(x(t_f))$ is not defined in our particular problem. Equations (4) and (5) are denoted by

$$\psi(x(t_f)) = 0 \tag{B3}$$

Now, according to optimal control theory, the Hamiltonian function can be written as follows:

$$H(t) = L(x,u) + \lambda^T(t) f(x,u) \tag{B4}$$

where $\lambda^T = -\frac{\partial H}{\partial x}$ are the co-states $\lambda_{M_{Cv}}$ and $\lambda_{M_{Cs}}$ associated to the states. The derivative of the Hamiltonian function (Bryson and Ho, 1975, page 49) is

$$\dot{H} = \frac{\partial L}{\partial t} + \lambda^T \frac{\partial f}{\partial t} + \frac{\partial H}{\partial u}\dot{u} \tag{B5}$$

The dynamic equations (B1) as well as the function L do not depend explicitly on time t. And at the stationary condition $\frac{\partial H}{\partial u} = 0$ is obtained. Therefore $\dot{H} = 0$, and $H(x^*, \lambda^*, u^*) = c$ for $0 \le t \le t_f$, where c is a constant (Bryson and Ho, 1975, Bryson, 1999).

Appendix C. Nicolet's model parameter values used during the optimizations.

Parameter	Value	Unit
a	1.7	m^2 [ground] mol^{-1}[C]
ε	0.04-0.07	mol[C]/mol [PAP]
σ	1.4e-3	ms^{-1}

c	0.0693	^{O}C
k	0.25e-6	s^{-1}
γ	0.61	$m^3Pa/mol[C]$
θ	0.3	dimensionless
ν	13	mol[C] m^{-2} [ground]
λ	1/1200	$m^3mol[C]^{-1}$
β	6	m^3Pa/mol [N]
T*	20	^{O}C
C_c^*	0.0011	mol[C] m^{-3}
b_p	0.8	dimensionless
b_g	0.2	dimensionless
s_p	10	dimensionless
s_g	10	dimensionless
η_{OMC}	0.03	Kg [dw]/mol[C]
η_{MMN}	0.148	Kg [dw]/mol [N]
pldens	18	Plants m^{-2}

6. General discussion and conclusions

6.1 General discussion

The aim of this work was to evaluate the performance of efficient Evolutionary Algorithms on optimal control problems. Within the class of Evolutionary Algorithms several approaches based on Genetic Algorithms, either with binary or floating-point representation, have been applied in the past. However, fundamental limitations have been found which show clearly that Genetic Algorithms are not good candidates to solve optimal control problems efficiently [1, 2]. Nowadays, it is well known that GAs can solve optimally only separable optimisation problems through applying the strategy 'one-parameter-at-time' and small mutations ($p_m < 1$). In this case the required computational complexity is $O(n \ln(n))$ [1, 2] where n is the dimension of the optimisation. When this strategy is applied to non-separable functions this complexity grows to $O(\exp(n \ln n))$ which is higher than that required by random search [2]. Crossover operators speed up the convergence without modifying this complexity. Therefore, many Evolutionary Algorithms based on classical GAs, which apply small mutations and high crossover probabilities, are not suitable to solve optimal control problems efficiently. This is confirmed by the results of this thesis.

Price [3] has pointed out that should a mutation mechanism be efficient enough to be capable of removing the drawbacks of GAs, it should have the following properties: *i*) use a zero-mean distribution for the generation of mutation vectors, *ii*) use dynamic scaling of the distribution to suit each variable and *iii*) use correlated mutations to ensure rotational invariance. It appears that Evolution Strategies, Evolutionary Programming and Differential Evolution algorithms fulfil these requirements.

Evolution Strategies and Evolutionary Programming explicitly use a Gaussian distribution with zero mean and standard deviation one, thereby satisfying property *i*. Also they implement self-adaptation for scaling and orienting a mutation vector. This implements property *ii*. To implement property *iii* ES use a strategy matrix that adds rotation angles to the scale factors. This increases the computational complexity of ES to $O(n^2)$ [3]. In practice the use of rotation angles can be avoided. Then the demanded computational complexity is only $O(n)$. But in this case, probably, ES are dealing worse with functions having interacting variables.

Differential Evolution algorithms are a family of evolutionary algorithms having the properties *i-iii* while having only $O(n)$ computational complexity. Firstly, DE guarantees a distribution with zero mean because random sampling from the population guarantees that a difference vector $\vec{x}_{r_1} - \vec{x}_{r_2}$ occurs as often as its opposite $\vec{x}_{r_2} - \vec{x}_{r_1}$. Secondly, DE scales mutation step sizes to suit each variable by sampling difference vectors, which present a scale comparable to each variable's interval of improvement, from a population of them. Thirdly, DE is rotational invariant since the mutation distribution generated by difference vectors will always have the same orientation as the contour lines of the objective function. In addition, it seems that DE algorithms have a property that Price has called a 'universal global mutation mechanism' or 'globally correlated mutation mechanism', which seems to be the main property responsible for the appealing performance of DE as global optimisers. Additional features of DE algorithms are described in Price [3].

The previous discussion justifies the focus of this thesis on Differential Evolution algorithms. Nonetheless, chapter 2.2 studied the potential application of the GEnetic algorithm for Numerical Optimization for COnstrained Problems (GENOCOP) to optimal control problems. This algorithm has been reported in the literature as being relatively efficient. Basically, GENOCOP uses a floating-point representation of the solutions and some specialized genetic operators. Several researchers have reported advantages of this algorithm over classical GAs and also over the Simulated Annealing algorithm [4, 5]. It is easy to see that, in addition to the benefits given by the floating-point representation, the use of a combination of several mutation and crossover operators considerably increases the performance of evolutionary algorithms similar to GENOCOP. Typically, the same mutation operators are used several times at each generation [5]. In chapter 2.2 the selected frequencies were: uniform mutation 4, non-uniform mutation 4, multi-non-uniform mutation 6 and boundary mutation 4. Given the way all GENOCOP's mutation operators work we would expect a low mutation probability, and accordingly, a computational complexity similar to that demanded by the Breeder Genetic algorithm. This issue deserves further investigation. For several applications the use of more than one mutation operator, like in GENOCOP, was reported to account for an improved algorithm performance. It can be expected that the introduction of the multi-non uniform mutation operator, which is applied to all the variables of a chromosome will have an additional beneficial effect, since in that situation the probability of mutation is $p_m \approx 1$. The number of function evaluations required by GENOCOP to obtain the solutions reported in chapter 2.2 was: 80,000-180,000 for the first problem, 180,000-300,000 for the second and 120,000 for the third. Since the dimension of the optimisation problem was low, serious limitations are expected when this dimension increases. In addition, in contrast to Differential Evolution, GENOCOP is a more complicated evolutionary algorithm with several parameters that need to be tuned. Despite this, with the proposed improvements (use of multi-non uniform mutation), it is worthwhile to investigate whether GENOCOP can compare with DE algorithms.

