Studies in Computational Intelligence 366

Mario Köppen
Gerald Schaefer
Ajith Abraham (Eds.)

Intelligent Computational Optimization in Engineering

Techniques and Applications

Springer

Mario Köppen, Gerald Schaefer, and Ajith Abraham (Eds.)

Intelligent Computational Optimization in Engineering

Studies in Computational Intelligence, Volume 366

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Mario Köppen, Gerald Schaefer, and
Ajith Abraham (Eds.)

Intelligent Computational Optimization in Engineering

Techniques and Applications

Springer

Editors

Dr. Mario Köppen
Kyushu Institute of Technology
Dept. Artificial Intelligence
680-4 Kawazu
Izuka-Shi, Fukuoka 820-8502
Japan
E-mail: mkoeppen@pluto.ai.kyutech.ac.jp

Gerald Schaefer
School of Engineering and Applied Science
Aston University
Aston Triangle
Birmingham B4 7ET
U.K.
E-mail: G.Schaefer@aston.ac.uk

Prof. Ajith Abraham
Machine Intelligence Research Labs
(MIR Labs)
Scientific Network for Innovation and Research Excellence
P.O. Box 2259
Auburn, Washington 98071-2259
USA
E-mail: ajith.abraham@ieee.org

ISBN 978-3-642-21704-3 e-ISBN 978-3-642-21705-0

DOI 10.1007/978-3-642-21705-0

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2011931877

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

Many engineering problems involve heuristic search and optimization where, for example, an input parameter vector for a given system has to be found in order to optimize the system response. Also, many engineering optimization problems, once discretized, may become combinatorial in nature, which gives rise to certain difficulties in terms of solution procedure. In the first instance many problems have enormous search spaces, are NP-hard and hence require heuristic solution techniques. A second difficulty is the lack of ability of classical solution techniques to determine appropriate (global) optima of non-convex problems involving numerous (local) optima. Under these conditions, approaches based on recent advances in computational optimization techniques have been shown to be advantageous and successful compared to classical approaches.

This book is the result of an open call for chapter contributions. Researchers and practitioners were asked to share their experience and newest methodologies with regard to intelligent optimization in various application domains. From the many contributions, twelve were selected that constitute the main part of this book. The focus is clearly on the application, and most of the chapters do provide a case study for the application of intelligent optimization techniques in a real-world application. Other chapters discuss the general experience and means to adapt the plain textbook algorithms within a group of applications.

We want to use the opportunity to thank all who have contributed to this collection, especially to all authors, to the anonymous referees who much helped to gain a high-quality selection and to further improve the quality of the contributions. Our thanks also goes to the publisher Springer for all their support, patience and assistance during the preparation of this book.

January 2011

Mario Köppen
Gerald Schaefer
Ajith Abraham

List of Contributors

János Abonyi
University of Pannonia,
Department of Process Engineering
P.O. Box 158.
Veszprém H-8200, Hungary
abonyij@fmt.uni-pannon.hu

Ajith Abraham
Machine Intelligence Research Labs
(MIR Labs)
P.O. Box 2259
Auburn, Washington 98071-2259, USA
ajith.abraham@ieee.org

Sameh Bennour
University of Sfax
Tunisia

Mauro Birattari
IRIDIA, CoDE,
Université Libre de Bruxelles
Av. F.D. Roosevelt 50
Bruxelles, 1050, Belgium
mbiro@ulb.ac.be

Arijit Biswas
Department of Electronics and
Telecommunication Engineering
Jadavpur University
Kolkata 700032, India
arijitbiswas87@gmail.com

Mariam Bougharriou
University of Sfax
Tunisia

Eya Bradai
University of Sfax
Tunisia

Jonathan Carter
Imperial College
Exhibition Road
South Kensington, London,
SW7 2BP, UK
j.n.carter@imperial.ac.uk

Yi Chen
University of Electronic Science and
Technology of China
2006 Xiyuan Road
Chengdu 611731,
China
leo.chen.yi@gmail.com

Mike Christie
Heriot Watt University
Riccarton
Edinburgh EH14 4AS,
UK
mike.christie@pet.hw.ac.uk

Erik Cuevas
CUCEI, Universidad de Guadalajara
Av. Revolución No. 1500
Guadalajara, Jal., México,
C.P. 44430
erik.cuevas@cucei.udg.mx

Swagatam Das
Department of Electronics and
Telecommunication Engineering
Jadavpur University
Kolkata 700032,
India
swagatamdas19@yahoo.co.in

Sambarta Dasgupta
Department of Electronics and
Telecommunication Engineering
Jadavpur University
Kolkata 700032, India
sambartadg@gmail.com

Radu-Codruţ David
"Politehnica" University of Timisoara
Bd. V. Parvan 2
RO-300223 Timisoara, Romania
davidradu@gmail.com

Vasily Demyanov
Heriot Watt University
Riccarton
Edinburgh EH14 4AS, UK
vasily.demyanov@pet.hw.ac.uk

Mourad Fakhfakh
University of Sfax
Sfax
Sfax, 3018 Tunisia
mourad.fakhfakh@ieee.org

Georgina Flores-Becerra
Instituto Tecnológico de Puebla
Av. Tecnológico no. 420
Maravillas,
Puebla. 72220 México
kremhilda@gmail.com

Emna Gaddour
University of Sfax
Tunisia

Luis Gerardo de la Fraga
CINVESTAV
Av. IPN 2508
México City. 07360 México
fraga@cs.cinvestav.mx

Alexander Gibrekhterman
ClickSoftware Technologies Ltd.
94 Em Hamoshavot Road
Petach Tikva, 49527, Israel
alex.gibrekhterman@
clicksoftware.com

Crina Grosan
Brunel University
Information Systems and Computing
St John's 201, Uxbridge
UB8 3PH, United Kingdom

Ivick Guerra-Gómez
INAOE
Luis Enrique Erro no. 1
Tonantzintla, Puebla. 72840 México
ivickguerra@yahoo.com.mx

He Guo
Department of Computer Science
Dalian University of Technology
Dalian 116023, China
guohe@dlut.edu.cn

Yasin Hajizadeh
Heriot Watt University
Riccarton
Edinburgh EH14 4AS, UK
yasin.hajizadeh@pet.hw.ac.uk

Shan Jiang
Imperial College
Exhibition Road
South Kensington, London,
SW7 2BP, UK
shan.jiang04@imperial.ac.uk

Mario Köppen
NDRC, Kyushu Institute of Technology
680-4 Kawazu, Iizuka
Fukuoka 820-8502, Japan
mkoeppen@ieee.org

András Király
University of Pannonia,
Department of Process Engineering
P.O. Box 158.
Veszprém H-8200, Hungary
kandras85@gmail.com

Amit Konar
Department of Electronics and
Telecommunication Engineering
Jadavpur University
Kolkata 700032, India
konaramit@yahoo.co.in

Hongbo Liu
Department of Computer Science
Dalian University of Technology
Dalian 116023, China
lhb@dlut.edu.cn

Mourad Loulou
University of Sfax
Tunisia

Linah Mohamed
Heriot Watt University
Riccarton
Edinburgh EH14 4AS, UK
linah.mahgoub@pet.hw.ac.uk

Lars Nolle
School of Science and Technology
Nottingham Trent University
Clifton campus, Nottingham,
Nottinghamshire
lars.nolle@ntu.ac.uk

Mohamed K. Omar
Nottingham University Business School
Jalan Broga
Semenyih, Selangor Darul Ehsan
43500 Malaysia
mkhaledomar@gmail.com

Paola Pellegrini
Università Ca' Foscari
Cannaregio 873
Venice, 30121, Italy
paolap@unive.it

Marco Perez-Cisneros
CUCEI, Universidad de Guadalajara
Av. Revolución No. 1500
Guadalajara, Jal., México, C.P. 44430
marco.perez@cucei.udg.mx

Emil M. Petriu
University of Ottawa
800 King Edward
Ottawa, ON, K1N 6N5 Canada
petriu@site.uottawa.ca

Said Polanco-Martagon
Instituto Tecnológico de Puebla
Av. Tecnológico no. 420
Maravillas, Puebla. 72220 México
cannon.dm@gmail.com

Radu-Emil Precup
"Politehnica" University of Timisoara
Bd. V. Parvan 2
RO-300223 Timisoara, Romania
radu.precup@aut.upt.ro

Stefan Preitl
"Politehnica" University of Timisoara
Bd. V. Parvan 2
RO-300223 Timisoara, Romania
stefan.preitl@aut.upt.ro

Carlos Alberto Reyes-Garcia
INAOE
Luis Enrique Erro no. 1
Tonantzintla, Puebla. 72840 México
kargaxxi@inaoep.mx

Gerardo Reyes-Salgado
CENIDET
Interior Internado
Palmira, Cuernavaca. 62490 M"exico

Gustavo Rodriguez-Gomez
INAOE
Luis Enrique Erro no. 1
Tonantzintla, Puebla. 72840 México
grodrig@inaoep.mx

Raúl Rojas
Institut für Informatik,
Freie Universität Berlin
Takustrasse 9
Berlin, Germany, PLZ 14195
rojas@inf.fu-berlin.de

Amin Sallem
University of Sfax
Tunisia

Gerald Schaefer
Department of Computer Science
Loughborough University
Loughborough, LE11 3TU, U.K.
g.schaefer@lboro.ac.uk

Yasothei Suppiah
Multimedia University
Jalan Ayer Keroh Lama,
Melaka 75450, Malaysia
yasothei.suppiah@mmu.edu.my

József K. Tar
Obuda University
Becsi ut 96/B
H-1034 Budapest, Hungary
tar.jozsef@nik.uni-obuda.hu

Esteban Tlelo-Cuautle
INAOE
Luis Enrique Erro no. 1
Tonantzintla, Puebla. 72840 México
etlelo@inaoep.mx

Dovi Yellin
ClickSoftware Technologies Ltd.
94 Em Hamoshavot Road
Petach Tikva, 49527, Israel
dovi.yellin@clicksoftware.com

Uzi Zahavi
ClickSoftware Technologies Ltd.
94 Em Hamoshavot Road
Petach Tikva, 49527, Israel
uzi.zahavi@clicksoftware.com

Daniel Zaldivar
CUCEI, Universidad de Guadalajara
Av. Revolución No. 1500
Guadalajara, Jal., México, C.P. 44430
daniel.zaldivar@cucei.udg.mx

Marko Žerdin
University of Southampton
Highfield
Southampton, SO17 1BJ, UK
marko@zanyants.com

Dilay Çelebi
Istanbul Technical University
34367 Maçka, Istanbul, Turkey
celebid@itu.edu.tr

Contents

Intelligent Computational Optimization in Engineering: Techniques and Applications

Lars Nolle, Mario Köppen, Gerald Schaefer, and Ajith Abraham

Abstract. Many problems can be formulated as optimization problems. Among the many classes of algorithms for solving such problems, one interesting, biologically inspired group is that of meta-heuristic optimization techniques. In this introductionary chapter we provide an overview of such techniques, in particular of Genetic Algorithms, Ant Colony Optimization and Particle Swarm Optimization techniques.

1 Introduction

Many engineering problems involve heuristic search and optimization where, for example, an input parameter vector for a given system has to be found in order to optimize the system response. Also, many engineering optimization problems, once discretized, may become combinatorial in nature, which gives rise to certain difficulties in terms of solution procedure. In the first instance many problems have enormous search spaces, are NP-hard and hence require heuristic solution techniques. A second difficulty is the lack of ability of classical solution techniques to determine appropriate (global) optima of non-convex problems involving numerous (local) optima. Under these conditions, approaches based on recent advances

Lars Nolle
School of Science and Technology, Nottingham Trent University, U.K.
e-mail: lars.nolle@ntu.ac.uk

Mario Köppen
NDRC, Kyushu Institute of Technology
e-mail: mkoeppen@ieee.org

Gerald Schaefer
Department of Computer Science, Loughborough University
e-mail: G.Schaefer@lboro.ac.uk

Ajith Abraham
Machine Intelligence Research Labs (MIR Labs)
e-mail: ajith.abraham@ieee.org

in computational optimization techniques have been shown to be advantageous and successful compared to classical approaches.

The single contributions of this book detail the successful use of computational optimization techniques that use (meta-)heuristic search to solve non-convex and complex engineering problems. In the following, we want to give a short overview of basic meta-heuristic optimization techniques from a more academic perspective, before detailing the application aspects of these methods.

2 Evolutionary Computing

Many scientific problems can be viewed as search or optimization problems, where an optimum input parameter vector for a given system has to be found in order to maximise or to minimise the system response to that input vector. Often, auxiliary information about the system, like its transfer function and derivatives, is not known and the measures might be incomplete and distorted by noise. This makes such problems difficult to be solved by traditional mathematical methods. Here, evolutionary optimization algorithms, which are based on biological principles borrowed from nature, can offer a solution. These algorithms work on a population of candidate solutions, which are iteratively improved so that an optimal solution evolves over time.

This chapter discusses the general problem of search and optimization before it introduces the systems view, followed by a definition of search space and fitness landscape. It then explains the process of optimization and the concept of optimization loops. It continuous with introducing biological-inspired evolutionary optimization algorithms, namely Genetic Algorithms and Genetic Programming. Other evolutionary inspired optimization techniques, namely Ant Colony Optimization and Particle Swarm Optimization are also discussed.

2.1 Systems

Every process or object can be seen as a system. Fenton and Hill [9] define a system as "*...an assembly of components, connected together in an organised way, and separated from its environment by a boundary. This organised assembly has an observable purpose which is characterised in terms of how it transforms input from the environment into output to the environment.*" By definition, a system has exactly one input channel x and exactly one output channel y (Figure 1). All interactions with the environment have to be made through these interfaces.

Both input and output can be vectors or scalars. The input is called the *independent variable* or *parameter*, because its value(s) can be chosen freely, and results in the output y, the so-called *dependent variable*. If the present state of the system does not depend on previous states but only on the current input, the system is said to be a *steady state system*, the output of the system can be described as a function of the input $y = f(x)$.

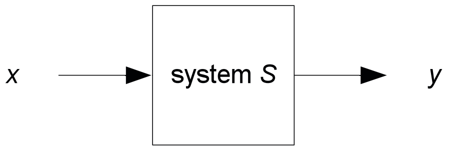


Fig. 1 Generic system.

2.2 *Objective Function*

In order to rate the quality of a candidate solution x, it is necessary to transform the system response to x into an appropriate measure, called the *objective* or *fitness*. If the system has only one output variable, the system output y equals the fitness. If y has more than one component the output variables of the system have to be combined into a single value, computed by the so called *objective function* or *fitness function*. In general, there are four approaches to judge the system output: *aggregation*, the *Changing Objectives Method*, the Use of *Niche Techniques* and *Pareto Based Methods* [10]. The most often used method is aggregation. In its simplest case, the fitness function $F(x)$ equals the weighted sum of the components $y_i = c_i \cdot F_i(x)$ of y, where c_i is the weight for component i:

$$F(x) = c_0 + c_1 \cdot F_1(x) + \cdots + c_n \cdot F_n(x) \tag{1}$$

2.3 *Search Space and Fitness Landscape*

If all the possible candidate solutions are collected in an ordered way, this collection is called the *search space*. Sometimes, this space is also referred to as input space. For an optimization problem of dimension n, i.e. a system with n independent parameters, the search space also has dimension n. By adding the dimension *Fitness* or *Costs* to the search space, one will get the $(n+1)$-dimensional *fitness landscape* [23].

2.4 *Optimization*

Optimization [21] is the process of selecting the best candidate solution from a range of possibilities, i.e. the search space. In other words, a system S, that has to be optimized in terms of a quality output value y, is brought into a new state that has a better quality output value y than the previous state. This is done by changing the independent input parameters x. The error function describes the difference between the predefined objective $y_{desired}$ and systems response $f(x)$ to the input x.

$$Error(x) = y_{desired} - f(x) \tag{2}$$

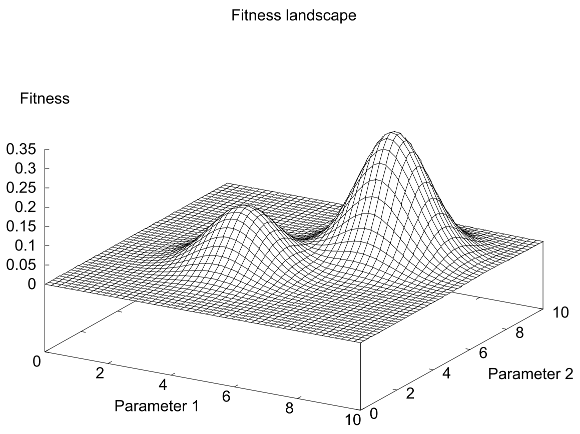


Fig. 2 Example of a fitness landscape for a system with two input parameters.

Usually, the aim is to find the vector x' that leads to a minimal error for the system S, i.e. the minimal departure from the optimal output value:

$$Error(x') = 0 \tag{3}$$

Often, a predefined target value is not known. In this case one tries to gain a fitness value that is as high as possible in case of maximisation, or as low a possible in the case of minimisation.

Ideally, one would evaluate all possible candidates and choose the best one. This is known as exhaustive search. However, often it is not feasible to consider all possible solutions, for example if the search space is too large and the evaluation of a single candidate is too expensive. In this case, only a subset of the solutions can be evaluated.

Optimization problems can be either function optimization problems or combinatorial problems. The first class of problems can be divided in continuous optimization and discrete optimization problems. In continuous function optimization, the independent variables are real numbers whereas for discrete function optimization, the independent variables can only be chosen from a predefined set of allowed and somehow ordered numbers, for example $\{10, 20, 30, 40\}$.

In combinatorial optimization problems, the optimum sequence or combination of a fixed set of input values has to be found. Here, the input values are symbols and might not be connected or ordered, for example $\{apple, orange, strawberry\}$. An example of a combinatorial optimization problem is the classical Travelling Salesman Problem (TSP), where a sales agent needs to visit a predefined set of cities and return to base. The problem here is to find an optimal route, i.e. the route that connects all cities whilst having the shortest travel distance, by choosing the order in which the cities are visited.

2.5 Optimization Loop

Mathematical or *calculus-based* methods use known functional relationships between variables and objectives to calculate the optimum of the given system. Therefore, an exact mathematical model of the process must exist. Edelbaum [8] introduced the differentiation of calculus-based methods in *direct methods* and *indirect methods*.

Direct methods solve the optimization problem by iterative calculation and derivation of the error function and moving in a direction to the maximum slope gradient. Indirect methods solve the optimization problem in one step – without testing – by solving a set of equations (usually non-linear). These equations result from setting the derivative of the error function equal to zero. Both classes of methods are local in scope, i.e. they tend to find only local optima. Therefore, they are not robust. They depend on the existence of derivatives. Real problem functions tend to be perturbed by noise and are not smooth, i.e. derivations may not exist for all points of functions. This class of problem cannot be solved by mathematical methods.

If the functional relations between input variables and objectives are not known, one can experiment on the real system (or a model of this system) in order to find the optimum. Access to the independent variables must exist for the whole multi-dimensional search space, i.e. the collection of all possible candidate solutions. Also a possibility of measuring the independent variable and the objective must be given. The optimization process is iterative, i.e. it has to be done in a closed *optimization loop* (Figure 3).

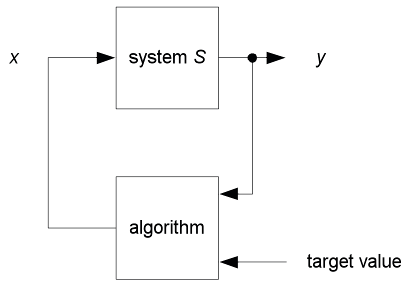


Fig. 3 Closed optimization loop consisting of a system and an optimization algorithm.

Experimental optimization methods can therefore be seen as a search for the optimum by traversing over the fitness landscape.

3 Genetic Algorithms

As Darwin's theory of natural selection articulates, nature is very effective at optimization, e.g. to enable life-forms to survive in a unfriendly and changing environment, only by means of the simple method of trial and error. Genetic Algorithms (GAs) simulate this evolutionary mechanism by using *heredity* and *mutation*. They were first introduced in 1975 by Holland [11] who also provided a theoretical framework for Genetic Algorithms, the *Schemata Theorem* [13].

For genetic algorithms, the independent input parameters of a system S (Figure 4) are coded into a binary string, the *genotype* of an individual (Figure 5).

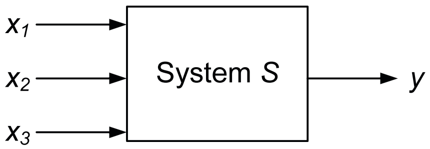
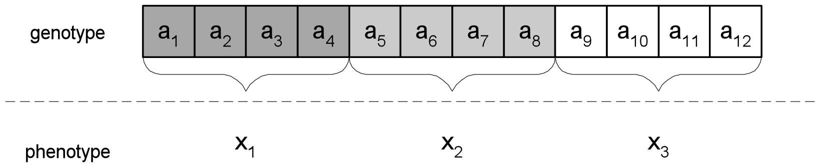


Fig. 4 System to be optimized.



genotype
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12}
phenotype
x_1
x_2
x_3

Fig. 5 Binary string representing one input pattern of the system.

The individual represented by genotype is called a *phenotype*. This phenotype has a certain quality or *fitness* to survive which can be determined by presenting the phenotype to the system S and measuring the system response.

The search is not only undertaken by one individual but by a population of n genotypes, the genepool. Therefore, the search space is tested at n points in parallel. All the individuals of the genepool at a time t_n are called a *generation*.

A new generation for time t_{n+1} is generated by selecting N individuals from the current population for breeding. They are copying into the genepool of the next generation and their genetic information is then recombined, using the *cross-over* operator (see 3.2), with a predefined cross-over probability p_c. The resulting offspring is then copied into the new genepool and mutation is applied to the offspring. Figure 7 shows the flowchart of a simple Genetic Algorithm.

The search will be carried out until at least one individual has a better fitness than the defined minimum fitness, or a maximum number of generations have been reached.

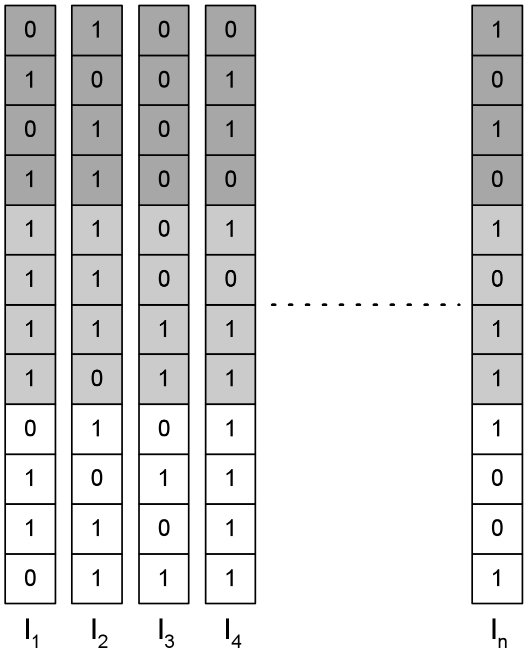


Fig. 6 Genepool consisting of individuals $I_1, \ldots, I_n$.

3.1 Selection

In general, there are three different approaches to choose individuals from the current generation for re-production, namely *Tournament Selection*, *Fitness Proportional Selection* and *Rank Based Selection*. In Tournament Selection, two or more individuals are randomly selected from the current generation of N genotypes to compete with each other. The individual with the highest fitness of this set is the winner and will be selected for generating offspring. The process is repeated N times in order to create the new population. Using Tournament Selection, the least fit individual can never be selected.

In fitness proportional selection, the chance of an individual to be selected is related to its fitness value. The most commonly used method of this type is Roulette Wheel Selection. Here, proportions of an imaginary roulette wheel are distributed in proportion to the relative fitness of an individual. Figure 8 shows and example for $N = 3$. In this example, the fitness of individual 3 is approximately four times higher than the fitness of individual 1, i.e. its chance to be selected is four times greater then the chance that individual one is selected. For a population of N individuals, the wheel is spun N times and the individual under the pointer is selected. In fitness proportional selection, all individuals have a chance of selection but high fitness individuals are more likely to be selected, because they occupy a larger portion of the wheel.

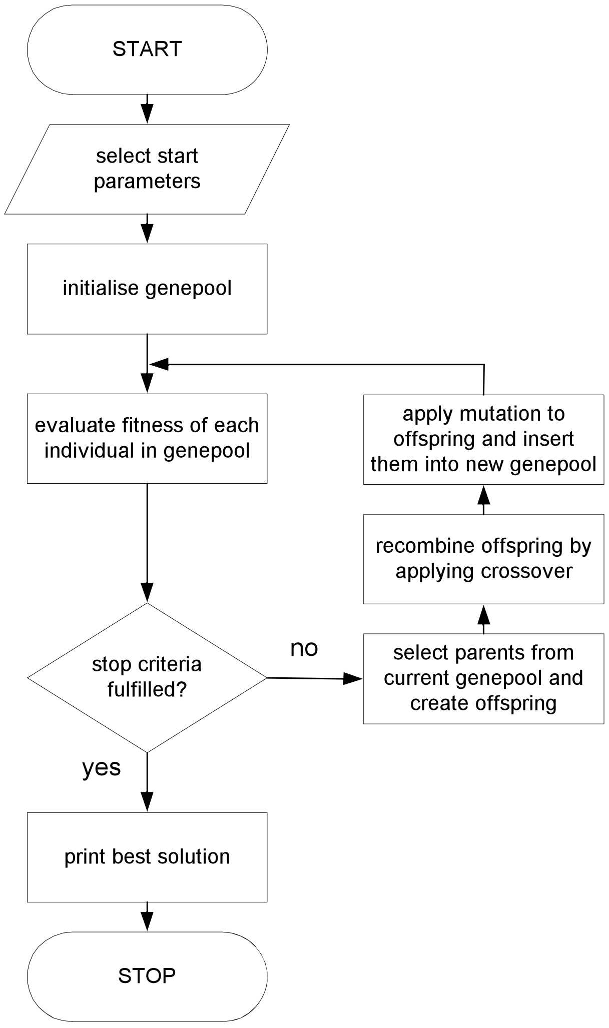

START
select start parameters
initialise genepool
evaluate fitness of each individual in genepool
stop criteria fulfilled?
no
yes
select parents from current genepool and create offspring
recombine offspring by applying crossover
apply mutation to offspring and insert them into new genepool
print best solution
STOP

Fig. 7 Flowchart of basic GA algorithm.

However, there is the statistical chance that the actual selected distribution might not reflect the expected distribution based on the fitness values. If the selection is too strong, it can lead to premature convergence, i.e. the population would converge before it has found the region of the search space that contains the global optimum. In other words, the exploitation would start before the search space is fully explored. On the other hand, if the selection is too weak, it can lead to stalled evolution, which means the search is reduced to randomly walking through search space.

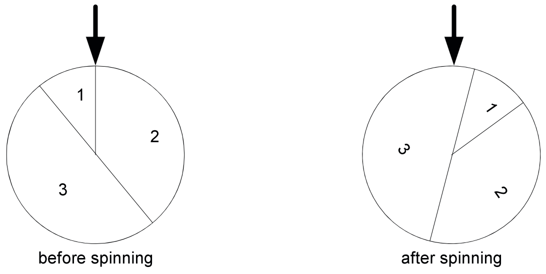


Fig. 8 Roulette Wheel selection.

These effects are overcome using Stochastic Universal Selection (SUS). Here, the same roulette wheel is used, but instead of using a single pointer, *N* equally-spaced pointers are used for a population of *N* individuals and the wheel is spun only once (Figure 9).

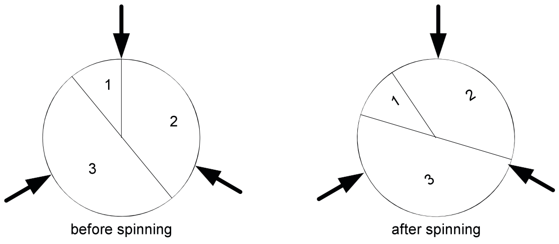


Fig. 9 SUS selection.

Instead of using the fitness of an individual for selection, a selective *s* value can be used, which is based on the rank position of an individual in the population (Equation 4).

$$s_i = Min + (Max - Min)\frac{rank_i - 1}{N - 1} \tag{4}$$

where

Min:	minimum fitness within a generation
Max:	maximum fitness within a generation
$rank_i$:	rank of individual i within the population in a generation
N:	number of individuals within population

So, instead of using the raw fitness to determine the proportion for an individual, the rank of the individual within the generation is used.

Sometimes the m fittest individuals in a generation are cloned into the next generation in order to make sure to preserve their genetic material. This is known as *elitism*.

3.2 Cross-Over

The most important operator in terms of robustness of the algorithm is the cross-over operator. Figure 10 shows the so-called *one-point cross-over* operator, which combines the information of two parents. They are aligning and then both cut at a randomly chosen cross-over point and the tails are swapped successively.

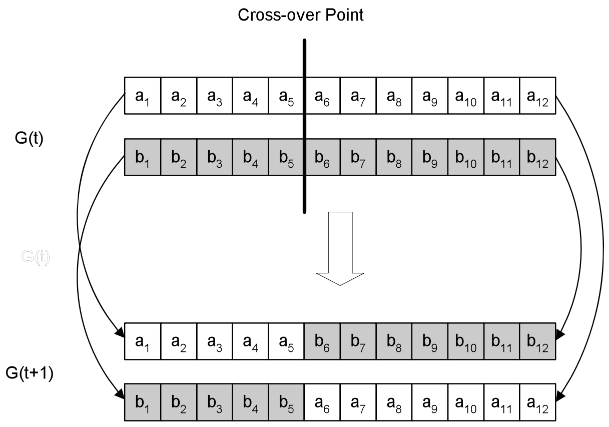

Cross-over Point
G(t)
a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12
G(t+1)
a1 a2 a3 a4 a5 b6 b7 b8 b9 b10 b11 b12
b1 b2 b3 b4 b5 a6 a7 a8 a9 a10 a11 a12

Fig. 10 Cross-over operator.

Instead of a single cross-over point, two or more random cross-over points can be used for recombining the genetic information of the parents.

Another form of cross-over is called *Uniform cross over* [22]. Here, every component of a parent individual X is randomly passed on either to offspring A or offspring B. If X passes on its component to A, the position in B is filled using the component from parent Y and vice versa.

3.3 Mutation

After the genetic information of the parents is recombined using cross-over, mutation is applied to every individual of the new generation. Here, every bit of the offspring is inverted (*mutated*) with probability p_m. The mutation operator is important for restoring lost information and therewith to result in a better effectiveness of the GA.

3.4 Discussion

The advantages of GAs are that they use payoff (objective function) information, not derivatives or other auxiliary knowledge, i.e. they are black box optimization methods. GAs tend to converge towards the global optimum rather than getting stuck in a local optimum and therefore they are very robust. On the other hand, it is not always straightforward to find the right GA parameters for a particular optimization problem, e.g. a suitable genepool size or mutation probability. Also, the efficiency of GAs relies heavily on the right coding of the input parameters, i.e. the chosen mapping function from phenotype to genotype, and they tend to fail if the inputs of the system are heavily correlated.

3.5 Schemata Theorem

Holland provided a theoretical foundation of GAs, i.e. a theoretical proof of convergence, which he called the *Schemata Theorem*. A schema is a template for binary strings, but built from a three letter alphabet containing the symbols *, 0 and 1. The * symbol is the ‘dont care symbol’ which either stands for 0 or 1. Figure 11 shows an example of a schema for chromosomes consisting of 12 bits, of which three are set to the dont care symbol and the remaining nine bits are set to fixed values.

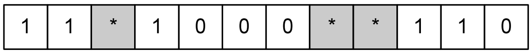


Fig. 11 Example of a schema in GA.

The distance between the first and the last fixed bit is called the *defined length* of the schema and the number of fixed bits is called the *order* of the schema. Figure 12 shows an example of a schema H and the different instances it represents.

A binary string s is an instance of a schema H if it fits into the template. Therefore, any binary string of length l does not just represent one candidate solution, it is also an instance of 2^l schemata at the same time. As a consequence, a GA with the genepool of size n does not only test n different solutions at the same time, but also a high number of different schemata. This is known as *implicit parallelism* in GA and provides an explanation for their effectiveness and efficiency.

According to Holland, the number of instances m of a schema H that are contained in the population at generation $t+1$ can be determined as follows:

$$m(H,t+1) = m(H,t) \cdot \frac{\bar{f}(H)}{\bar{f}} \tag{5}$$

where

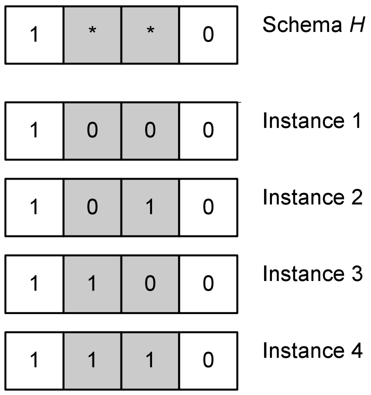


Fig. 12 Example of a schema H and the instances it represents.

H:	Schema or "Building Block" with at least one instance in the last generation,
$m(H,t)$:	number of instances of H at time t,
$m(H,t+1)$:	number of instances of H at time $t+1$,
$\bar{f}(H)$:	average fitness of the instances of schema H,
$\bar{f}$:	average fitness of the whole population.

This is a simplified version of the schemata theorem, because it does not take into account the effects of the cross-over and the mutation operator. However, it is sufficient to demonstrate the basic idea. A more detailed description can be found, for example, in the book of Goldberg [13].

Suppose that a particular schemata H remains above-average an amount $c \cdot \bar{f}$ with c being a constant factor, equation 4 can be rewritten as follows:

$$m(H,t+1) = m(H,t) \cdot \frac{\bar{f} + c \cdot \bar{f}}{\bar{f}} = (1+c) \cdot m(H,t) \tag{6}$$

Assuming c is stationary and starts at $t = 0$, equation 5 can be rewritten as follows:

$$m(H,t) = m(H,0) \cdot (1+c)^{t} \tag{7}$$

It can be seen that this equation is similar to the formula of interest: the number of instances of a schema H with a fitness above-average grows exponentially to generation t. Hence, schemata with good fitness will survive and ones with a fitness below average will eventually die out. Therefore, the fitter building blocks, i.e. the better partial solution, will take over the genepool within finite time. However, the schemata theorem is controversial, because it assumes that the factor c is constant over time.

3.6 Coding Problem

Traditionally, GAs use binary stings. However, if an input variable is coded using standard binary coding, this can lead to the problem that a small change in the phenotype would require a large number of bits of the genotype to be inverted. An example of the coding problem is given in Figure 13.