In chapter 3 Differential Evolutionary algorithms were investigated in more detail. A number of optimal control problems that are difficult to solve, or are unsolvable by using classical methods, were selected to illustrate several advantages of DE algorithms. The results indicate a relationship between the multi-modality of the problem and the proper choice of the algorithm parameters crossover constant (CR) and the mutation factor (F). Additional work on other multi-modal optimal control problems (especially those with a larger dimension) is needed to further confirm this. The possibility, suggested by Price [3], to solve multi-modal optimal control problems using small population sizes to achieve the highest possible speed of convergence, was investigated. This is especially important, because for optimal control problems, each objective function evaluation involves a system simulation. Our results confirmed this possibility.

Several of the resolved optimal control problems have bounds on the controls. Originally DE works only for unconstrained problems. Whenever unfeasible solutions were generated by the DE algorithm they were set to the limit they exceeded. Price has mentioned that this method probably has the effect of reducing the diversity of the population [3]. However, our experience was that this strategy worked well. A comparison of three possible techniques to deal with the control bounds is presented in table 1 for the case of the *DE/rand/1/bin* algorithm. The first technique (I) randomly selects a value from the interval $\alpha_i(t) \le u_i(t) \le \beta_i(t)$; $i = 1,..,m$ each time an unfeasible control value is generated [6]. The second technique (II) sets offending control values to the limit they exceed (see equation 14 in chapter 3). The third technique (III) is the one suggested by Price [3] which is described by the following equation:

$$a'_{ji} = \begin{cases} (a_{ji} + \alpha_j)/2 & if \quad a_{ji} < \alpha_j \\ (a_{ji} + \beta_j)/2 & if \quad a_{ji} > \beta_j \\ a_{ji} & otherwise \end{cases}, \; j = 1,2,...,d \,, ;i = 1,2,...,\mu \qquad (1)$$

where a_{ji} represents a variable after the application of mutation and crossover operators. The selected DE algorithm parameters were $\mu = 20$, $CR = 0.5$ and $F = 0.5$. Using each technique the algorithm was executed ten times. Almost the same solutions were obtained while our technique (II) required the least number of function evaluations.

Table 1. Averaged results of comparison of three methods for constrained control inputs in *DE/rand/1/bin* on CSTR multimodal optimal control problem

Strategy	I	II	III
Generations	244.5	220	221.3
J*	0.135586	0.135584	0.135584
CE (%)	100	100	100
Function evaluations	4890	4400	4426

Table 2 presents similar results which relate to the highly multi-modal optimal control problem of the tubular reactor, described in chapter 3. The population size was $\mu = 25$, the crossover constant $CR = 0$ and the mutation parameter $F = 0.9$. Again the algorithm was executed ten times.

Table 2. Averaged results of comparison of three methods for constrained control inputs in *DE/rand/1/bin* on the bifunctional catalyst blend optimal control problem

Strategy	I	II	III
Generations	235	126.9	194.75
J*	10.09184	10.09410	10.09308
CE (%)	100	100	80
Function evaluations	5875	3172.50	4868.75

Again our technique (II) required the least number of function evaluations and therefore seems to be preferable.

The results of chapter 3 regarding Breeder Genetic algorithms (BGA's) confirm that these algorithms are not good candidates to solve multi-modal optimal control problems efficiently. However, the use of sub-populations improves the efficiency of the BGA's. This option deserves further study.

The results regarding DE algorithms in chapter 3 confirm that these are more efficient than other Evolutionary Algorithms in solving optimal control problems. The efficiency of DE's turned out to be comparable with that of the Iterative Dynamic Programming (IDP) algorithm, which is especially designed for optimal control problems. Moreover, DE algorithms do not demand extensive preliminary experimentation with algorithm parameter values as is required by IDP. In spite of the efficiency already achieved with DE algorithms there still remains the challenge of further improving their efficiency.

In chapter 4 the problem of improving the efficiency of DE algorithms through automatic adjustment of the algorithm parameters was investigated. Price has recognized the fact that keeping the algorithm parameters in DE constant is not likely to be an optimal solution [3]. Undoubtedly, our strategy is only a first attempt and could be improved in the future. For instance, it still does not modify the population size. Its main idea is to incorporate information present in the population to select the appropriate values of the algorithm parameters mutation and crossover. Although the efficiency improved one would expect further improvement by designing new heuristic rules based on additional information on the population to modify the population size as well. This is an open field of research for the future.

To explore some potential practical applications of Differential Evolution algorithms to optimal control problems that appear in agriculture two main issues were addressed in chapters 5.2 and 5.3. Firstly, the *DE/best/2/bin* algorithm was applied to approximate a solution of the continuous-time optimal control problem of nitrate concentration in lettuce. A penalty function approach was applied to deal with terminal state constraints. General characteristics of optimal control problems in agriculture (i.e. greenhouse cultivation) are non-linearity and (in some cases) non-convexity, which can induce a multiplicity of solutions [7]. Although the optimal control problem described and solved in chapter 5.2 is highly non-linear, it seems that it is not multi-modal. Therefore a classical method can solve it more efficiently and accurately than a direct method like Differential Evolution. Nonetheless, there are two options for improvement of the results obtained by DE that might be studied in greater detail; a larger number of time intervals and variable-length time intervals. The latter constitutes optimisation of the piecewise constant control parameterisation as well. This will increase the dimension of the optimisation problem but as DEAs are efficient this should not present a major problem. Another issue that has not yet been investigated is the use of a more sophisticated parameterisation of the controls like linear piece-wise or cubic B-splines polynomials.