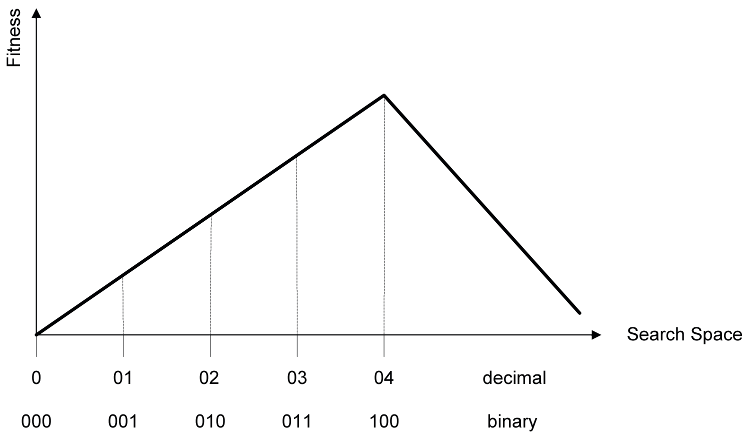


Fig. 13 Differences between decimal and standard binary coding.

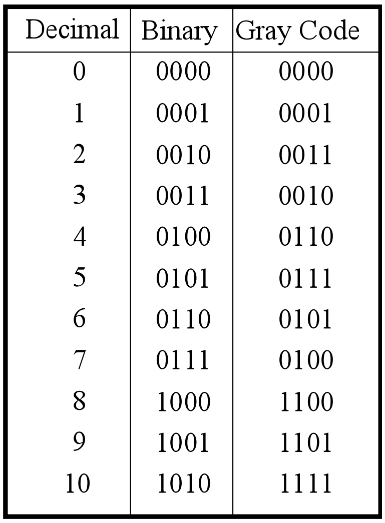
As it can be seen from the figure, a step from 3_{10} to 4_{10} requires flipping 3 bits in binary representation whereas it only changes the least significant digit in decimal representation by one. One solution is to use Gray Code, which has the advantage that only one bit changes between any two positions, i.e. it has a constant Hamming Distance of one.

4 Ant Colony Optimization

Ant Colony Optimization (ACO) [4] refers to a class of discrete optimization algorithms, i.e. a meta-heuristic, which is modelled on the collective behaviour of ant colonies.

Real ants are very limited in their individual cognitive and visual capabilities, but an ant colony as a social entity is capable of solving complex problems and tasks in order to survive in an ever-changing hostile environment. For example, ants are capable of finding the shortest path to a food source [14]. If the food source is depleted, the ant colony adapts itself in a way that it will explore the environment and discover new food sources.

Ants communicate indirectly with other ants by depositing a substance called pheromone on the ground while they are walking around. This pheromone trail can then be used by the ant to find its way back to its nest after the ant has found a food

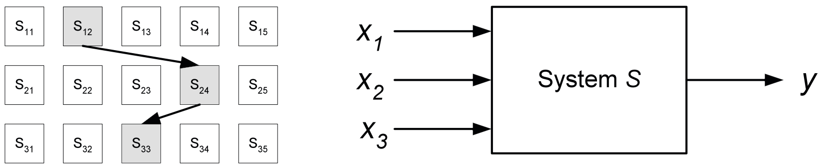


Decimal	Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111

Fig. 14 Gray Code.

source and other ants can also sense it. Ants have the tendency to follow existing paths with high pheromone levels. If there is no existing pheromone trail, they walk around in a random fashion. If an ant has to make a decision, for example to choose a way around an obstacle in its way, it follows existing paths with a high probability. However, there is always a chance that the ant explores a new path or a path with a lower pheromone level. If an ant has chosen an existing path, the pheromone level of this path will be increased because the ants deposit new pheromone on top of the existing one. This makes it more likely that other ants will also follow this path, increasing the pheromone level again. This positive feedback process is known as autocatalysis [5]. Although the pheromone evaporates over time, the entire colony builds up a complex solution based on this indirect form of communication, called stigmergy [4].

Figure 15 demonstrates the basic principle of the ACO meta-heuristic, which is modelled after the behaviour described above. In this example, the system S that has to be optimized has three independent variables $x_1 \dots x_3$ and the quality of the solution can be measured by the achieved fitness value y. Each input can have one of five different discrete alternative values s_{ij}, where i represents the input and j the chosen alternative for that input. Each alternative has an associated probability value, which is randomly initialised. The collection of probability distributions can be seen as a global probability matrix. Each artificial ant in the colony has to choose randomly a ‘path’ through state space, i.e. the input value for each independent variable. In the example in Figure 15, the ant chooses alternative s_{12} for input x_1, s_{24} for input x_2 and s_{33} for input x_3. The chosen path depends on the probabilities associated with the states, i.e. a state with a high probability is more likely to be selected for a trial solution than states with a low probability value. This probability values are refereed to as the pheromone level τ.



s_{11} s_{12} s_{13} s_{14} s_{15}
s_{21} s_{22} s_{23} s_{24} s_{25}
s_{31} s_{32} s_{33} s_{34} s_{35}
x_1
x_2
x_3
System S
y

Fig. 15 Example of an artificial ant constructing a trial vector by traversing through state space.

A chosen path represents one candidate solution, which is evaluated and the probabilities of the states that the ant has visited on that trail is updated based on the achieved fitness. In the next generation, the updated probability matrix is used, which means that states that have proven fit in the past are more likely to be selected for the subsequent trail. However, it should be pointed out that a 'path' is not actually traversing through the state space; it simply refers to the collection of chosen alternatives for a particular candidate solution. The order in which the states are selected does not have any effect on the candidate solution itself, i.e. one could start with determining the input for x_1 first or, alternatively, with x_2 or x_3. The resulting candidate solutions would still be the same.

A major advantage of ACO is that adjacent states in the neighbourhood do not need to show similarities, i.e. the state space does not need to be ordered. This is different to most optimization heuristics, which rely on ordered collections of states, i.e. fitness landscapes.

Figure 16 shows a flowchart of the basic ACO meta-heuristic for a colony consisting of n artificial ants. During one iteration, called a time-step, every ant generates a trial solution, which is evaluated and based on the fitness of the solution the pheromone level of the states involved in the trail is updated in a local probability matrix for the ant. After one iteration, i.e. time-step, all the local probability matrices are combined and added to the global one, which is usually scaled down in order to simulate the evaporation process of real pheromone trails. This helps to avoid search stagnation and ensures that ants maintain their ability to explore new regions of the state space.

The main principle of ACO is that a colony of artificial ants builds up discrete probability distributions for each input parameter of a system to be optimized. Figure 17 shows an example of a probability distribution for an input i with ten alternative states.

It can be seen that state s_{i8} has the highest pheromone level, i.e. probability, and hence has a high chance to be selected for a trial. States s_{i7} and s_{i10}, on the other hand, have a pheromone level of zero and can never be selected. However, even states with a low pheromone level, e.g. s_{i3} in Figure 17, have a certain chance to be selected.

Initially, every possible choice for each of the input variables is set to a very low probability, which is the equivalent to the pheromone level in the real world. Each individual ant then chooses randomly one value for each input parameter, i.e. builds up a candidate solution, based on the probability distributions of the input values.

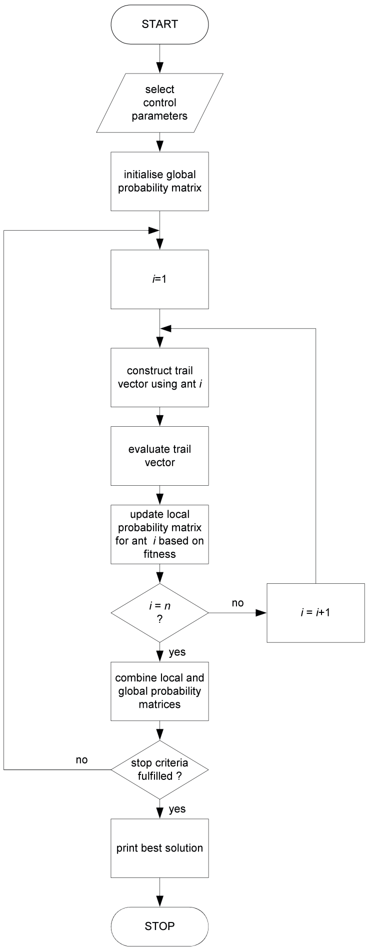

START
select control parameters
initialise global probability matrix
i=1
construct trail vector using ant *i*
evaluate trail vector
update local probability matrix for ant *i* based on fitness
i = *n* ?
no
i = *i*+1
yes
combine local and global probability matrices
stop criteria fulfilled ?
no
yes
print best solution
STOP

Fig. 16 Flowchart of ACO meta-heuristic.

Depending on the quality of the resulting candidate solution, the probability values of the chosen input values are updated. The whole process is repeated in iterations called time-steps until a suitable solution is found or the algorithm has converged, i.e. has reached a stable set of probability distributions. It has been proved, for example by Stützle and Dorigo [19] and Gutjahr [15], that ACO algorithms are capable of converging towards the global optimum within finite time.

The first computational optimization algorithm based on ant colonies was the Ant System (AS) algorithm [2]. It was successfully applied to the Travelling Salesman Problem and the Quadratic Assignment Problem. This was later followed by the Ant Colony System (ACS) [6], the Max-Min Ant System (MMAS) [20] and the Rank-Based Ant System (RBAS) [1].

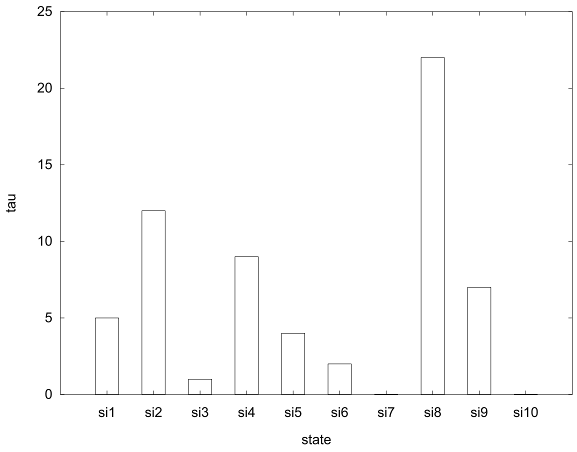


Fig. 17 Example of a probability distribution based on pheromone level.

For ACO the probability of a state s_{ij} to be chosen as input parameter i can be calculated using the following transition rule (Equation 8):

$$p(s_{ij}) = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\Sigma_{j=1}^{m} \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}} & \text{if } s_{ij} \in N_i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Where τ_{ij} is the pheromone level for state s_{ij}, η_{ij} is a heuristic value related to the fitness of the solution, α and β are control parameters that determine the relative importance of pheromone versus fitness, m is the number of alternatives for input parameter i, and N_i is the set of possible alternatives for input i. If the heuristic value η_{ij} is set to a constant value of one, the algorithm becomes the Simple Ant Colony Optimization algorithm (SACO) [7].

The evaporation after time-step t can be computed using Equation 9, where $\rho \in (0,1])$ is the evaporation rate.

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) \tag{9}$$

The pheromone updating rule is given in Equation 10, with $\Delta\tau_{ij}(t) = f(y_1, y_2, \ldots, y_n)$:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{10}$$

Unlike real ants, artificial ants can be equipped with additional capabilities, for example with look ahead capabilities [17] and backtracking [3] in order to improve efficiency. They can also be combined with local search methods [4, 18].

However, one problem related to ACO is that it is not a straightforward task to find optimum control parameter settings for an ACO application [12].

5 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a simple but effective algorithm that was original developed by Kennedy and Eberhart [16] for continuous function optimization. It is based on the social behaviour of a collection of animals that can be observed, for example, in fish schooling and bird flocking. PSO uses a population of agents where the population is referred to as swarm and the agents are called particles. Each particle represents an input vector for the system and is randomly initialised.

Each particle i has a position $x_{ij}(t)$ and a velocity $v_{ij}(t)$ for each dimension j of the search space. In every iteration of the algorithm, i is 'flying' through search space by adjusting the position vector $x_i(t)$ using the velocity vector $v_i(t)$ as follows:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{11}$$

It should be stressed that, in the physical world, a velocity and a position cannot be added. The velocity would need to be multiplied with a time interval in order to get a distance that could then be added to the original position. However, if one thinks of an iteration as a time step, the velocity vector could be multiplied with one time unit, which would not change the actual value but it would change the unit. The velocity vector itself is determined using the following equation:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_2 (x_{i\_best} - x_{ij}(t)) + c_2 r_2 (x_{global\_best} - x_{ij}(t)) \tag{12}$$

Here, r_1 and r_2 are random numbers, c_1 and c_2 are tuning constants, x_{i_best} is the best position that particle i found during the search so far and x_{global_best} is the best position the swarm found so far. The second term in Equation 12 is called the component cognitive component whereas the third one is called the social component. Figure 18 shows a flow-chart of the basic PSO algorithm.

One variation of the basic PSO algorithm is that, instead of using the global best position, the best position of the neighbourhood of particle i is used in the social component.

6 Overview

The single chapters of this book will demonstrate the practical application of the before-mentioned meta-heuristic optimization techniques. It can be seen how the pure academic perspective taken so far conveys into the complex aspects of an engineering application.

The chapter "Learning Automata in Control Planning Strategies" explores the use of planning in order to improve the performance of feedback-based control schemes considering only one probabilistic approach known as Learning Automata (LA). Authors propose a novel scheme that has been motivated by the human ability of choosing among different alternative plans while solving daily-life problems. Considering that Planning provides a very general and easy applicable methodology, the overall approach combines the use of Model predictive Control as regulation

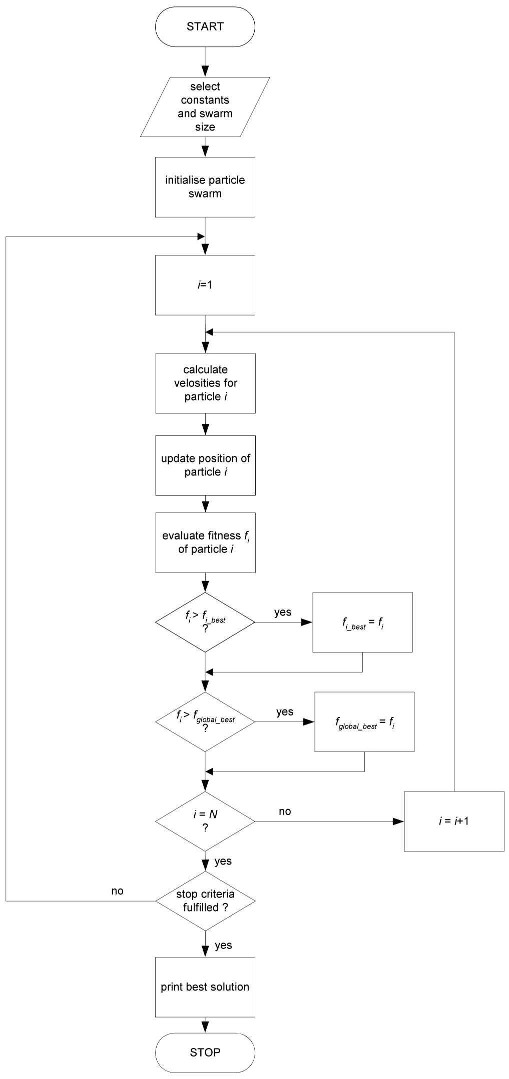

START
select constants and swarm size
initialise particle swarm
i=1
calculate velocities for particle i
update position of particle i
evaluate fitness f_i of particle i
$f_i > f_{i\_best}$?
yes
$f_{i\_best} = f_i$
$f_i > f_{global\_best}$?
yes
$f_{global\_best} = f_i$
i = N ?
no
i = i+1
yes
stop criteria fulfilled ?
no
yes
print best solution
STOP

Fig. 18 Flowchart of PSO algorithm.

strategy and LA as optimization technique. The latter provides remarkable capabilities for global optimization on multimodal surfaces. By using LA, the search for the optimum is done over a probability space rather than exploring the parameter domain as it is commonly done by traditional algorithms. Some experiments and comparisons are conducted over a conventional control plant in order to demonstrate the proposed framework's performance.

In the chapter "Optimization Strategies for Restricted Candidate Lists in Field Service Scheduling," authors formally define a real-world class of combinatorial optimization problems called Field Service Scheduling (FSS), and position it in relation to some optimization problems more often encountered in the literature. The presented research was motivated by the challenges encountered at routinely solving large instances of these problems in an industrial setting. The authors propose a generic framework that combines the expressiveness to address a wide variety of such problems with the ability to tune the optimization process to the type of instances being solved. Stategies for restricted candidate lists are introduced as a way to accelerate the pace of convergence when particular problem features are present. Two such strategies, constrained clustering and bundling, are explored in some detail, yielding some surprising results and material for further research. The presented experimental results were obtained on top of a GRASP meta-heuristic, but the presented approach of using strategies does not depend in any meaningful way on the particular meta-heuristic in use and can be generalized to other optimization methods.

The chapter "Framework for Integrating Optimization and Heuristic Models for Solving Planning and Scheduling Problem in a Resin Manufacturing Plant" describes a methodology that combines optimization and heuristic models to provide a solution to the planning and scheduling problem in a batch chemical plant. Real-life industrial data from a resin manufacturing plant was used to validate and test the robustness of the proposed methodology. The results of the proposed methodology are encouraging and provide substantial benefits to practitioners.