In chapter 5.3 a DE algorithm is used to approximate the global solution of an optimal control problem sufficiently close after which a local gradient-based search method is used to obtain it accurately and efficiently. Unfortunately the particular optimal control problem under consideration turned out not to be multimodal, while this approach is specifically promising for multi-modal optimal control problems. Still the results indicate clearly the further improvement of the efficiency of this global optimal control algorithm. These types of algorithms are also an important area for future research.

6.2 Conclusions

The feasibility and possible advantages of applying Evolutionary algorithms to solve complex optimal control problems being high dimensional, multi-modal and non-differentiable were investigated. This study revealed that DE algorithms are significantly more efficient than other Genetic Algorithms, such as Breeder Genetic Algorithms (BGA), when applied to multi-modal optimal control problems since they do not share several theoretical and practical limitations that other Genetic Algorithms have. The efficiency of DE is comparable to the efficiency of Iterative Dynamic Programming (IDP), a global optimisation approach specifically designed for optimal control. Moreover the DE algorithms turned out to be significantly less sensitive to problems concerning the selection or tuning of algorithm parameters and the initialisation of the algorithm.

Although it is not a DE algorithm, the GENOCOP algorithm is considered to be one of the most efficient genetic algorithms with real-valued individuals and specialized evolutionary operators. This algorithm was the starting point of our research. In Chapter 2 it was applied to some optimal control problems from chemical engineering. These problems were high dimensional, non-linear, multivariable, multi-modal and non-differentiable. Basically with GENOCOP the same solutions were obtained as with Iterative Dynamic Programming. Moreover GENOCOP is more successful in locating the global solution in comparison with for instance IDP and other local optimisation algorithms. GENOCOP'S efficiency however is rather poor and the algorithm parameter tuning rather complicated. This motivated us to seek for more efficient evolutionary algorithms.

Mathematical arguments found in the literature state that DE algorithms outperform other Genetic algorithms in terms of computational efficiency. Therefore in chapter 3, DE algorithms, generally used to solve continuous parameter optimisation problems, were used to solve two multi-modal (benchmark) optimal control problems. Also some Breeder Genetic Algorithms (BGA) were applied to solve these problems. The results obtained with these algorithms were compared to one another, and to the results obtained with IDP. The comparison confirmed that DE algorithms stand out in terms of efficiency as compared to the Breeder Genetic algorithms. Moreover, in contrast to the majority of Evolutionary Algorithms which have many algorithm parameters that need to be selected or tuned, DE has only three algorithm parameters that have to be selected or tuned. These are the population size (μ), the crossover constant (CR) and the differential variation amplification (F). The population size plays a crucial role in solving multi-modal optimal control problems. Selecting a smaller population size enhances the computational efficiency but reduces the probability of finding the global solution. During our investigations we tried to find the best trade-off. One of the most efficient DE algorithms is denoted by *DE/best/2/bin*. All the investigated DE algorithms solved the two benchmark multi-modal optimal control problems properly and efficiently. The computational efficiency achieved by the DE algorithms in solving the first low multi-modal problem, was comparable to that of IDP. When applied to the second, highly multi-modal problem, the computational efficiency of DE was slightly inferior to the one of IDP, after tuning of the algorithm parameters. However, the selection or tuning of the algorithm parameters for IDP is more difficult and more involved.

From our investigation the following guidelines were obtained for the selection of the DE algorithm parameters. Take the population size less than or equal to two times the number of variables to be optimised that result from the control parameterisation of the original optimal control problem ($\mu \leq 2n_u$). Highly multi-modal optimal control problems require a large value of the differential variation amplification ($F \geq 0.9$) and a very small or zero value for the crossover constant ($0 \leq CR \leq 0.2$). Low multi-modal optimal control problems need a medium value for the differential variation amplification ($0.4 \leq F \leq 0.6$) and a large or medium value for the crossover constant ($0.2 \leq CR \leq 0.5$). In contrast to IDP, finding near-optimal values for the algorithm parameters is very simple for DE algorithms.

Generally, the DE algorithm parameters are kept constant during the optimization process. A more effective and efficient algorithm may be obtained if they are adjusted on-line. In Chapter 4, a strategy that adjusts the differential variation amplification (F) and the crossover constant (CR) on-line using a measure of the diversity of the individuals in the population, was proposed. Roughly, the proposed strategy takes large values for F and small values for CR at the beginning of the optimization in order to promote a global search. When the population approaches the solution, F is decreased in order to promote a local search, and the crossover parameter CR is enlarged to increase the speed of convergence. When implemented on the DE algorithm *DE/rand/1/bin* and applied to the two benchmark multi-modal optimal control problems, the computational efficiency significantly improved and also the probability of locating the global solution.

To judge the opportunities and advantages of using Genetic Algorithms to solve problems related to optimal control, in chapter 5 several engineering applications concerning optimal greenhouse cultivation control were considered. In Chapter 5.1 genetic algorithms with binary individuals (Simple Genetic algorithm) and floating-point representations (GENOCOP) are used to estimate some of the parameters of a two-state dynamic model of a lettuce crop, the so-called NICOLET model. This model is intended to predict dry weight and nitrate content of lettuce at harvest time. Parameter estimation problems usually suffer from local minima. This study showed that Genetic Algorithms are suitable to calibrate the parameters of a dynamic model. However the required computation time is significant. Partly this is due to the high computational load of a single function evaluation which for parameter optimisation problems involves a system simulation. Even though parameter optimisation is very often performed off-line, thus making computation time perhaps less important, more efficient evolutionary algorithms like DE are to be preferred.

In chapter 5.2 an optimal control problem of nitrate concentration in a lettuce crop was solved by means of two different algorithms. The ACW (Adjustable Control-variation Weight) gradient algorithm which searches for local solutions and the DE algorithm *DE/best/2/bin* that searches for a global solution. The dynamic model is a modified two-state dynamic model of a lettuce crop (NICOLET B3) and the control problem has a fixed final time and control and terminal state constraints. The DE algorithm was extended in order to deal with this. The results showed that this problem probably does not have local solutions and that the control parameterisation required by the DE algorithm causes some difficulties in accurately approximating the continuous solution obtained by the ACW algorithm. On the other hand the computational efficiency of the evolutionary algorithm turned out to be impressive. An almost natural conclusion therefore is to combine a DE algorithm with a gradient algorithm.

In chapter 5.3 the combination of a DE algorithm and a first order gradient algorithm is used to solve an optimal control problem. The DE algorithm is used to approximate the global solution sufficiently close after which the gradient algorithm can converge to it efficiently. This approach was successfully tried on the optimal control of nitrate in lettuce, which unfortunately in this case, seems to have no local solutions. Still the feasibility of this approach, which is especially interesting for multi-modal optimal control problems, was clearly demonstrated.

6.3 References

[1] Muhlenbein H., Schlierkamp-Voosen D., Predictive Models for the Breeder Genetic Algorithm, I. Continuous Parameter Optimization, *Evolutionary Computation* 1 (1), 1993, 25-50.

[2] Salomon R., Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms, *BioSystems* 39 (1996) 263-278.

[3] Price K. V. An Introduction to Differential Evolution in Corne D., Dorigo, M. and Glover F., New Ideas in Optimization, Mc GrawHill, 1999.

[4] Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs, Third, revised and extended edition, Springer-Verlag, 1996.

[5] Houck, C., Joines, J.A., Kay, M.G ., A genetic algorithm to function optimization: A MATLAB implementation, NCSU-IE TR 95-09, 1995.

[6] Lampinen J., Solving problems subject to multiple nonlinear constraints by differential evolution, in Radek Matoušek and Pavel Ošmera (eds.) Proceedings of MENDEL 2001, 7th International conference on Soft Computing, June 6-8, 2001, Brno, Czech Republic.

[7] Chalabi Z. S., Optimal control methods for agricultural systems, in Chalabi Z. S. and DayW. (editors) Proceedings of the second IFAC/ISHS workshop on mathematical and control applications in agriculture and horticulture, Silsoe UK, 12-15 september 1994, pp. 221-227.

[8] Storn R. and Price K., Differential Evolution –A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, *Journal of Global Optimization* 11:341-359, 1997.

Summary

If optimal control problems are solved by means of gradient based local search methods, convergence to local solutions is likely. Recently, there has been an increasing interest in the use of global optimisation algorithms to solve optimal control problems, which are expected to have local solutions. Evolutionary Algorithms (EAs) are global optimisation algorithms that have mainly been applied to solve static optimisation problems. Only rarely Evolutionary Algorithms have been used to solve optimal control problems. This may be due to the belief that their computational efficiency is insufficient to solve this type of problems. In addition, the application of Evolutionary Algorithms is a relatively young area of research. As demonstrated in this thesis, Evolutionary Algorithms exist which have significant advantages over other global optimisation methods for optimal control, while their efficiency is comparable.

The purpose of this study was to investigate and search for efficient evolutionary algorithms to solve optimal control problems that are expected to have local solutions. These optimal control problems are called multi-modal. An important additional requirement for the practical application of these algorithms is that they preferably should not require any algorithm parameter tuning. Therefore algorithms with less algorithm parameters should be preferred. In addition guidelines for the choice of algorithm parameter values, and the possible development of *automatic* algorithm parameter adjustment strategies, are important issues.

This study revealed that Differential Evolution (DE) algorithms are a class of evolutionary algorithms that do not share several theoretical and practical limitations that other Genetic Algorithms have. As a result they are significantly more efficient than other Genetic Algorithms, such as Breeder Genetic Algorithms (BGA), when applied to multi-modal optimal control problems. Their efficiency is comparable to the efficiency of Iterative Dynamic Programming (IDP), a global optimisation approach specifically designed for optimal control. Moreover the DE algorithms turned out to be significantly less sensitive to problems concerning the selection or tuning of algorithm parameters and the initialisation of the algorithm.

Although it is not a DE algorithm, the GENOCOP algorithm is considered to be one of the most efficient genetic algorithms with real-valued individuals and specialized evolutionary operators. This algorithm was the starting point of our research. In Chapter 2 it was applied to some optimal control problems from chemical engineering. These problems were high dimensional, non-linear, multivariable, multi-modal and non-differentiable. Basically with GENOCOP the same solutions were obtained as with Iterative Dynamic Programming. Moreover GENOCOP is more successful in locating the global solution in comparison with other local optimisation algorithms. GENOCOP'S efficiency however is rather poor and the algorithm parameter tuning rather complicated. This motivated us to seek for more efficient evolutionary algorithms.

Mathematical arguments found in the literature state that DE algorithms outperform other Evolutionary Algorithms in terms of computational efficiency. Therefore in Chapter 3, DE algorithms, generally used to solve continuous parameter optimisation

problems, were used to solve two multi-modal (benchmark) optimal control problems. Also some Breeder Genetic Algorithms (BGA) were applied to solve these problems. The results obtained with these algorithms were compared to one another, and to the results obtained with IDP. The comparison confirmed that DE algorithms stand out in terms of efficiency as compared to the Breeder Genetic algorithms. Moreover, in contrast to the majority of Evolutionary Algorithms, which have many algorithm parameters that need to be selected or tuned, DE has only three algorithm parameters that have to be selected or tuned. These are the population size (μ), the crossover constant (CR) and the differential variation amplification (F). The population size plays a crucial role in solving multi-modal optimal control problems. Selecting a smaller population size enhances the computational efficiency but reduces the probability of finding the global solution. During our investigations we tried to find the best trade-off. One of the most efficient DE algorithms is denoted by *DE/best/2/bin*. All the investigated DE algorithms solved the two benchmark multi-modal optimal control problems properly and efficiently. The computational efficiency achieved by the DE algorithms in solving the first low multi-modal problem, was comparable to that of IDP. When applied to the second highly multi-modal problem, the computational efficiency of DE was slightly inferior to the one of IDP, after tuning of the algorithm parameters. However, the selection or tuning of the algorithm parameters for IDP is more difficult and more involved.