The chapter "Evolutionary Algorithms in the Optimal Sizing of Analog Circuits" highlights the application of two multi-objective evolutionary algorithms (NSGA-II and MOEA/D) in the optimization of analog integrated circuits. The algorithms use the circuit simulator SPICE to evaluate performances of unity-gain cells, current conveyors and CFOA. The results on the sizing process show that the genetic operator known as differential evolution increases the dominance of both evolutionary algorithms. Finally, some heuristics regarding the effectiveness of NSGA-II and MOEA/D in the sizing of analog integrated circuits are summarized.

Managing the core of a nuclear reactor so as to maximise the energy produced whilst meeting all of the safety requirements is a difficult and complex task, which is the topic of the chapter "Application of Estimation of Distribution Algorithms for Nuclear Fuel Management." A reactor core consists of many vertical channels into which a tube, containing the fuel, can be placed. Typically the fuel in each tube is different, so that one needs to place N different objects into each of N locations. This is a difficult problem in combinatorial optimization. The best available algorithm for this problem, a genetic algorithm with a specially designed crossover operator, was created nearly twenty years ago. In this research, an estimation of distribution algorithm supplemented with heuristic information to tackle the problem was used. The estimation of distribution algorithm produces a probability distribution function for each channel that indicates which fuel tubes are likely to be found in that channel in an optimal solution. Conversely it indicates which fuel tubes will not be found in the channel. The algorithm starts with a uniform probability distribution, i.e. all solutions are equally likely. The distribution is then sampled and the proposed solutions tested, this then allows the probability distribution to be updated.

The process can then be repeated until the algorithm converges onto an invariant probability distribution function. This algorithm has significantly outperformed the previous state-of-the-art method and represents a major improvement in the ability to tackle this problem.

The chapter "Optimal Control Systems with Reduced Parametric Sensitivity Based on Particle Swarm Optimization and Simulated Annealing" offers the design of optimal control systems with a reduced parametric sensitivity on the basis of Particle Swarm Optimization (PSO) and Simulated Annealing (SA) algorithms that belong to the popular category of nature-inspired optimization algorithms. PSO and SA algorithms are employed in solving optimization problems that optimize the control system responses and reduce the sensitivity with respect to parametric variations of the controlled process. The objective functions are expressed as integral quadratic performance indices that depend on the control error and on the squared output sensitivity functions. The PSO-based and SA-based minimization of the objective functions has as direct result the optimal tuning parameters of the controllers. The new optimization problems use the advantages of nature-inspired optimization algorithms to improve the control systems performance indices. This chapter contains recommendations for the practitioners that contribute to practical implementations with good computational efficency and fast convergence rate.

A comparative study of ant colony, differential evolution, particle swarm optimization and the neighbourhood algorithms for history matching of reservoir simulation models is provided in the chapter "Comparison of Evolutionary and Swarm Intelligence Methods for History Matching and Uncertainty Quantification in Petroleum Reservoir Models." In history matching, simulation models are calibrated to reproduce the historical observations from the oil and gas fields. History matching is an inverse problem with non-unique solution. Multiple history matched reservoir models are used to quantify uncertainty of future hydrocarbon production from a field. In our assisted history matching workflow, different evolutionary and swarm intelligence algorithms are use to explore the plausible parameter space and find good-fitting reservoir models. These algorithms are also integrated within a Bayesian framework to quantify uncertainty of the predictions. Two petroleum reservoir cases illustrate different aspects of this comparative study. The results present comparison of best history-matched models, convergence speeds for different algorithms, ability of the algorithms in navigating the search space and their effect on uncertainty of the predictions for ultimate oil recovery.

A contribution from the general field of logistics can be found in the chapter "Optimization of Multiple Traveling Salesmen Problem by a Novel Representation based Genetic Algorithm." The aim of logistics is to get the right materials to the right place at the right time, while optimizing a given performance measure and satisfying a given set of constraints. In most distribution systems goods are transported from various origins to various destinations. It is often economical to consolidate the shipments of various origin-destination pairs and transport such consolidated shipments in the same truck at the same time. Obviously the challenge is to find the optimal i.e. the best consolidation according to some objective functions.

In case of this chapter, the problem is an asymmetric multiple Traveling Salesman Problem with Time Windows, where additional special constraints exist, like an upper bound for the number of salesmen, the maximum travelling distance, or a time window at each location. This is a numerical optimization problem, obviously an NP-hard task.

The main motivation of the research presented in this chapter was that there was no available algorithm that is "intelligent" enough to handle constraints on tour lengths, asymmetric distances, and the number of salesmen is not predefined, and can vary during the evolution of the individual solutions. Furthermore the representation is so transparent that supports not only the implementation, but the initialization and heuristic fine-tuning of the individual routes.

In this chapter, authors propose a general, novel genetic representation for the so-called mTSP problems. A new genetic algorithm based on the novel representation is presented, which can handle the constraints for the routes and the time windows for the locations too, as well as its representation is more similar for the characteristic of the problem than the ones until now. Thus it can be more easily understandable and realizable. The algorithm was implemented in MATLAB and integrated with Google Maps to provide a complete framework for distance calculation, definition of the initial routes and visualization of the resulted solutions. The novel approach and the novel representation proved to be more effective in terms of flexibility, complexity and transparency, and also in efficiency than the previous methods.

In the chapter "Out-of-the-box and Custom Implementation of Metaheuristics. A Case Study: The Vehicle Routing Problem with Stochastic Demand" authors propose an experimental analysis that studies the impact of development effort on the relative performance of metaheuristics. Five algorithms for the vehicle routing problem with stochastic demand that have been proposed in the literature are considered: Tabu Search, Simulated Annealing, Genetic Algorithms, Iterated Local Search and Ant Colony Optimization. As measure of the development effort the time devoted to tune the parameters of the algorithms is considered. In this way, such effort can be easily measured. If such a minor implementation issue allows to point out a significant difference in the relative behavior of metaheuristics, then this difference can be expected to grow when one considers other issues. The same algorithms in their out-of-the-box version, i.e., with no parameter tuning, and in their custom one, i.e., with fine-tuned parameters are compared. The results support the main claim of the chapter: one should clearly state in which context one develops algorithms, since the results obtained in an out-of-the-box context are not necessarily extendable to a custom one, and vice versa. Moreover, in experimental analysis we often consider also the values of parameters indicated in the paper in which the algorithm was proposed. This analysis allows one to observe that the results that are reported in the literature cannot be a priori related to one of the two contexts.

Optimal analogue circuit sizing is investigated in the chapter "Analogue Circuit Optimization Through a Hybrid Approach." It is shown that hybridization of a global optimization approach with a local one leads to better results in optimization of such circuits than using classical approaches. The case of merging Genetic

Algorithms with the Simulated Annealing technique is considered. The hybrid algorithm is detailed and is evaluated using test functions. It is shown through three application examples, i.e. optimization of performances of a current conveyor, an operational transconductance amplifier and a low noise amplifier, that such hybrid algorithms yield optimal solutions in a much shorter time, when compared to conventional meta-heuristics.

The scope of the chapter "Evolutionary Inventory Control for Multi-Echelon Systems" is confined to the use of Genetic Algorithms (GA)s for handling operational issues of inventory control and management in multi-echelon inventory networks. It provides an extensive review of literature on use of GAs to solve multi-echelon inventory control problems and evaluates the state of GA applications in these areas. The chapter also presents a novel GA structure for a stochastic lot sizing problem in a centralized distribution system. Numerical experiments conducted on several test cases with different operational parameters show that proposed GA structure can be used as an effective algorithm for solving the multi-echelon inventory distribution problem under stochastic demand.

To improve the ride comfort is one of the most important design objectives in automotive engineering by reducing the vibration transmission and keeping proper tyre contact. This is the main topic of the chapter "Fuzzy Skyhook Surface Control using Micro-Genetic Algorithm for Vehicle Suspension Ride Comfort." The semi-active suspension systems are developed to achieve a better ride comfort performance than the passive suspension system. A polynomial function supervised fuzzy sliding mode control collaborated with a skyhook surface method is introduced for the ride comfort of a two degree of freedom vehicle semi-active suspension. The multi-objective micro-genetic algorithm has been utilised to this proposed controller's parameter alignment in a training process with three ride comfort objectives. The numerical results have shown that this hybrid control method is able to provide a real-time enhanced level of ride comfort performance for the semi-active suspension system.

References

1. Bullheimer, B., Hartl, R.F., Strauss, C.: A new rank-based version of the Ant System: A computational study. Central European Journal for Operations Research and Economics 7(1), 25–38 (1999)
2. Dorigo, M.: Optimization, Learning and Natural Algorithms. PhD Thesis, Politecnico di Milano, Italy (1992)
3. Di Caro, G., Dorigo, M.: AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)
4. Dorigo, M., Di Caro, G.D., Gambardella, L.M.: Ant Algorithms for Discrete Optimization. Artificial Life 5, 137–172 (1999)
5. Dorigo, M., Maniezzo, V., Colorni, A.: Positive Feedback as a Search Strategy, Technical Report No 91-016, Politecnico di Milano (1991)
6. Dorigo, M., Gambardella, L.: Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)

7. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
8. Edelbaum, T.N.: Theory of Maxima and Minima. In: Leitmann (ed.) Optimization Techniques with Applications to Aerospace Systems, p. 132. Academic Press, New York (1962)
9. Fenton, N., Hill, G.: Systems construction and analysis: a mathematical and logical framework. McGraw-Hill, New York (1993)
10. Fonseca, C.M., Fleming, P.J.: An Overview of Evolutional Algorithms in Multiobjective Optimization. Evolutionary Computation (31), 1–16 (1995)
11. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
12. Gaertner, D., Clark, K.: On Optimal Parameters for Ant Colony Optimization algorithms. In: Proceedings of the International Conference on Artificial Intelligence 2005, Las Vegas, USA, pp. 83–89 (2005)
13. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
14. Goss, S., Aron, S., Deneubourg, J.L., Pasteels, J.M.: Self-organized shortcuts in the Argentine ant. Naturwissenschaften 76, 579–581 (1989)
15. Gutjahr, W.: A graph-based ant system and its convergence. Future Generation Computer Systems 16(8), 873–888 (2000)
16. Kennedy, J., Eberhart, E.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE Press, Los Alamitos (1995)
17. Michel, R., Middendorf, M.: An Island Model Based Ant System with Lookahead for the Shortest Supersequence Problem. In: Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature, Amsterdam, The Netherlands, pp. 692–701 (1998)
18. Shmygelska, A., Hoos, H.H.: An Ant Colony Optimisation Algorithm for the 2D and 3D Hydrophobic Polar Protein Folding Problem. BMC Bioinformatics 6, 6–30 (2005)
19. Stützle, T., Dorigo, M.: A short convergence proof for a class of ant colony optimisation algorithms. IEEE Transactions on Evolutionary Computation 6(4), 358–365 (2002)
20. Stützle, T., Hoos, H.H.: MAX-MIN Ant System. Future Generation Computer Systems 16(8), 889–914 (2000)
21. Schwefel, H.-P.: Evolution and Optimum Seeking. John Wiley & Son, Inc., New York (1995)
22. Syswerda, G.: Uniform Crossover in Genetic Algorithms. In: Proceedings of International Conference on Genetic Algorithm 1989 (ICGA 1989), pp. 2–9 (1989)
23. Wright, S.: Evolution in Mendelian populations. Genetics 16, 97–159 (1931)

Part I

Frameworks

Learning Automata in Control Planning Strategies

Erik Cuevas, Daniel Zaldivar, Marco Perez-Cisneros, and Raúl Rojas

Abstract. Intelligent Computational Optimization has been successfully applied to several control approaches. For instance, Planning Control uses information regarding a problem and its environment to decide whether a plan is the most suitable to achieve a required control objective or not. Such algorithm is commonly embedded into a conveniently located model inside a control loop. Planning provides a general and easy methodology widely used by a number of approaches such as receding horizon control (RHC) and model predictive control (MPC). Actually, MPC is the planning approach that has recently acknowledged a wide acceptance for industrial applications despite being highly constrained by lits computational complexity. For MPC, the evaluation of the overall plan is based upon time-consuming approaches such as dynamic programming and gradient-like methods. This chapter explores the usefulness of planning in order to improve the performance of feedback-based control schemes considering one probabilistic approach known as the Learning Automata (LA). Standard gradient methods develop a plan evaluation scheme whose solution lies on a neighbourhood distance from the previous point, forcing to explore the space extensively. Remarkably, LA algorithms are based on stochastic principles considering newer points for optimization as being determined by a probability function with no constraints whatsoever on how close they lie from previous optimization points. The proposed LA approach is considered as a planning system to select the plan holding the highest probability of yielding the best closed-loop results. The system's performance is tested through a nonlinear benchmark plant, comparing its results to the Levenberg-Marquardt (LM) algorithm and some other Genetic algorithms (GA).

Erik Cuevas · Daniel Zaldivar · Marco Perez-Cisneros
CUCEI, Universidad de Guadalajara
Av. Revolución No. 1500, C.P. 44430, Guadalajara, Jal., México
e-mail: {erik.cuevas,daniel.zaldivar,marco.perez}@cucei.udg.mx

Raúl Rojas
Institut für Informatik, Freie Universität Berlin
Takustrasse 9, PLZ 14195, Berlin, Germany
e-mail: rojas@inf.fu-berlin.de

Keywords: Reinforcement learning; Learning systems; Model Predictive Control; Learning Automata; CARLA.

1 Introduction

Advances in computational intelligence have brought new opportunities and challenges for researchers seeking for new ways to deal with complex and uncertain systems. In Engineering, many systems are too complex to be represented by an accurate mathematical model but still they demand the use of other approaches for designing, optimizing or controlling their behaviour. In recent years, several computational intelligence based techniques have emerged as successful tools for solving difficult optimization problems commonly mishandled by traditional optimization methods.

The presence of nonlinearities is commonly the main challenge. They impose several conditions to most industrial processes including actuator nonlinearities such as saturations, dead-zones and backlash. On the other hand, model inaccuracy also imposes hard constraints when a given mathematical model cannot exactly reproduce the plant's behaviour.

According to the control framework, planning requires the ability to build representations similar to daily-life models. In turn, this fact allows generating predictions on how the environment would react to several plans. The ability of choosing among different alternative plans and executing among several sequences of actions has been mastered, almost exclusively by humans. Planning is the approach which allows generating complex behaviours surpassing the simple reaction to what is sensed. Moreover, Planning Control uses information about the problem and its environment, often embedded into some type of a model which considers many options, also known as plans. It aims to choose the best plan in order to achieve the required objectives in the control loop.

Planning also provides a very general and easy methodology to apply. It has been exploited extensively in conventional control, e.g. receding horizon control and model predictive control. In comparison to intelligent approaches such as neural networks (Liu, 2001) or evolutionary algorithms (Fleming & Purshouse, 2002), it exploits the use of an explicit approximated model to decide what actions to take. However, like the fuzzy and expert system approaches, it is still possible to incorporate heuristics to specify which control actions are the best to use. In broad sense, planning approaches attempt to use both heuristic knowledge and model-based decisions in order to exert control. It is the fundamental reason for selecting a planning strategy over a simple rule-based system. It is a bad engineering practice to prefer the use of heuristics and ignore the information provided by a good mathematical model considering that planning strategies provide a way to incorporate this information.

Planning has been successfully applied to solve several engineering problems (Ying-Pin et al., 2009; Huang, 2009), despite only few examples portraying its application to control dynamical systems (Chauvin et al., 2008 and Son, 2006). Several planning system approaches may be considered depending upon the problem and the number of plans considered by the solution. An classic example is the

Belief-Desire-Intention method (Seow & Sim, 2008), an effective scheme for finite and sensibly small plan number, which unfortunately constrains its use for control.

The Model Predictive Control (Camacho & Bordons, 2008) is the planning approach that has recently acknowledged a wide acceptance for industrial applications. The control signal generation in MPC involves the on-line use of one parametric plant model, assuming an efficient control plan. Major design techniques of MPC include Model Algorithm Control, Dynamic Matrix Control, Internal Model Control and Generalized Predictive Control, among others (Garcia et al., 1989). The strategy of MPC is, at any given time, to solve on-the-fly a receding open-loop optimal control problem over a finite time horizon, taking only the first result in the control sequence. MPC algorithms are very intuitive and easy to understand with practical constraints commonly imposed to the on-line algorithm (Mayne et al., 2000). MPC has received worldwide attention because of its simple implementation on industrial applications. In particular, chemical processes have shown a relatively slow dynamics which may easily accommodate the on-line optimization (Garcia et al., 1989).

Several variants of the MPC methodology have been published. In Camacho & Bordons (2007), the plan evaluation is done over non-linear models while Nagy et al., (2007) have applied a similar approach to an industrial batch reactor. Furthermore, the idea of mixing iterative learning control to feedback model-based control is discussed by Cueli & Bordons (2008). The use of Set-Membership (SM) methodologies for approximating Model Predictive Control schemes (MPC) and their laws for linear systems has been recently proposed in Canale et al., (2009). Predictive control has demonstrated an excellent performance for both theoretical studies and industrial applications. However, its deployment for controlling non-linear processes is complicated as the algorithm limits the kind of functions which can be effectively minimized by the optimization method.

However, much of the work has been limited to optimization strategies (for plan evaluation and selection) which are based on dynamic programming or gradient methods. The use of such optimization techniques for non-linear control problems is multimodal, yielding a slow speed operation and a high computational complexity. This chapter explains the use of a stochastic approach known as Learning Automata (LA) to overcome such problems.