From our investigation the following guidelines were obtained for the selection of the DE algorithm parameters. Take the population size less than or equal to two times the number of variables to be optimised that result from the control parameterisation of the original optimal control problem ($\mu \le 2n_u$). Highly multi-modal optimal control problems require a large value of the differential variation amplification ($F \ge 0.9$) and a very small or zero value for the crossover constant ($0 \le CR \le 0.2$). Low multi-modal optimal control problems need a medium value for the differential variation amplification ($0.4 \le F \le 0.6$) and a large or medium value for the crossover constant ($0.2 \le CR \le 0.5$). In contrast to IDP, finding near-optimal values for the algorithm parameters is very simple for DE algorithms.

Generally, the DE algorithm parameters are kept constant during the optimization process. A more effective and efficient algorithm may be obtained if they are adjusted on-line. In Chapter 4, a strategy that on-line adjusts the differential variation amplification (F) and the crossover constant (CR) using a measure of the diversity of the individuals in the population, was proposed. Roughly, the proposed strategy takes large values for F and small values for CR at the beginning of the optimization in order to promote a global search. When the population approaches the solution, F is decreased in order to promote a local search, and the crossover parameter CR is enlarged to increase the speed of convergence. When implemented on the DE algorithm *DE/rand/1/bin* and applied to the two benchmark multi-modal optimal control problems, the computational efficiency significantly improved and also the probability of locating the global solution.

To judge the opportunities and advantages of using Evolutionary Algorithms to solve problems related to optimal control, in Chapter 5 several engineering applications concerning optimal greenhouse cultivation control are considered. In Chapter 5.1 genetic algorithms with binary individuals (Simple Genetic Algorithm) and floating-point representation (GENOCOP) for the individuals are used to estimate some of the parameters of a two-state dynamic model of a lettuce crop, the so-called NICOLET model. This model is intended to predict dry weight and nitrate content of lettuce at harvest time. Parameter estimation problems usually suffer from local minima. This study showed that Evolutionary Algorithms are suitable to calibrate the parameters of a dynamic model. However the required computation time is significant. Partly this is due to the high computational load of a single objective function evaluation, which for parameter optimisation problems involves a system simulation. Even though parameter optimisation is very often performed off-line, thus making computation time perhaps less important, more efficient evolutionary algorithms like DE are to be preferred.

In Chapter 5.2 an optimal control problem of nitrate concentration in a lettuce crop was solved by means of two different algorithms. The ACW (Adjustable Control-variation Weight) gradient algorithm, which searches for local solutions, and the DE algorithm *DE/best/2/bin* that searches for a global solution. The dynamic system is a modified two-state dynamic model of a lettuce crop (NICOLET B3) and the control problem has a fixed final time and control and terminal state constraints. The DE algorithm was extended in order to deal with this. The results showed that this problem probably does not have local solutions and that the control parameterisation required by the DE algorithm causes some difficulties in accurately approximating the continuous solution obtained by the ACW algorithm. On the other hand the computational efficiency of the evolutionary algorithm turned out to be impressive. An almost natural conclusion therefore is to combine a DE algorithm with a gradient algorithm.

In Chapter 5.3 the combination of a DE algorithm and a first order gradient algorithm is used to solve an optimal control problem. The DE algorithm is used to approximate the global solution sufficiently close after which the gradient algorithm can converge to it efficiently. This approach was successfully tried on the optimal control of nitrate in lettuce, which unfortunately in this case, seems to have no local solutions. Still the feasibility of this approach, which is important for all types of optimal control problems of which it is unknown a-priori whether they have local solutions, was clearly demonstrated.

Finally, in Chapter six this thesis ends with an overall discussion, conclusions and suggestions for future research.

Samenvatting

Als optimale besturingsproblemen worden opgelost met locale gradiënt methodes is convergentie naar lokale oplossingen waarschijnlijk. Recentelijk is er een toenemende interesse voor globale optimalisatie algoritmen voor het oplossen van optimale besturingsproblemen waarvan de verwachting is dat ze lokale oplossingen hebben. Evolutionaire Algoritmen zijn globale optimalisatie algoritmen die tot nu toe voornamelijk zijn gebruikt voor het oplossen van statische optimalisatie problemen. In slechts enkele gevallen zijn dit soort algoritmen gebruikt voor het oplossen van optimale besturingsproblemen. Dit zou kunnen worden veroorzaakt door het geloof dat dit soort algoritmen niet efficiënt genoeg is voor het oplossen van deze categorie problemen. Daar komt bij dat Evolutionaire Algoritmen een betrekkelijk nieuw onderzoeksterrein vormen. In dit proefschrift is aangetoond dat er Evolutionaire Algoritmen bestaan die belangrijke voordelen hebben ten opzichte van andere globale optimalisatie methodes voor het oplossen van optimale besturingsproblemen, terwijl hun efficiency vergelijkbaar is.

Het doel van deze studie was het zoeken naar en onderzoeken van efficiënte Evolutionaire Algoritmen voor het oplossen van optimale besturingsproblemen waarvan verwacht wordt dat ze locale oplossingen bezitten. Dit soort optimale besturingsproblemen wordt in het Engels aangeduid met de term multi-modal. Een belangrijke additionele voorwaarde voor de praktische toepassing van dit soort algoritmen is dat ze bij voorkeur geen parameter aanpassing behoeven. Daarom verdienen algoritmen met weinig parameters de voorkeur. Bovendien zijn richtlijnen voor de keuze van algoritme parameters, en de mogelijke ontwikkeling van strategieën voor het *automatisch* aanpassen van algoritme parameters, van groot belang.

Deze studie heeft laten zien dat Differential Evolution (DE) algoritmen een klasse van Evolutionaire Algoritmen is zonder de praktische en theoretische bezwaren die andere evolutionaire (genetische) algoritmen hebben. Daarom zijn ze belangrijk meer efficiënt dan andere evolutionaire (genetische) algoritmen, zoals de Breeder Genetic Algoritmen (BGA), wanneer ze worden gebruikt voor het oplossen van optimale besturingsproblemen die locale minima bezitten. De efficiency is vergelijkbaar met de efficiency van Iteratief Dynamisch Programmeren (IDP), een globale optimalisatie methode speciaal ontworpen voor optimale besturingsproblemen. Bovendien blijken de DE algoritmen belangrijk minder gevoelig voor problemen die verband houden met het aanpassen van algoritme parameters en de initialisatie van de algoritmen.