Few works have been reported using some stochastic methodology either to incorporate LA into MPC or to generate a planning structure. Some exceptions are reported by Potočnik et al. (2008) whose work considers a probabilistic neural-network as part of a MPC system, and by Chen et al. (2009) or Nagya et al. (2001), both reporting a Genetic algorithm as optimization method.

The Learning Automata (LA) (Narendra & Thathachar, 1989) is an adaptive decision making method that operates within unknown random environments while progressively improving its performance via a learning process. LA is very useful for optimization of multi-modal functions, in particular when such a function is unknown and only noise-corrupted evaluations are available (Beigy & Meybodi, 2006). For such cases, a probability density function, which is defined over the parameter (action) space, is used to select the next point. The reinforcement signal (objective function) and the learning algorithm are used by the LA to update the probability density function at each stage. Such automaton improves its performance to obtain an optimal parameter (action). Therefore, the parameter showing the highest probability would correspond to a minimum as it has been demonstrated through rigorous proofs of convergence by Narendra & Thathachar (1989), Najim & Poznyak (1994), Thathachar & Sastry (2004) and Beigy & Meybodi (2009).

The LA method does not need knowledge of the environment or any other analytical reference to the function to be optimized. It is actually its main advantage. Additionally, it offers fast convergence for estimation of several parameters (Torkestani & Meybodi, 2010). Other Gradient-based methods, such as the LM, require iterative updating procedures within the parameter space that usually exhibit a slow convergence or local minima trapping (Park et al., 2000). The LA's search for the optimum is performed over a probability space rather than seeking through the parameter space as it is commonly done by gradient optimization algorithms (Meybodi & Beigy, 2002). Opposite to well-known Genetic Algorithms (GA) which commonly bias the whole chromosome population towards the best candidate solution exclusively (see Gao et al., 2009), LA can effectively handle challenging multimodal optimization tasks by means of effectively exploring the search space (Ikonen & Najimz, 2008).

LA has been used for solving different sorts of engineering problems at several fields such as pattern recognition, adaptive control (Zeng et al., 2000), signal processing (Howell & Gordon, 2001), power systems (Wu, 1995) and computer networks (Torkestani & Meybodi, 2010). Some effective algorithms have been lately proposed for multimodal complex function optimization based on the LA (see (Howell & Gordon, 2001; Thathachar & Sastry, 2002; Zeng & Liu, 2005; Beygi & Meybodi, 2006; Beigy & Meybodi, 2009)). Furthermore, it has been shown experimentally that the performance of such optimization algorithms may surpass the genetic algorithm (GA) as they reduce the searching space yielding a fast convergence (see for instance, Zeng & Liu (2005)). This chapter discusses the use of the *continuous action reinforcement learning automata* (CARLA) as the chosen LA approach.

The CARLA algorithm was first introduced by Howell, Frost, Gordon and Wu (1997). It has been demonstrated its effectiveness to solve some optimization tasks for a wide range of applications. In Howell et al., 2000, the CARLA algorithm is used to simultaneously perform on-line tuning of an PID-controller which has been applied to an engine idle-speed system. On the other hand, Kashki et al., 2008, have shown experimentally that CARLA's performance for tuning PID coefficients is superior to the performance shown by the Genetic algorithms (GA) and the Particle Swarm Optimization (PSO) operating over the same problem.

This chapter also discusses how to emulate the functionality of planning in order to decide how to control a plant. The study focuses on typical plants considered in conventional control. The planning strategy is the MPC methodology, incorporating Learning Automata as the optimization algorithm. The use of a stochastic approach deals appropriately with the multimodal problem of the error surface as it accelerates the computation process and eliminates the controller complexity. The algorithm's performance is measured over a well-known non-linear process: the surge-tank plant. The solution is compared to the Levenberg-Marquardt algorithm

(Kelley, 2000) and one Genetic Algorithm (Chen et al., 2009). The LM algorithm has been chosen because it has been regarded as the most popular for planning strategies showing a fair balance between precision and speed. In the same sense, GA is a well-known stochastic optimization methodology.

The chapter is organized as follows: Section 2 presents a brief review on control planning strategies while section 3 discusses the foundations and theory of Learning Automata. In Section 4, the LA approach is implemented using the surge tank plant as a non-linear example. In Section 5, experimental results are presented while Section 6 concludes the chapter.

2 Planning Strategy Design

The concept of planning is commonly understood following common sense such as the case when humans plan their activities for the weekend or when a solution for a daily-life problem is discussed among them. The solution normally arises from a collection of actions to be followed aiming to achieve specific goals. Such kind of action-sorting can be named as an *action plan* and may fall into the following planning steps:

1. **Planning domain.** Refers to the first representation of the problem to be solved. (i.e. a model).
2. **Setting goals.** Essential to planning to define the required behaviour or overall aims.
3. **Sticking to the plan.** Considering that sometimes humans simply react to situations with no considerations about the consequences of their actions. For the scope of this chapter, it would be better to fully develop a plan by reaching the goals completely.
4. **Selecting a strategy.** The selection of the plan commonly involves projections into the future by means of a model. It requires considering a variety of sequences of task and sub-goals to be executed. An optimization algorithm is required to choose the best plan to be followed by assuming a partial model of the problem.
5. **Executing the plan.** After the selection, it must be decided how to execute that plan.

It is important to consider that the chapter focuses on plants that are typically considered for conventional control. In the approach, planning systems are considered as computer programs that emulate the way in that experts may do planning in order to solve a given control problem. The following section discusses on several issues regarding the model, the plan generation and the selection process.

2.1 Closed-Loop Planning Configuration

A generic planning system can be set on the architecture of a standard control system as it is shown by Figure 1. According to the human planning and solving framework, the problem domain is the plant and its environment. There are measured outputs $y(k)$ which are variables of the problem domain that are obtained at step k, control actions $u(k)$ which can affect the problem domain, disturbances $d(k)$ which represent random events that can affect the problem domain and hence the measured variable $y(k)$, and the goal $r(k)$ which is called the reference input in conventional control terminology as it represents what is to be achieved within the problem domain. There are closed-loop specifications to define the performance and stability requirements. The types of plants which are considered in this section are defined as follows:

$$y(k+1) = f(x(k), u(k), d(k)) \tag{1}$$

where $y(k)$ is the measured output and f is a generally unknown smooth function of the state $u(k)$, the measurable state is $x(k)$ and the disturbance $d(k)$.

$$x(k) = [y(k), y(k-1), \ldots, y(k-p), u(k-1), u(k-2), \ldots, u(k-q)]^T \tag{2}$$

where p and q represent the system order. The system is therefore considered to be causal, yielding $y(k-p) = 0$ and $u(k-q) = 0$, if $k<p$ or $k<q$.

Let

$$e(k) = |r(k) - y(k)| \tag{3}$$

Equation (3) is also known as the tracking error. Generally, the objective is to always make the tracking error as small as possible as it asymptotically approaches zero forcing the output to follow the reference input.

Considering a plan to be a sequence of possible control inputs and the i^{th} plan of length N at time k being structured as follows

$$u^i[k, N] = u^i(k,0), u^i(k,1), \ldots, u^i(k, N-1) \tag{4}$$

The algorithm aims to develop a controller that is based on the planning strategy. One model and the optimization method are used to evaluate and score each plan (e.g. MPC). This will in turn provide a quality ranking for each plan. The plan is thus chosen (plan $i*$) using the control input at each time instant k as follows:

$$u(k) = u^{i^*}(k,0) \tag{5}$$

The best plan $u^{i^*}[k, N]$ is chosen, using the *first* input from the control sequence as input to the plant. The process is repeated through each time step. Clearly, it is possible to use a lower frequency for the re-planning using for instance a new plan at each sampling step and executing the *first two* inputs from the optimal plan.

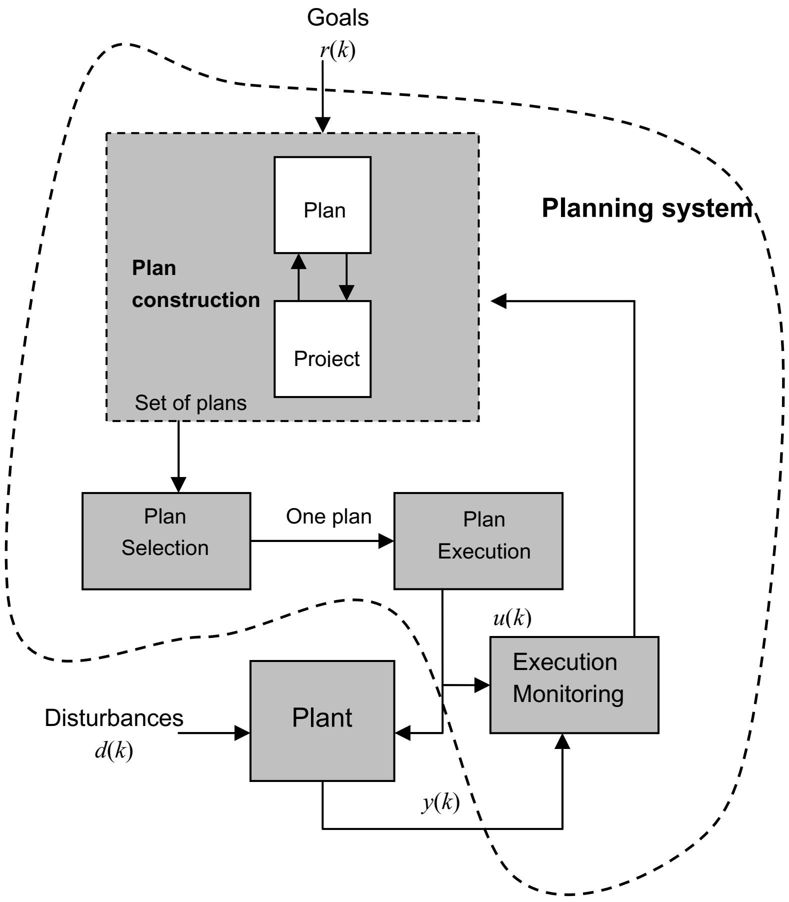


Fig. 1 Closed-Loop planning system.

Also, the use of a number of controllers may be an option to implement the planning system. The current state and the given reference input, both may be considered as a “plan template” which in turn represents a particular plan. An optimization algorithm may thus be used to evaluate the performance considering the approximated model of the plant that must also include uncertainty. Considering that a continuous interval for the parameters might generate an infinite number of plans, an optimization algorithm must be employed for finding the best plan for a particular situation.

2.2 Models and Projections into the Future

A wide range of models are available depending on the problem domain, the capabilities of the planner to store and use the model features and the goals to be achieved. For instance, a planning model could be continuous or discrete (e.g., a differential or difference equation) and it could be linear or nonlinear. It may be deterministic or it may contain an explicit representation of the uncertainty of the problem, so plans may also be chosen considering such factors (Mayne et al., 2000).

There is no comprehensive model which can be able to fully represent the plant and the environment yielding uncertainty. Hence there is always a bound regarding the amount of time which is required to simulate the model far into the future. Such projection into the future may become useless after some time as it may go too far as a result of inaccurate predictions which yield poor information on how to select the best plan. The difficulty emerges from knowing how good the model is and how far it may be projected into the future. In this chapter a deeper analysis on such problems will not be considered.

Considering a general nonlinear discrete time model as:

$$y(j+1) = f(x(j), u(j)) \tag{6}$$

being $y(j+1)$ the output, $x(j)$ the state and $u(j)$ the input for j = 0,1,2,..., N-1. Notice that this model can be quite general, if required. However in practical terms, only a linear model is generally available and may be sufficient. Let $y^i(k,j)$ denote the j^{th} value at time k using the i^{th} plan defined by $u^i[k,N]$, and the state $x(k,j)$. In order to predict the effect of plan i (as it is projected into the future) at each time k, it is required to calculate a step-set ahead considering j= 0,1,2,..., N-1 , as follows:

$$y^i(k, j+1) = f(x(k, j), u^i(k, j)) \tag{7}$$

Considering a simulation forward in time k, for j = 0, it begins with $x(k,0) = x(k)$ generating $y(k, j+1)$ with j= 0,1,2,..., N- 1. It is required to appropriately shift values in x at each step yielding values of $u^i(k, j)$, j = 1,2,..., N-1, for each i.

2.3 Optimization Procedure and Plan Selection Method

The set of plans (strategies) is "pruned" to only one which is considered as the best one to be applied at the current time as optimization is very important for planning. The specific type of optimization approach that is used for plan selection should be able to operate in multimodal surfaces, showing a light and fast computation. The previous requirements are usually difficult to solve by means of t raditional optimization algorithms, yielding relevance for the use of the LA as an optimization procedure.

Prior to the optimization procedure selection, it is necessary to define a specific criterion to decide the best plan. Although there exist different performance criteria (Bloemen et al., 2004), a cost function of the type $J(u^i[k,N])$ is used at this chapter to quantify the quality of each candidate plan $u^i[k,n]$ by means of the model *f*. First, it is assumed that the reference input *r*(*k*) is either known all the time or at least at time *k*, while it is also known up to the time *k* + *N*. Therefore, the cost function is defined as follows:

$$J(u^i[k,N]) = \omega_1 \sum_{j=1}^{N} (r(k+j) - y_m^i(k,j))^2 + \omega_2 \sum_{j=1}^{N-1} (u^i(k,j))^2 \tag{8}$$

being $\omega_1 > 0$ and $\omega_2 > 0$ the scaling factors for weighting the importance of reducing the tracking error (first term) or minimizing the use of control energy (second term) to reduce the tracking error. Often ω_1 and ω_2 hold similar values. In order to specify the control at time *k,* it is necessary to take the best plan, as it is measured by $J(u^i[k,N])$, calling it the plan $u^{i^*}[k,N]$, and generating the control using $u(k) = u^{i^*}(k,0)$ (i.e. the first control input in the sequence of inputs which is the best).

An important consideration is therefore the selection of the optimization method that will converge to the optimal plan and the choosing of one that can cope with the complexity presented by a large number of candidate plans. First, by focusing on the complexity aspect, it should be noticed that the inputs and states for the plant under consideration can take on a continuum number of values, despite of particular applications which may only consider a finite number of values. This is the case for analog-control systems in particular considering actuator saturation. For digital control systems, one data acquisition scheme may be available, yet hosting some quantization and theoretically yielding a finite number of inputs, states, and outputs, for the model f_m. Digital computers are commonly used despite the fact that the number of operations may be *very* large. In general, there exist an infinite number of possible plans that must compute their own cost, ranking them according to such cost and hence selecting the best one.

If non-linear and uncertain system characteristics dominate to the extent that a linear model is not sufficient for generating plans, then a nonlinear model can be used within the planner. Some type of nonlinear optimization method may therefore be used for the parameters that evaluate the infinite set of feasible plans. However, this may become a troublesome task since a non-linear model is used for plan generation. In turn, it forces the overall solution to consider non-linear optimization with generally no analytical solution available.

There exists a wide variety of algorithms to tackle this problem such as steepest descent, Levenberg-Marquardt, etc. Such methods, however, do not guarantee convergence to an optimal plan or they may get stuck into local minima, generating divergent solutions or even not reaching one at all. Therefore the resulting plan after the non-linear optimization procedure cannot be assured to yield the optimal closed-loop performance. It is important to recall that for some practical industrial problems, engineers have managed to develop effective solutions via such a nonlinear optimization approach. This fact has given way to the main motivation beneath the use of LA as an optimization algorithm because it offers global optimization when dealing with multimodal surfaces. The search for the optimum is done within a probability space rather than seeking within a parameter space as it is done by other optimization algorithms. An Automata is commonly understood as an automaton, acting embedded into an unknown random environment and improving its performance to obtain an optimal action.

3 Learning Automata

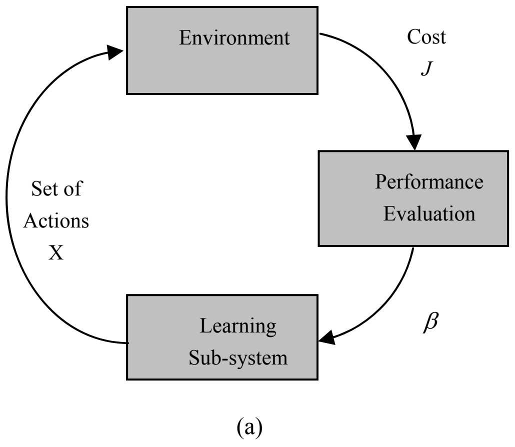
The concept of learning automata was first introduced by the pioneering work of Tsetlin (Tsetlin, 1973). He was interested into the behaviour modelling of biological systems and subsequent research has considered the use of such learning paradigm for engineering systems. Although LA and reinforcement learning aim to solve similar problems, their methodologies and algorithms greatly differ (Thathachar & Sastry, 2002). LA operates by selecting actions via a stochastic process. Such actions operate within an environment while being assessed according to a measure of the system performance. Figure 2a shows the typical learning system architecture. The automaton probabilistically selects an action (**X**). Such actions are applied to the environment, and the performance evaluation function provides a reinforcement signal β. In turn, such signal is used to update the automaton's internal probability distribution whereby actions that achieve desirable performance are reinforced via an increased probability, while those not-performing actions are penalised or left unchanged depending on the particular learning rule which has been employed. Over time, the average performance of the system will improve until a given limit is reached. In terms of optimization problems, the action with the highest probability would correspond to the global minimum as demonstrated by rigorous proofs of convergence available in Narendra & Thathachar (1989), Najim & Poznyak (1994), Thathachar & Sastry (2004) and Beigy & Meybodi (2009).

A wide variety of learning rules have been reported in the literature. One of the most widely used algorithms is the linear reward/inaction (L_{RI}) scheme, which has been shown to guarantee convergence properties (see (Narendra & Thathachar, 1989)). In response to action $\mathbf{x}_i$, being selected at time step k, the probabilities are updated as follows:

$$\begin{aligned} p_i(k+1) &= p_i(k) + \theta \cdot \beta(k) \cdot (1 - p_i(k)) \\ p_j(k+1) &= p_j(k) - \theta \cdot \beta(k) \cdot p_j(k)\,, \text{ if } i \neq j \end{aligned} \tag{9}$$

being θ a learning rate parameter $0 < \theta < 1$ and $\beta \in [0,1]$ the reinforcement signal; $\beta = 1$ indicates the maximum reward and $\beta = 0$ is a null reward. Eventually, the probability of successful actions will increase to become close to unity. In case that a single and foremost successful action prevails, the automaton is deemed to have converged.

Considering a large number of discrete actions, the probability of selecting any particular action becomes low and the convergence time can become excessive. In order to avoid such situation, the automata can be connected into a parallel setup



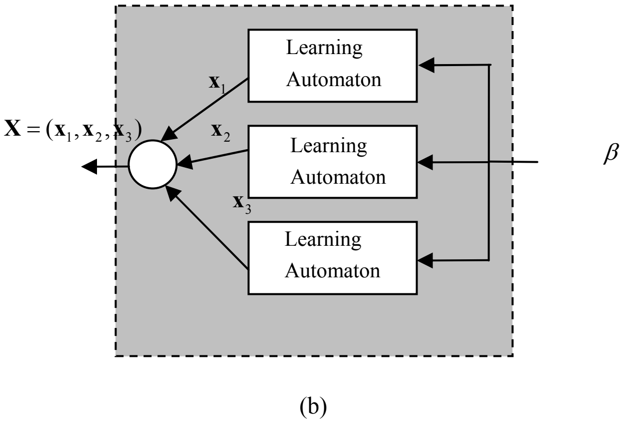


Fig. 2 (a) Reinforcement learning system and (b) Parallel interconnected automata.

as it is shown by Figure 2b. Each automaton operates a small number of actions and the 'team' works together in co-operative manner. This scheme can also be used if multiple actions are required.

Discrete stochastic learning automata can be used to determine global optimal states for control applications with multi-modal mean square error surfaces. However, the discrete nature of the automata requires the discretization of a continuous parameter space, and the level of quantization tends to reduce the convergence rate. A sequential approach may be adopted (Howell & Gordon, 2001) to overcome such problem by means of an initial coarse quantization. It may be later refined using a re-quantization around the most successful action. In this chapter, an inherently continuous form of the learning automaton is used to speed the learning process avoiding its own complexity.

3.1 CARLA Algorithm

The continuous action reinforcement learning automata (CARLA) was developed as an extension of the discrete stochastic learning automata for applications involving searching of continuous action space in a random environment (Howell & Gordon, 2001). Several CARLA can be arranged in parallel just like the discrete automata (Figure 2b) searching multidimensional action spaces. As each CARLA algorithm operates on independent actions, the automata set runs within a parallel implementation defining several parameter values (see Fig. 2b). Communication between several CARLA's algorithms is done through the environment and one performance evaluation function.

The automaton's discrete probability distribution is replaced by a continuous probability density function which is used as the basis for action selection. It operates a reward/inaction learning rule similar to the discrete learning automata. Successful actions receive an increase on their probability for future selection via a Gaussian neighbourhood function. The probability density is thus increased within the vecinity of such a successful action. The initial probability distribution may be equally probable over a desired range, yielding numerous iterations and converging to a Gaussian distribution around the best action value.

If action x is defined over the range $(x_{\min}, x_{\max})$, the probability density function $f(x,n)$ at iteration n is updated according to the following rule:

$$f(x,n+1) = \begin{cases} \alpha \cdot [f(x,n) + \beta(n) \cdot H(x,r)] & \text{if } x \in (x_{\min}, x_{\max}) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

With α being chosen to re-normalize the distribution according to the following condition

$$\int_{x_{\min}}^{x_{\max}} f(x,n+1)dx = 1 \tag{11}$$

with $\beta(n)$ being the reinforcement signal in the performance evaluation and $H(x,r)$ a symmetric Gaussian neighbourhood function centred on $r = x(n)$. It yields

$$H(x,r) = \lambda \cdot \exp\left(-\frac{(x-r)^2}{2\sigma^2} \right) \tag{12}$$

Where λ and σ are parameters that determine the height and width of the neighbourhood function. They are defined in terms of the range of actions as follows:

$$\sigma = g_w \cdot (x_{\max} - x_{\min}) \tag{13}$$

$$\lambda = \frac{g_h}{(x_{\max} - x_{\min})} \tag{14}$$

where the value g_w controls the width and the parameter g_h sets the height of the Gaussian function which is added to the distribution (Equation 10). The speed and resolution of the learning process are thus controlled by the free parameters g_w and g_h. Such parameters are set experimentally as they depend on the system to be optimized.

Let action $x(n)$ be applied to the environment at iteration n, returning a cost or performance index $J(n)$. Current and previous costs values are stored within a vector $R(n)$ for computing the median and minimum values J_{med} and $J_{\min}$. Both values are required to calculate $\beta(n)$ as follows:

$$\beta(n) = \max\left\{0, \frac{J_{\text{med}} - J(n)}{J_{\text{med}} - J_{\min}}\right\} \tag{15}$$

To avoid problems with infinite storage and to allow the system to adapt to changing environments, only the last m values of the cost functions are stored in $R(n)$. Equation (15) limits $\beta(n)$ to values between 0 and 1 and only returns nonzero values for costs which lie below the median value. It is easy to understand how $\beta(n)$ affects the learning process as follows: during the learning, the performance and the number of selecting actions can be wildly variable generating extremely high computational costs. However, $\beta(n)$ is insensitive to such extremes and to extreme values of $J(n)$ resulting from a poor choice of actions. As learning continues, the automaton converges towards more worthy regions of the parameter space as the actions within such regions are chosen for evaluation increasingly often. As more of such responses are being received, J_{med} gets reduced. Decreasing J_{med} in the performance index effectively enables the automaton to refine its reference around the better responses previously received. Hence, it yields a better discrimination between the competing selected actions.

In order to define an action value $x(n)$ which has been associated to this probability density function, an uniformly distributed pseudo-random number $z(n)$ is generated within the range of [0, 1]. Simple interpolation is then employed to equate such value to the cumulative distribution function:

$$\int_{x_{\min}}^{x(n)} f(x,n)dx = z(n) \tag{16}$$

For CARLA optimization methods, the probability density function is associated to each decision variable. It is through modification of such probability density functions and a sufficient number of iterations, that the optimal value of the decision variables is determined. At each step, the modification process is trigged by the reinforcement signal $\beta(n)$ that corresponds to a predefined cost function. For implementation purposes, the distribution is stored at discrete points with an equal inter-sample probability. Linear interpolation is used to determine values at intermediate positions (see full details in (Howell & Gordon, 2001)).

4 Implementation

The proposed approach represents the overall planning system based on approximated models of the plant. Therefore, several plans may be available and the election of the best plan must be defined by the LA through considerations on the performance of the approximate model and the prospective results for some future instants. The election of each plan is made at each sampling instant *k*, just as it is discussed in sub-section 2.3. In the following section, the proposed planning strategy is applied to a conventional control plant commonly known as the "surge tank". The discussion begins by introducing the control problem which later moves to the designing and testing of the planning strategy.

4.1 Level Control in a Surge Tank

Consider the "surge tank," shown in Figure 3 modelled by

$$\frac{dh(t)}{dt} = -\frac{\bar{d}\cdot\sqrt{2gh(t)}}{A(h(t))} + \frac{\bar{c}}{A(h(t))}\cdot u(t) \tag{17}$$

where *u(t)* is the input flow (control input) which can be positive or negative (it can either pull liquid out of the tank or contribute to fill it in); *h(t)* is the liquid level (the output of the plant); $A(h(t)) = \left|\bar{a}\cdot h(t) + \bar{b}\right|$ is the cross-sectional area of the tank with $\bar{a} > 0$ and $\bar{b} > 0$ (their nominal values are $\bar{a}$ = 0.01 and $\bar{b}$ = 0.2); g = 9.81; $\bar{c} \in [0.9,1]$ is a "clogging factor" for a filter in the pump actuator with $\bar{c}$ = 0.9. There also exists some obstruction in the filter, however in case $\bar{c}$ = 1, the filter is clean so there is no clogging ($\bar{c}$ = 1 will be taken as its nominal value). $\bar{d}$ > 0 is a parameter related to the diameter of the output pipe (and its nominal value is $\bar{d}$ = 1). It is assumed that all these plant parameters are fixed but unknown.

Let *r(t)* be the desired level of the liquid in the tank (the reference input) and $e(t) = \left|r(t) - h(t)\right|$ be the tracking error (here *h*(*t*) is considered as the system's output *y*(*t*)). It is assumed that the reference trajectory is known in advance and *h(0)* = 1. In order to convert the problem to a discrete-time approach, the Euler approximation is used considering a sampling time of *T* = 0.1 seconds.

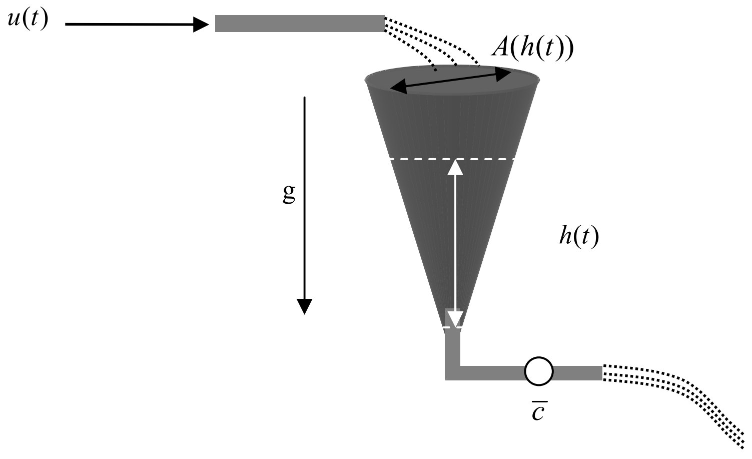


Fig. 3 The surge tank system

4.2 Planning System

For planning purposes, an uncertain and imprecise version of the nonlinear discrete-time model (the "truth model") is considered. The model (here referred as *m*) can be considered as an approximation of the problem over which several plans are to be tested. The candidate plans are generated using such a model while the evaluation follows the LA approach from last section. Taking the model from last subsection as the true plant model, the planning strategy considers a quite different cross-sectional area in comparison to the truth model in (17), yielding:

$$A_m(h(t)) = \overline{a}_m (h(t))^2 + \overline{b}_m \tag{18}$$

with $\overline{a}_m$ = 0.002 and $\overline{b}_m$ = 0.2. The same nonlinear equations in Eq. 16 are used assuming the values of $\overline{c}_m$ = 0.9 and $\overline{d}_m$ = 0.8. Figure 4 shows the cross-sectional area of the actual plant and the value considered in the model. There are evident differences between the real plant and its model which is used by the planning strategy.

In order to apply the planning methodology to the controlled plant, a simple proportional integral (PI) controller is considered. In particular, if $e(t) = |r(t) - h(t)|$, it yields

$$u(k) = K_p \cdot e(k) + K_i \cdot \sum_{j=0}^{k} e(j) \tag{19}$$

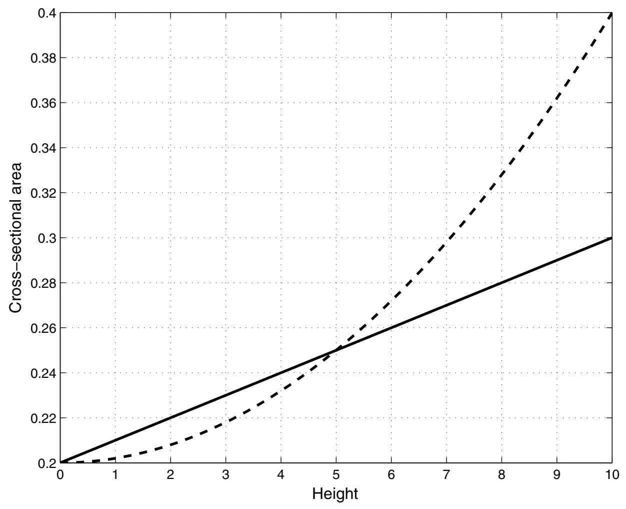


Fig. 4 Cross-sectional area *A(h)* for the actual plant (solid) and the projected model (dashed).

Each plan will be considered as a controller with two coefficients yielding an infinite number of plans as such variables are continuous. The complete structure of the planning system contains the model that evaluates each plan and the optimization system which determines the best coefficients for the values –they must match the acting indexes based on the error evaluation. Figure 5 shows a block representation of the system.

4.3 LA Optimization

Each plan yields a controller considering coefficients K_p and K_i. The problem thus focuses on finding the appropriate plan representing the couple of coefficients showing the best performance by using the projection of the control action. The intervals for each variables are chosen as $K_p \in [0, 0.2]$ and $K_i \in [0.15, 0.4]$. For instance, if the PI controller of Equation 19 is calibrated using conventional techniques to control the plant, the values K_p = 0.01 and K_i = 0.3 are assumed to be optimal. Figure 6 shows the controller performance. It is important to notice the fast response despite the overshoot.

K_p and K_i are not constant as they are calculated at each time instant *k* by means of the optimization system (LA in this chapter). The LA algorithm chooses the parameters K_p and K_i according to a probability distribution, projecting them into the planning strategy. The probability distribution should be modified if an inconvenient result emerges from the minimization period once it has finished according to the performance index in Equation 8. After several iterations, it must converge to a probability distribution around the optimal parameter value. Equation 8 rules the minimization representing twenty projections into the future (N = 20), $\omega_1 = 1$ and $\omega_2 = 1$, with the reference input remaining constant at each time.

In the optimization process, two LA (one for each parameter) are used. They are coupled only through the environment (model). The set R is limited to 10 values while only 50 iterations of CARLA are applied. The CARLA parameters are fixed at $g_w^{K_p} = 0.02$ and $g_h^{K_p}$ =0.3 for K_p, and $g_w^{K_i} = 0.02$ and $g_h^{K_i}$ for K_i.

Next, the overall CARLA algorithm for the optimization is described:

```
i    Set iteration $n$=0.
ii   Define the action set $A(n) = \{K_p, K_i\}$ such that $K_p \in [0, 0.2]$ and
     $K_i \in [0.15, 0.4]$
iii  Initialize $f(K_p, n)$ and $f(K_i, n)$ to a uniform distribution between the
     defined limits.
iv   Repeat while $n \leq 50$
     (a)  Use a pseudo-random number generator between 0 and 1, for
          each selected action $z_p(n)$ and $z_i(n)$.
     (b)  Select $K_p \in A(n)$ and $K_i \in A(n)$, considering that the area
          under the probability density function is
          $\int_0^{K_p(n)} f(K_p, n) = z_p(n)$ and $\int_{0.15}^{K_i(n)} f(K_i, n) = z_i(n)$.
     (c)  Project the control over a 20 discrete time intervals.
     (d)  Evaluate the performance using Equation (8).
     (e)  Append to $R$ and evaluate the minimum $J_{\min}$, and median,
          $J_{\text{med}}$, values of $R$, considering $m$=25.
     (f)  Evaluate $\beta(n)$ via Equation (15).
     (g)  Update the probability density functions $f(K_p, n)$ and
          $f(K_i, n)$ using Equation (10).
     (h)  Increment the iteration number $n$.
```

The learning system searches into a two-dimensional parameter space of K_p and K_i, aiming to reduce the values for J in Equation (8).

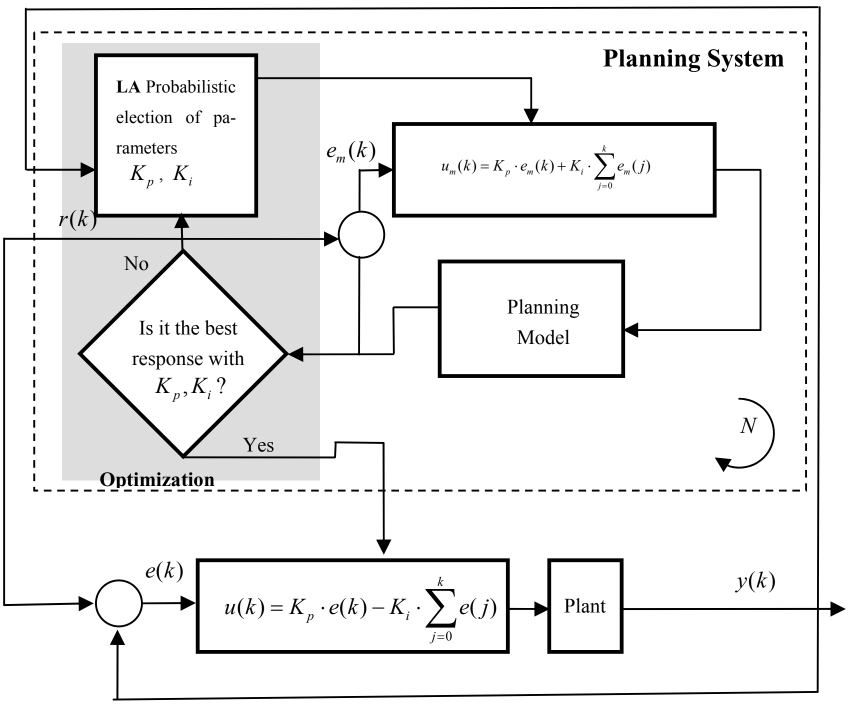

Planning System
LA Probabilistic election of parameters K_p, K_i
$e_m(k)$
$u_m(k) = K_p \cdot e_m(k) + K_i \cdot \sum_{j=0}^{k} e_m(j)$
$r(k)$
No
Is it the best response with K_p, K_i?
Planning Model
N
Yes
Optimization
$e(k)$
$u(k) = K_p \cdot e(k) - K_i \cdot \sum_{j=0}^{k} e(j)$
Plant
$y(k)$

Fig. 5 Block representation of the system including the planning strategy.