Hoewel het geen DE algoritme is, wordt het zogenaamde GENOCOP algoritme beschouwd als een van de meest efficiënte Evolutionaire (genetische) Algoritmen met individuen gerepresenteerd door reële getallen en met specialistische evolutionaire operatoren. Dit algoritme vormde het startpunt voor dit onderzoek. In hoofdstuk 2 werd het toegepast op een aantal optimale besturingsproblemen uit de chemie. Deze problemen zijn hoog dimensionaal, niet lineair, multivariabel, multi-modal, en niet differentieerbaar. Met GENOCOP werden in essentie dezelfde oplossingen gevonden als met Iteratief Dynamisch Programmeren. Bovendien is GENOCOP beter in het vinden van het globale minimum in vergelijking met andere, locale optimalisatie algoritmen. De efficiency van GENOCOP is echter vrij pover en het aanpassen van de

algoritme parameters vrij ingewikkeld. Daarom hebben we gezocht naar meer efficiënte Evolutionaire Algoritmen.

Wiskundige argumenten, gevonden in de literatuur, stellen dat DE algoritmen beter zijn, voor wat betreft hun efficiency, dan andere Evolutionaire Algoritmen. DE algoritmes die in het algemeen worden gebruikt voor continue parameter optimalisatie problemen werden daarom in hoofdstuk 3 gebruikt voor het oplossen van optimale besturingproblemen met locale minima (benchmarks). Daarnaast werden deze problemen opgelost met een aantal BGA algoritmen. De resultaten werden met elkaar vergeleken en met die verkregen door middel van Iteratief Dynamisch Programmeren. Deze vergelijking bevestigde dat DE algoritmes efficiënter zijn dan de BGA algoritmen. Bovendien, in tegenstelling tot andere Evolutionaire Algoritmen welke een groot aantal parameters hebben die moeten worden aangepast, behoeven er in DE algoritmen slechts drie parameters te worden aangepast of gekozen. Dit zijn de populatie grootte (μ) de crossover (CR) en de differential variation amplification (F). De populatie grootte speelt een cruciale rol bij het oplossen van optimale besturingsproblemen. Een kleine populatiegrootte bevordert de efficiency van het algoritme maar vermindert de kans op het vinden van de globale oplossing. Tijdens het onderzoek hebben we gezocht naar het beste compromis. Eén van de meest efficiënte DE algoritmen wordt aangeduid met *DE/best/2/bin*. Alle onderzochte DE losten de twee optimale besturingsproblemen (benchmarks) op. De efficiency van het DE algoritme was vergelijkbaar met die van IDP voor het eerste probleem dat naast het globale slechts één locale oplossing had. Toegepast op het tweede probleem, welke een groot aantal locale minima bezat, was het DE algoritme iets minder efficiënt na het aanpassen van de parameters van beide algoritmen. Echter het aanpassen van de parameters in het IDP algoritme is moeilijker en vraagt meer tijd.

Ons onderzoek leverde de volgende richtlijnen op voor het selecteren van de drie DE algoritme parameters. Neem de populatiegrootte kleiner of gelijk aan twee keer het aantal te optimaliseren variabelen die voortvloeien uit de stuurparameterisatie van het optimale besturingsprobleem. ($\mu \leq 2n_u$). Problemen met veel locale minima vereisen een grote waarde van de differential variation amplification ($F \geq 0.9$) en een heel kleine waarde of een waarde nul voor de crossover constante ($0 \leq CR \leq 0.2$). Problemen met weinig locale minima vereisen een gemiddelde waarde van de differential variation amplification ($0.4 \leq F \leq 0.6$) en een hoge of gemiddelde waarde van de crossover constante ($0.2 \leq CR \leq 0.5$). In tegenstelling tot IDP is het vinden van goede waarden voor de algoritme parameters van een DE algoritme erg eenvoudig.

In het algemeen worden de DE algoritme parameters constant gehouden tijdens de optimalisatie. Een meer effectief en efficiënt algoritme kan worden verkregen als deze tijdens de optimalisatie worden aangepast. In hoofdstuk 4 werd daartoe een strategie ontwikkeld en toegepast die de differential variation amplification (F) en de crossover constant (CR) tijdens de optimalisatie aanpast, op grond van een maat voor de diversiteit van de populatie. Ruwweg worden in het begin grote waarden voor F gekozen en kleine voor CR. Als de populatie neigt naar de oplossing wordt F verlaagd om het locaal zoeken te bevorderen terwijl CR wordt verhoogd om de convergentiesnelheid op te voeren. Als deze strategie wordt geïmplementeerd neemt de efficiency significant toe en ook neemt de kans toe op het vinden van het globale minimum bij toepassing van het DE algoritme *DE/rand/1/bin* op de twee benchmarks.

Teneinde de mogelijkheden van het gebruik van Evolutionaire Algoritmen te beoordelen, voor het oplossen van optimale besturingsproblemen, werden in hoofdstuk 5 een aantal toepassingen die betrekking hebben op klimaat regeling in kassen beschouwd. In hoofdstuk 5.1 werden genetische algoritmen met binair gecodeerde individuen (Simple Genetic Algorithms) en algoritmen met een floating-point representatie (GENOCOP) van individuen toegepast voor het schatten van een aantal parameters van het dynamisch model met twee toestanden van een krop sla (NICOLET model). Dit model beschrijft het drooggewicht en de nitraat concentratie van de krop sla. Parameter optimalisatie problemen bezitten vaak locale minima. Het onderzoek toonde aan dat Evolutionaire Algoritmen geschikt zijn voor het kalibreren van de parameters van dit model. De rekentijd die dit vraagt is echter aanzienlijk. Dit is gedeeltelijk te wijten aan de grote rekentijd die een functie evaluatie vergt, omdat een functie evaluatie een simulatie vergt van het systeem. Hoewel parameter optimalisatie (kalibratie) meestal off-line plaatsvindt, en de daarvoor benodigde rekentijd derhalve niet kritisch is, zijn meer efficiënte Evolutionaire Algoritmen, zoals DE algoritmen, te prefereren.