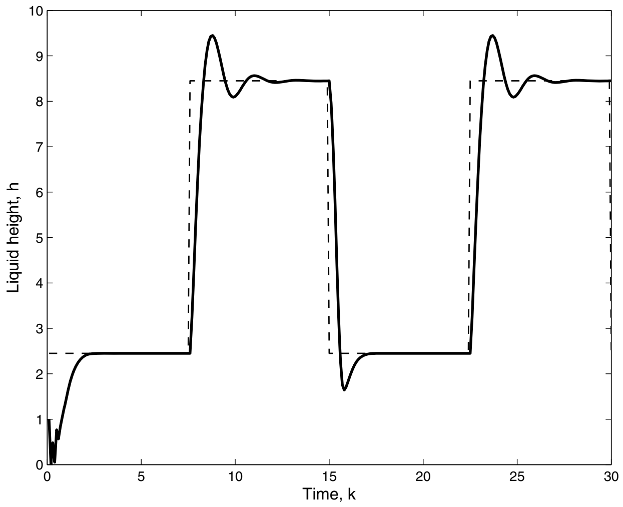
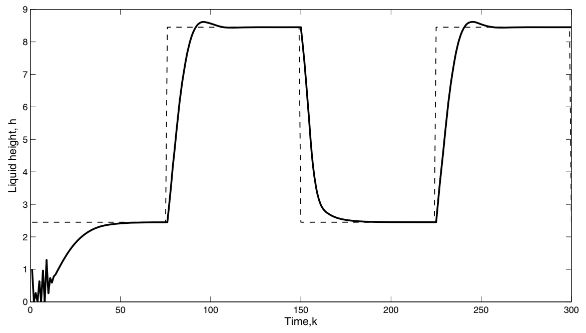


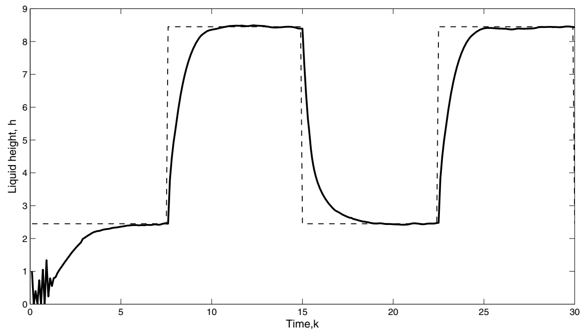
Fig. 6 PI controller performance with $K_p = 0.01$ and $K_i = 0.3$.

5 Results

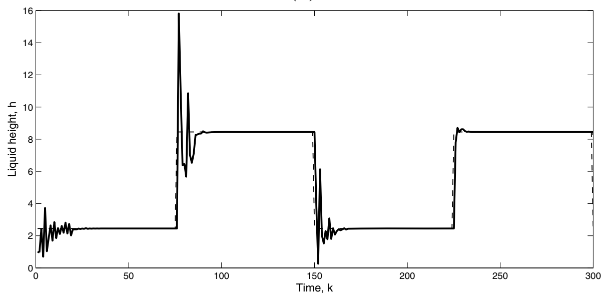
In order to test the operation of the planning strategy, the complete system is simulated during 30 seconds assuming a pre-determined signal reference $r(k)$. Figure 7 shows the performance of the proposed approach applied to the plant with different values for ω_1 and ω_2. It is easy to identify different responses depending upon the chosen value of ω_1 and ω_2.



(a)



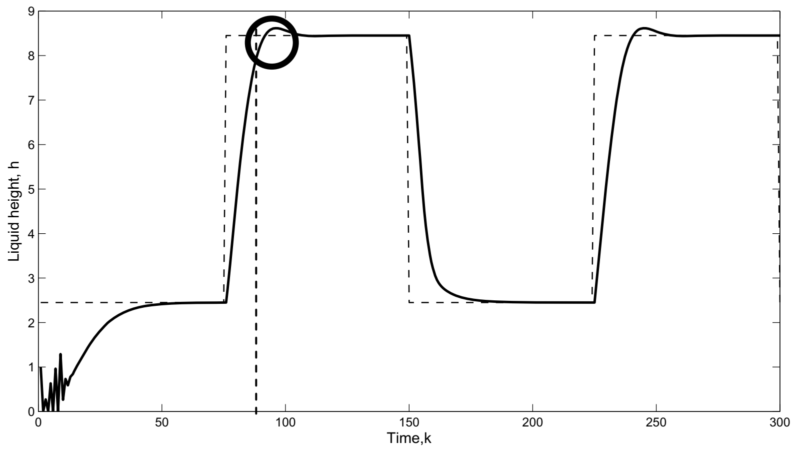
(b)



(c)

Fig. 7 Performance of the planning strategy applied to the plant considering different values for ω_1 and ω_2. (a) Setting ω_1 =1 and ω_2 =1, (b) setting ω_1 =0.8 and ω_2 =0.8 and (c) setting ω_1 =5 and ω_2 =1.

Figure 7a presents results from setting $\omega_1 = 1$ and $\omega_2 = 1$. A slower rise-time can be seen in contrast to Figure 6 which uses the PI controller. The system manages to tune the planning strategy by adjusting ω_1 and ω_2 so that there is a small overshoot, still showing a reasonable rise-time. Figure 7b is obtained by setting $\omega_1 = 0.8$ and $\omega_2 = 0.8$ while Figure 7c resulted after setting $\omega_1 = 5$ and $\omega_2 = 1$.



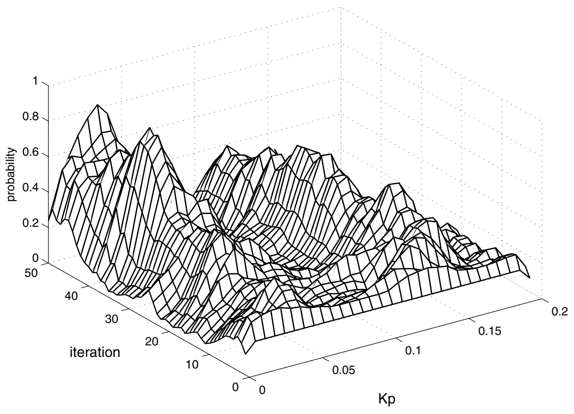


Fig. 8 Evolution of the probability-density functions in k=93 for K_p and K_i considering $\omega_1 = 1, \omega_2 = 1$.

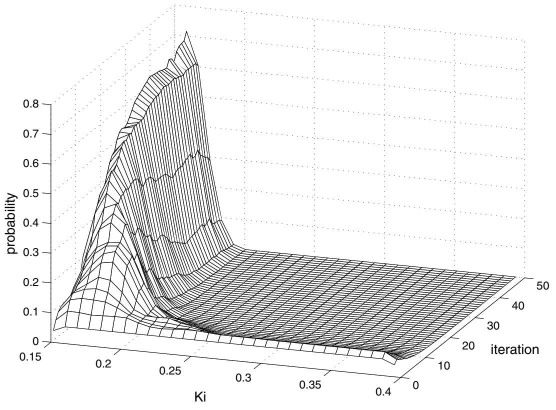


Fig. 8 (*continued*)

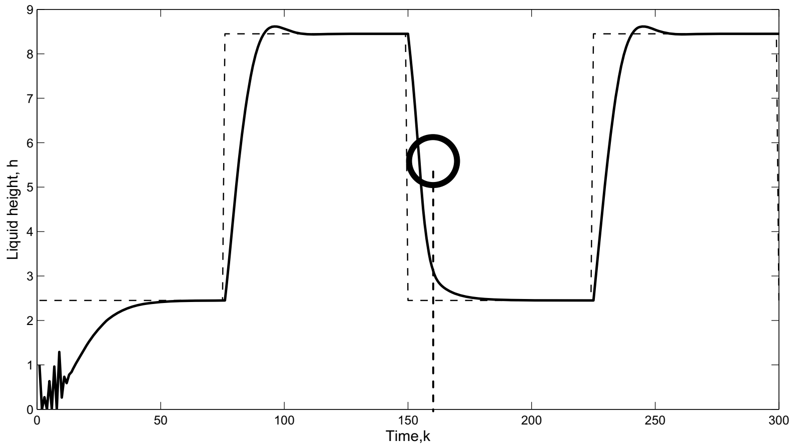
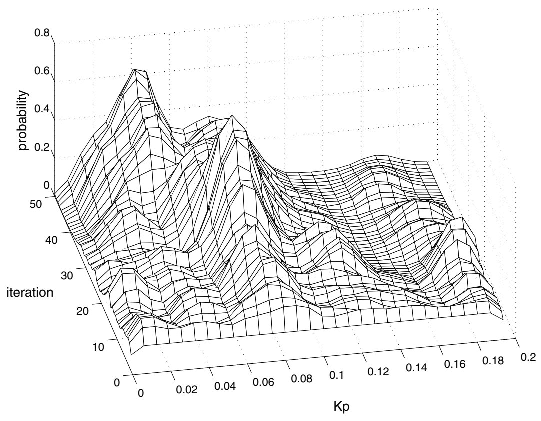


Fig. 9 Evolution of the probability-density functions obtained for K_p and K_i in k=156 considering ω_1 =1 and ω_2 =1.



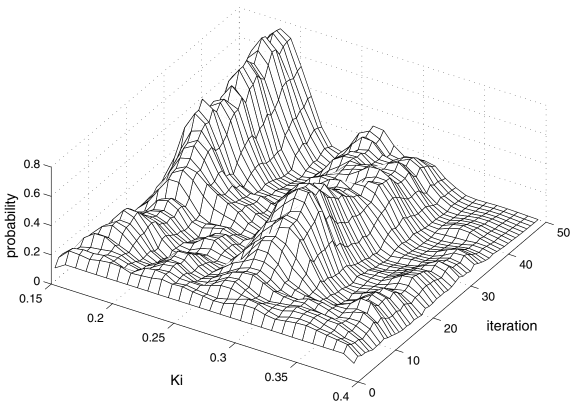


Fig. 9 (*continued*)

Two CARLA automata are employed for each parameter K_p and K_i respectively, both initialised by a uniform distribution. The election of each plan (values of K_p and K_i) is made at each sampling time according to the CARLA algorithm (see sub-section 4.3). The evolutions of the two probability-density functions are shown in Figure 8 and 9 considering two different sampling instants. Figure 8 shows the evolution of values K_p and K_i for k=93 while Figure 9 for k=156. It is straightforward to identify how the probabilities converge to a maximum through the iterations. The highest probability values K_p and K_i are used as parameters for the controller, just as it is provided by Equation 19.

In order to test the algorithm's performance, it is compared to the solutions provided by the Levenberg-Marquardt method (Kelley, 2000) and Genetic Algorithms (Chen et al., 2009). The former has been regarded as the most common in planning strategy with control applications, showing an interesting trade-off between precision and speed. On the other hand GA is the most well-known stochastic optimization methodology.

In particular, the Levenberg-Marquardt gradient-based algorithm (LM) is implemented according to Press et al., (1992). The method minimizes Equation (8) and updates all parameters following the equation:

$$\theta(n+1)=\theta(n)-(\nabla\varepsilon(\theta(n)\cdot\nabla\varepsilon(\theta(n))^{\mathrm{T}}+\Lambda(n))^{-1}\nabla\varepsilon(\theta(n)\varepsilon(\theta(n)) \tag{20}$$

where θ represents K_p and K_i, which are to be found, being $\varepsilon(\theta(n))=\omega_1\cdot(r(n+j)-y_m(n))+\omega_2\cdot u(n)$ $\nabla\varepsilon(\theta(n))$ the Jacobian and $\Lambda(n)$ the Cholesky factorization term $\Lambda(n)=\lambda\mathbf{I}$ with $\lambda>0$. As for the LA algorithm, the computation of K_p and K_i according to (20) is also employed by the control strategy.

On the other hand, the GA algorithm described in (Chen et al., 2009) takes the following values: population size=100, crossover probability = 0.55, mutation probability = 0.10, and the number of elite individuals = 2. The roulette-wheel-like selection algorithm and the 1-point crossover method are considered. The parametric setup is taken from the best set according to (Chen et al., 2009) which has considered lots of hand tuning experiments.

The values ω_1 and ω_2 in Eq. (8) are chosen in the simulation as $\omega_1=\omega_2=0.8$. Figures 10 and Figure 11 show the controller's performance using the Levenberg-Marquardt and the Genetic Algorithm procedure for the optimization.

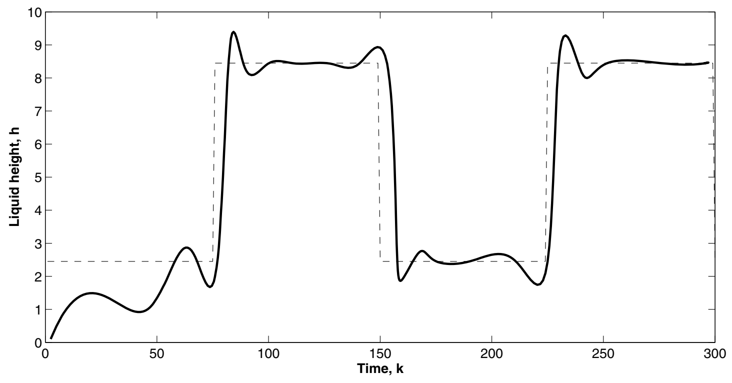


Fig. 10 Performance of the planning strategy using the Levenberg-Marquardt (LM) method as optimization algorithm

The results are averaged over 50 runs and summarized in Table 1. The values correspond to the worst case after simulation. Two different conditions are considered: first the optimization algorithm running 50 cycles and second reaching 120 cycles. The results show that for the CARLA method, the settling time converges about 42% faster than other methods at 120 cycles showing a minimal overshoot. On the other hand, the LM algorithm seems to surpass the GA at 50 cycles. However, just the opposite performance is obtained at 120 cycles.

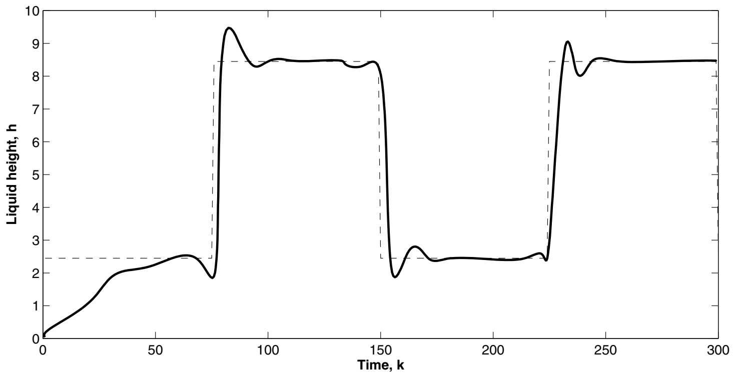


Fig. 11 Performance of the planning strategy using the Genetic Algorithm (GA) method as optimization algorithm

Table 1 Results obtained by the Leverberg-Marquad (LM) method, the Genetic Algorithm (GA) and the Learning Automata (CARLA) approach.

Method	50 cycles Average value ± Standard deviation		120 cycles Average value ± Standard deviation	
	Settling time (s)	Percent overshoot (%)	Settling time (s)	Percent overshoot (%)
LM	3.61±0.3	10.21±1.27	3.05±0.25	8.97±1.11
GA	4.33±0.41	14.42±2.33	2.11±0.12	4.16±0.79
CARLA	2.1±0.4	1.1±0.13	1.41±0.2	0.69±0.012

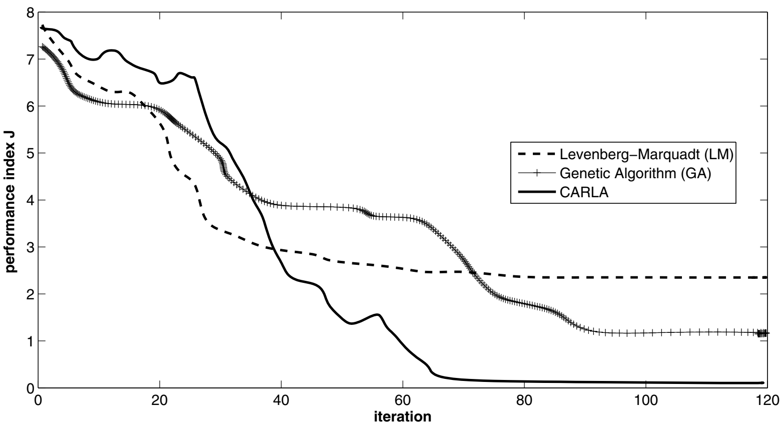


Fig. 12 Optimization evolution for LM, GA and CARLA algorithms.