In hoofdstuk 5.2 werd een optimaal besturingsprobleem aangaande de nitraat concentratie in sla opgelost met behulp van twee verschillende algoritmen. Het ACW (Adjustable Control Weight) gradiënt algoritme, welke locaal zoekt, en het DE algoritme *DE/best/2/bin* welke zoekt naar een globale oplossing. Het dynamisch model (NICOLET B3) heeft twee toestanden en is een gemodificeerde versie van het eerder beschreven model. Het optimale besturingsprobleem heeft een vrije eindtijd, een begrensde sturing en eindvoorwaarden. Het DE algoritme werd overeenkomstig uitgebreid. De resultaten tonen dat dit probleem waarschijnlijk geen locale oplossingen heeft. De stuurparameterisatie, nodig voor het toepassen van het DE algoritme, veroorzaakt lichte problemen bij het nauwkeurig benaderen van de continue oplossing die wordt gevonden door het ACW algoritme. Aan de andere kant toont dit probleem de indrukwekkende efficiency van het DE algoritme. Een voor de hand liggend idee is beide algoritmen te combineren.

Deze combinatie werd onderzocht in hoofdstuk 5.3. Een DE algoritme werd gebruikt voor het voldoende nauwkeurig benaderen van de globale oplossing waarna een gradiënt algoritme werd gebruikt voor efficiënte convergentie naar dit globale minimum. Deze aanpak is met succes toegepast op het optimale besturingsprobleem betreffende regeling van de nitraat concentratie in sla, welke ongelukkigerwijs, slechts één globaal minimum lijkt te bezitten. Niettemin is de toepasbaarheid van deze aanpak, welke van belang is voor alle optimale besturingsproblemen waarvan niet a-priori bekend is of ze locale minima bezitten, duidelijk aangetoond.

Tenslotte eindigde dit proefschrift in hoofdstuk 6 met een discussie, conclusies en aanbevelingen voor toekomstig onderzoek.

Resumen

Si problemas de control óptimo son resueltos mediante métodos locales de búsqueda basados en gradientes es muy probable su convergencia a soluciones locales. Recientemente ha aumentado el interés en la aplicación de algoritmos globales de optimización para resolver problemas de control óptimo que se espera posean soluciones locales. Los Algoritmos Evolutivos (AEs) son algoritmos globales de optimización que han sido usados principalmente para solucionar problemas de optimización estática. Raramente se han aplicado algoritmos evolutivos para resolver problemas de control óptimo debido a que se cree no son suficientemente eficientes para esta clase de problemas. Además la aplicación de algoritmos evolutivos es un área de investigación relativamente reciente. Como es demostrado en esta tesis, hay algoritmos evolutivos que tienen ventajas significativas sobre otros métodos globales de optimización mientras que su eficiencia es comparable.

El objetivo de este estudio fué investigar y buscar algoritmos evolutivos eficientes para resolver problemas de control óptimo que se espera tengan soluciones locales. Estos problemas de control óptimo son llamados multimodales. Un importante requerimiento adicional para la aplicación práctica de estos algoritmos es que no requieran algún ajuste de parámetros. Por eso deberan preferirse algoritmos con el menor número de parametros. Aparte de eso, la obtención de recomendaciones para la elección de los valores de los parámetros del algoritmo y el desarrollo de estrategias para un ajuste *automático* de los mismos, son cuestiones importantes.

Este estudio reveló que los algoritmos Evolución Diferencial (DE) son una clase de algoritmos evolutivos que no comparten las limitaciones teóricas y practicas que poseen los Algoritmos Genéticos. Como resultado son significativamente mas eficientes que otros Algoritmos Genéticos como los Algoritmos Genéticos Breeder (BGA) cuando son aplicados a problemas de control óptimo que tienen multiples soluciones. Su eficiencia es comparable a aquella de Programación Dinámica Iterativa (IDP) que es un enfoque global de optimización específicamente diseñado para problemas de control óptimo. Además los algoritmos DE resultaron ser menos sensibles en lo que se refiere a problemas de selección y ajuste de sus parámetros así como a su inicialización.

Aunque GENOCOP no es un algoritmo basado en Evolución Diferencial, es considerado como uno de los algoritmos genéticos mas eficientes que usa una representación de individuos como valores reales y operadores evolutivos especializados. Este algoritmo fué el punto de partida de nuestra investigación. En el Capítulo 2 este algoritmo fué aplicado a algunos problemas de control óptimo encontrados frecuentemente en ingenieria quimica. Estos problemas se caracterizan por ser altamente dimensionales, no lineales, multivariable, multimodal y no diferenciables. GENOCOP obtuvo las mismas soluciones que el algoritmo IDP. Aparte de eso GENOCOP es mas exitoso que otros algoritmos locales de optimización en encontrar el óptimo global. Sin embargo la eficiencia de GENOCOP es muy pobre y la afinación de sus parametros bastante complicada. Esta fué nuestra motivación para investigar otros algoritmos evolutivos mas eficientes.

Argumentos matemáticos encontrados en la literatura establecen que los algoritmos DE superan a otros algortimos evolutivos en cuanto a eficiencia computacional se refiere. Por lo tanto en el Capitulo 3 los algoritmos DE que generalmente se usan para resolver problemas de optimización de parametros continuos fueron usados para solucionar dos problemas de control óptimo multimodales (benchmark). Algunos algoritmos BGA fueron aplicados también. Los resultados obtenidos fueron comparados entre ellos y también con aquellos obtenidos con IDP. Esta comparación confirmó que los algoritmos DE destacan en terminos de eficiencia comparados con los algoritmos BGA. Mas aún, a diferencia de la mayoria de algoritmos evolutivos, los cuales tienen muchos parámetros que es necesario seleccionar o ajustar, DE tiene solamente tres parametros. Estos parámetros son el tamaño de la población (μ), la constante de cruzamiento (CR) y amplificación de la variación diferencial (F). El tamaño de la población tiene un papel crucial en la solución de problemas de control óptimo multimodales. Seleccionando una población pequeña se mejora la eficiencia computacional pero se reduce la probabilidad de encontrar la solución global. Durante nuestras investigaciones se intento la mejor compromiso. Uno de los algoritmos DE mas eficientes es nombrado *DE/best/2/bin*. Todos los algoritmos DE solucionaron los dos problemas de control óptimo multimodales debida y eficientemente. Su eficiencia computacional alcanzada en la solución del primer problema fue comparable a aquella de IDP. Cuando se aplicaron en el segundo problema la efficiencia resulto ser ligeramente inferior a la de IDP. Sin embargo la selección o ajuste de los parámetros en el algoritmo IDP es mas dificil y complicada.

Como resultado de nuestra investigación algunas normas fueron obtenidas para la selección de los parametros de los algoritmos DE. Seleccionar el tamaño de la población menor o igual al doble del número de variables a ser optimizadas que son el resultado de la parametrización del problema de control óptimo original ($\mu \leq 2n_u$). Problemas de control altamente multimodales parecen necesitar un valor grande del parámetro del operador de mutación ($F \geq 0.9$) y un valor muy pequeño o cero para la constante del operador de cruzamiento ($0 \leq CR \leq 0.2$). Problemas no altamente multimodales requieren un valor termino medio para el parámetro en el operador de mutación ($0.4 \leq F \leq 0.6$) y un valor pequeño o grande para la constante del operador de cruzamiento ($0.2 \leq CR \leq 0.5$). A diferencia de algoritmo IDP, encontrar valores cercanos al óptimo para estos parámetros en DE es muy sencillo.

Generalmente, los parámetros de los algoritmos DE son mantenidos constantes durante una optimización. Podría obtenerse un algoritmo mas efectivo y eficiente si estos parámetros se ajustaran automáticamente. En el Capítulo 4, fue propuesta una estrategia que on-line ajusta los valores de los parametros amplificación de la variación diferencial (F) y la constante de cruzamiento (CR) usando una medida de la diversidad de los individuos en la población. A grandes rasgos, esta estrategia utiliza valores grandes para F y pequeños para CR al inicio de la optimización, con la idea de favorecer una búsqueda global. Cuando la población se aproxima a una solución, F es reducido con objeto de favorecer una búsqueda local y el parámetro de cruzamiento CR es aumentado para incrementar la velocidad de convergencia del algoritmo. Esta estrategia fué implementada en el algoritmo *DE/rand/1/bin* y aplicada a los dos problemas de control óptimo multimodales, la eficiencia computacional mejoró significativamente y tambien la probabilidad de localizar el óptimo global.

Con la finalidad de conocer las oportunidades y ventajas que los algoritmos evolutivos tienen en la solución de problemas relacionados con control óptimo en el Capítulo 5 son consideradas varias aplicaciones de ingeniería acerca de control óptimo de cultivos en invernaderos. En el Capítulo 5.1 algoritmos geneticos con representación binaria de los individuos (Algoritmo Genético Simple) y con representacion de numeros reales (GENOCOP) son usados para estimar algunos de los parámetros de un modelo dinámico de un cultivo de lechuga con dos estados, el llamado modelo NICOLET. Este modelo tiene el propósito de predecir material seca y contenido de nitratos en lechuga al momento de la cosecha. Los problemas de estimación de parámetros generalmente presentan minimos locales. Este estudio mostró que los algoritmos evolutivos son adecuados para calibrar los parámetros de un modelo dinámico. No obstante, el tiempo de computación requerido es significativo. Parcialmente esto se debe a la alta carga computacional que la evaluación de la función objetivo requiere, lo cual en este caso implica la ejecución de una simulación completa. Si bien es verdad que la estimación de parámetros se lleva a cabo comúnmente fuera de linea (off-line), lo cual hace que el tiempo de computación sea menos importante, es preferible usar algoritmos evolutivos mas eficientes como los algoritmos DE.

En el Capítulo 5.2 un problema de control óptimo de concentración de nitratos en un cultivo de lechuga fué solucionado mediante dos algoritmos. El algoritmo basado en gradientes (ACW Adjustable Control-variation Weight), un método local de optimización y el algoritmo *DE/best/2/bin*, un método global de optimización. El sistema dinámico utilizado es un modelo de un cultivo de lechuga de dos estados (NICOLET B3) y el problema de control óptimo toma en cuenta tiempo final fijo y restricciones terminales de los estados. El algoritmo DE fué ampliado para tener en cuenta restricciones tanto en los estados como en los controles. Los resultados mostraron que este problema no tiene soluciones locales y que la parametrización de los controles requerida por DE, genera algunas dificultades para que el método directo sea capaz de encontrar con exactitud la solución obtenida por el algoritmo ACW. Pero por otro lado, la eficiencia computacional del algoritmo evolutivo fue impresionante. Por eso una conclusión evidente es la combinación de un algoritmo DE con un algoritmo basado en gradientes.

En el Capítulo 5.3 una combinación de un algoritmo DE y un algoritmo de primer orden basado en el cálculo de gradientes es usada para resolver un problema de control óptimo. El algoritmo DE es primeramente usado para obtener una solución aproximada en la vecindad del optimo global despues el algoritmo local es usado para refinar la solución. Este enfoque fué exitosamente probado en la solución del problema de control óptimo de niveles de nitratos en lechuga, el cual desafortunadamente parece no tener soluciones locales. Aun así, la viabilidad de este enfoque, que es interesante para problemas todo tipo de problemas de control óptimo de los cuales se desconoce a-priori si tienen soluciones locales, fué claramente demostrada.

Finalmente, en el Capítulo seis esta tesis finaliza con una discusión general, conclusiones y algunas sugerencias para investigación futura.

Curriculum Vitae

Irineo Lorenzo López Cruz was born in July 1961 in Tamazulapan, Oaxaca, Mexico. He obtained his Bachelor degree in Agricultural Engineering with honours in 1983 from the Universidad Autonoma Chapingo, Mexico. After he finished the credits of the MSc. program he created in 1992 the Central Computer Lab of Universidad Autonoma Chapingo. He received training in Relational Database Management Systems in 1994 from Oracle Mexico. He also received training on UNIX computer operating system and C programming language from UNAM in 1995. He obtained his MSc. Degree in Artificial Intelligence with honours in 1996 from Fundación Arturo Rosenblueth, Mexico. In November 1997, he started his PhD studies at the Systems and Control Group of the Wageningen University, The Netherlands. He has taught in Universidad Autonoma Chapingo. From August 2002 he will be working (as lecturer and researcher) at the Postgraduate program of Agricultural Engineering of Universidad Autonoma Chapingo in Chapingo, Mexico.