Figure 12 presents the performance evolution for all methods (LM, GA and CARLA) with respect to the objective function *J*. It is evident the stochastic nature of the CARLA method as it tests several parameter values following a probabilistic approach until a minimum is reached. There is no deterministic relationship among the chosen values because they are generated while the probabilistic density function evolves (according to Equation 10). It is worth to notice that the performance index is decreasing gradually and one local minimum trapping has appeared for the LM method as a result of the parameter (n+1) being a modified version of the former (n). On the other hand, although GA does not reach an acceptable minimum after 50 cycles, it does manage at 120 cycles. The evolution of the performance index *J* is summarized in Table 2, with averaging over fifty runs. Again the performance is analyzed for two different iteration conditions: 50 and 120 cycles.

Table 2 Performance index *J* as it is generated by the Leverberg-Marquard (LM) algorithm, the Genetic Algorithm (GA) and the Learning Automata method (CARLA).

Method	50 cycles Average value ± Standard deviation Performance index *J*	120 cycles Average value ± Standard deviation Performance index *J*
LM	3.11±1.24	2.57±1.24
GA	4.14±1.51	1.19±1.012
CARLA	1.23±0.88	0.13±0.2

6 Conclusions

This chapter has discussed how to emulate the functionality of planning in order to exert control over non-linear plants. The procedure adopts the MPC methodology as planning strategy, following the CARLA algorithm as the optimization method. The system's performance is tested over a nonlinear plant. The results are compared to similar procedures built upon the Levenberg-Marquardt (LM) algorithm and Genetic Algorithms (GA).

In this chapter, the LA is applied to select optimal parameters K_p and K_i belonging to a PI control structure. The MPC and the optimization algorithm run over an uncertain plant's model. The CARLA algorithm has shown its abilities to probabilistically explore and reach optimal parameters.

The approach is also suitable for real-time applications. Although it requires learning in real-time, it can be effectively applied to nonlinear optimization problems with slow convergence under more conventional methods.

The proposed method also allows increasing the optimization speed in comparison to other algorithms such as LM and GA. The searching for optimal points is performed on the probability space rather than on the parameter space. Finally, the CARLA approach is faster to reach the minimum performance index ***J*** in comparison to the LM and GA methods.

Despite Table 1 and 2 indicate that the CARLA method can yield better results with respect to the LM and GA algorithms, it should be noticed that the chapter contribution is not intended to beat all the optimization methods which have been proposed earlier, but to show that the CARLA systems can effectively serve as an attractive alternative to traditional optimization methods for control purposes.

References

Beygi, H., Meybodi, M.R.: A new action-set learning automa-ton for function optimization. Int. J. Franklin Inst. 343, 27–47 (2006)

Beigy, H., Meybodi, M.R.: A learning automata-based algorithm for determination of the number of hidden units for three-layer neural networks. International Journal of Systems Science 40(1), 101–118 (2009)

Bloemen, H., Van den Boom, T., Verbruggen, H.: Optimization Algorithms for Bilinear Model–Based Predictive Control Problems. American Institute of Chemical Engineers 50, 1453–1461 (2004)

Camacho, E., Bordons, C.: Model Predictive Control (Advanced Textbooks in Control and Signal Processing). Springer, Berlin (2008)

Camacho, E.F., Bordons, C.: Nonlinear model predictive control: An introductory review. Assessment and future directions of nonlinear model predictive control. Springer, Heidelberg (2007)

Canale, M., Fagiano, L., Milanese, M.: Set Membership approximation theory for fast implementation of Model Predictive Control laws. Automatica 45, 45–54 (2009)

Chauvin, J., Corde, G., Petit, N., Rouchon, P.: Motion planning for experimental airpath control of a diesel homogeneous charge-compression ignition engine. Control Engineering Practice 16(9), 1081–1091 (2008)

Chen, W., Li, X., Chen, M.: Suboptimal Nonlinear Model Predictive Control Based on Genetic Algorithm. In: 2009 Third International Symposium on Intelligent Information Technology Application Workshop (2009)

Cueli, R., Bordons, C.: Iterative nonlinear model predictive control. Stability, robustness and applications. Control Engineering Practice 16, 1023–1034 (2008)

Fleming, P.J., Purshouse, R.C.: Evolutionary algorithms in con-trol systems engineering: a survey. Control Engineering Practice 10(2002), 1223–1241 (2002)

Gao, X.Z., Wang, X., Ovaska, S.J.: Fusion of clonal selection algorithm and differential evolution method in training cascade–correlation neural network. Neurocomputing 72, 2483–2490 (2009)

Garcia, C.E., Prett, D.M., Morari, M.: Model Predictive Control: Theory and Practice - A Survey. Automatica 25, 335 (1989)

Howell, M., Gordon, T.: Continuous action reinforcement learning automata and their application to adaptive digital filter design. Engineering Applications of Artificial Intelligence 14, 549–561 (2001)

Howell, M.N., Frost, G.P., Gordon, T.J., Wu, Q.H.: Continuous action reinforcement learning applied to vehicle suspension control. Mechatronics 7(3), 263–276 (1997)

Huang, L.: Velocity planning for a mobile robot to track a moving target—a potential field approach. Robotics and Autonomous Systems 57, 55–63 (2009)

Ikonen, E., Najimz, K.: Online optimization of replacement policies using learning automata. International Journal of Systems Science 39(3), 237–249 (2008)

Kashki, M., Lofty, Y., Abdel-Magid, Abido, M.A.: Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence. In: Huang, et al. (eds.) A Reinforcement Learning Automata Optimization Approach for Optimum Tuning of PID Controller in AVR System, pp. 684–692. Springer, Berlin (2008)

Kelley, C.T.: Iterative Methods for Optimization. SIAM Fron-tiers in Applied Mathematics (18) (2000); ISBN 0-89871-433-8

Liu, G.P.: Nonlinear Identification and Control: A Neural Net-work Approach. Springer, Berlin (2001)

Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O.M.: Constrained Model Predictive Control: Stability and Optimality. Automatica 36(6), 789 (2000)

Meybodi, M.R., Beigy, H.: A note on learning automata-based schemes for adaptation of BP parameters. Neurocomputing 48, 957–974 (2002)

Nagya, Z., Agachia, S., Allgowerb, F., Findeisenb, R., Diehlc, M., Bockc, H.G., Schloderc, J.P.: Using genetic algorithm in robust nonlinear model predictive control. Computer Aided Chemical Engineering 9, 711–716 (2001)

Nagy, Z.K., Mahn, B., Franke, R., Allgower, F.: Evaluation study of an efficient output feedback nonlinear model predictive control for temperature tracking in an industrial batch reactor. Control Engineering Practice 15(7), 839–850 (2007)

Najim, K., Poznyak, A.S.: Learning Automata - Theory and Applications. Pergamon Press, Oxford (1994)

Narendra, K.S., Thathachar, M.A.L.: Learning Automata: an Introduction. Prentice-Hall, London (1989)

Park, H., Amari, S., Fukumizu, K.: Adaptive natural gradient learning algorithms for various stochastic models. Neural Networks 13, 755–764 (2000)

Potočnik, B., Mušič, G., Škrjanc, I., Zupančič, B.: Model-based Predictive Control of Hybrid Systems: A Probabilistic Neural-network Approach to Real-time Control. J. Intell. Robot Syst. 51, 45–63 (2008)

Press, W., Flannery, B.: Numerical Recipes in C: The Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge (1992)

Seow, K., Sim, K.: Collaborative assignment using belief-desire-intention agent modeling and negotiation with speedup strategies. Information Sciences 178(2), 1110–1132 (2008)

Seyed-Hamid, Z.: Learning automata based classifier. Pattern Recognition Letters 29, 40–48 (2008)

Son, C.: Comparison of intelligent control planning algorithms for robots part micro-assembly task. Engineering Applications of Artificial Intelligence 19(1), 41–52 (2006)

Thathachar, M.A.L., Sastry, S.: Techniques for Online Stochastic Optimization. Springer, Heidelberg (2004)

Thathachar, M.A.L., Sastry, P.S.: Varieties of learning automata: An overview. IEEE Trans. Systems. Man Cybernet. Part B: Cybernet. 32, 711–722 (2002)

Torkestani, J.A., Meybodi, M.R.: An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata. Computer Networks 54, 826–843 (2010)

Tsetlin, M.L.: Automaton Theory and Modeling of Biological Systems. Academic Press, New York (1973)

Wu, Q.H.: Learning coordinated control of power systems using inter-connected learning automata. Int. J. Electr. Power Energy Syst. 17, 91–99 (1995)

Ying-Pin, C., Low, C., Shih-Yu, H.: Integrated feasible direction method and genetic algorithm for optimal planning of harmonic filters with uncertainty conditions. Expert Systems with Applications 36, 3946–3955 (2009)

Zeng, X., Zhou, J., Vasseur, C.: A strategy for controlling non-linear systems using a learning automaton. Automatica 36, 1517–1524 (2000)

Zeng, X., Liu, Z.: A learning automaton based algorithm for optimization of continuous complex function. Information Sciences 174, 165–175 (2005)

Optimization Strategies for Restricted Candidate Lists in Field Service Scheduling

Marko Žerdin, Alexander Gibrekhterman, Uzi Zahavi, and Dovi Yellin

Abstract. Field service scheduling (FSS) is a large class of practical optimization problems combining features of the vehicle routing problem (VRP), scheduling problems and the general assignment problem (GAP). In some cases the problem reduces to well known variants of VRP, while other, more common circumstances give rise to a distinct set of optimization problems that have so far received very little attention in the literature. In this chapter we show how strategies for restricted candidate lists (RCL) – methods for pre-calculating and contextualizing the candidate list reduction procedures within a context of a generic optimization framework, can be used to efficiently solve a wide spectrum of FSS instances in a real-life industrial environment. A comparison of results obtained using a greedy randomized adaptive search procedure (GRASP) meta-heuristic with and without the use of certain RCL strategies is presented as it applies to specific variants of the problem.

1 Introduction

Field Service Scheduling (FSS) is very common in the service industry [27], but has not yet been extensively explored in scientific literature, although some research on

Marko Žerdin
University of Southampton, UK
e-mail: marko.zerdin@zanyants.com

Alexander Gibrekhterman
ClickSoftware Ltd. Israel
e-mail: alex.gibrekhterman@clicksoftware.com

Uzi Zahavi
ClickSoftware Ltd. Israel
e-mail: uzi.zahavi@clicksoftware.com

Dovi Yellin
ClickSoftware Ltd. Israel
e-mail: dovi.yellin@clicksoftware.com

the subject has appeared in print [22, 30, 23]. In an earlier paper on FSS [6] we showed how meta-heuristics such as genetic algorithms (GA) [10] and ant colony optimization (ACO) [11, 15] can be applied to instances of this NP-hard combinatorial problem, yielding competitive results.

In this chapter we explore the potential benefits of using what we introduce in section 3 as strategies for RCL. We try to show how these strategies can accompany various meta-heuristics in order to improve their performance without being coupled with any specific optimization approach. We illustrate the effects of such strategies by using them to optimize different variants of FSS.

In section 2 we define the FSS class of problems and survey its characteristics. We enumerate some of its most commonly encountered variants, and compare it to other well known combinatorial optimization problems, most extensively to the VRP.

In section 3 we discuss a generic framework for solving FSS problems using various meta-heuristics. This framework was described in more detail in [6]. Here we focus on and expand upon the important role of restricted candidate lists in unifying the environment in which optimization algorithms operate and through this unified view enabling the use of various optimization strategies. We define the meaning of strategies in this context, and discuss their significance for making the generic framework produce high quality results in different business cases encountered in practice.

In section 4 we briefly review a GRASP meta-heuristic [12] that was used for obtaining results provided in this chapter. We then move on to describe specific FSS instances that served as a basis for experimental results in later sections.

In sections 5 and 6 we show how specific strategies become effective as the variant of the FSS that is being solved changes, and how some of their features affect the suitability of particular scheduling approaches. We demonstrate this on two strategies: a geographical constrained clustering approach using hard constraints where we pre-calculate resource work areas based on demand distribution, both geographically and taking their required skills and other constraints into account; and a bundling approach, in which demands at the same location are grouped together into bigger demands, thus eliminating the repeat visits to the same site or at least minimizing their number.

Our findings have an interesting parallel in a series of comprehensive studies by Beck et al. [2, 3, 5, 4]. They studied instances of Job-Shop Scheduling Problem (JSP) and instances of VRP, defined a mapping between them, and compared the performance of specialized algorithms for both kinds of problems on native and transformed instances. Their analysis tried to identify the features of the problems make them so distinctive that completely different optimization frameworks need to be used in order to obtain good quality solutions. In their conclusion, VRP algorithms outperform JSP algorithms as long as certain temporal and specialization constraints are relaxed. Even though the typical instances of JSP and VRP are clearly more distinct from each other than variants of FSS are amongst themselves, the central idea is that a change in some of the features of the problem makes its instances suddenly much harder to solve using the same solution framework. Strategies for restricted candidate lists, such as the ones presented in this chapter, are our attempt at counteracting this effect in order to make the same generic framework continue to produce high quality results across a wide spectrum of FSS problems.

2 The Field Service Scheduling Problem

FSS is a class of real-life optimization problems that is in its simplest distinctive form characterized by the following:

- A set of geographically distributed resources (technicians, installers, meter readers) with limited availability (working hours in their calendar) and a specific skill-set.
- A set of geographically distributed demands of specified duration and revenue, each requiring specific skills for its completion.
- The goal is to assign as many demands as possible to resources so that work and travel are completed subject to resource availability. Each assigned demand has to be assigned to a resource that has the skills it requires, each resource can only be at one place at a time, and the times between start of work, consecutively scheduled demands and end of work have to be sufficient to accommodate the time required to travel between their respective locations (see figure 1).
- The objective function consists of maximizing the value of the schedule, which in its simplest form equals the total revenue minus the travel cost.

Given the fact that resource availability is limited, some demands will typically remain unscheduled. This is trivially true when total demand duration exceeds total resource availability, but will also be true in many other realistic cases where travel time and resource differentiation assert their discriminating roles. Therefore, the challenge in solving FSS is twofold: to choose the subset of demands that will fit into the availability and yield the greatest revenue, and at the same time to minimize travel time either to simply minimize cost or, ideally, to fit more demands into the schedule instead of spending the time on travel.

The FSS problem just described contains sufficient elements to distinguish it from other optimization problems commonly encountered in the literature. However, if we relax appropriate constraints in the right way and adjust the objective function, specific variants of FSS become analogous with or come close[1] to variants of VRP, JSP and GAP.

FSS is a straighforward generalization of the traveling salesman problem (TSP). Any instance of TSP can be transformed into FSS following these steps:

1. The only resource is located at one of the TSP cities;
2. Each other city is represented by a demand of the same constant duration; and
3. The resource's working time interval is long enough to accommodate all demands regardless of their order in the schedule.

[1] By this metric two combinatorial optimization problems are considered to be close if not only can an instance of one be mapped into an instance of the other, but also such a mapping is natural to the problem in a sense that it preserves the suitability of algorithms that would typically be used to solve it.

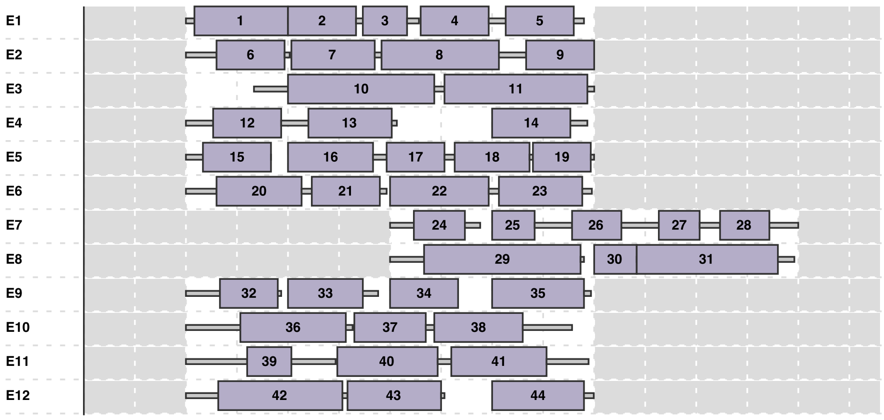


Fig. 1 A Gantt chart of a schedule for an example FSS instance with 12 resources and 45 demands. Only 44 demands are scheduled in this solution. Resource availability times are represented by white, while gray background represents non-working hours. Large rectangles represent scheduled demands, while narrow rectangles represent travel times. White spaces between some demands are waiting times, as this particular instance of FSS defines fixed two-hour appointment slots for each demand (see section 2.1.1).

This transformation is clearly linear in the number of cities in the TSP instance. It is also obvious that the solution to such an FSS instance would give the solution to the TSP instance that was transformed into it due to one-to-one relationship between TSP cities and FSS demands. Since TSP is NP-complete [17] it follows that FSS is NP-hard.

2.1 *Variants of the Field Service Scheduling Problem*

All FSS problems share the features described in the previous section, and some practical instances perfectly fit that description and require no more. However, the whole FSS class of problems is very rich, and we commonly encounter instances that introduce additional constraints. In this section we'll try to systematically present the most common variants of FSS encountered in practice.

2.1.1 Field Service Scheduling with Time Windows (FSS-TW)

The most commonly encountered additional constraint are demand time windows. Time windows define a time interval during which the demand needs to either be started or completed. They are specified in almost every realistic scenario, and can come from a number of possible sources: appointment booking, service level agreements (SLA), regulations, maintenance planning etc. We'll call the instances of this problem *Field Service Scheduling with Time Windows* (FSS-TW). In keeping with the common definition of VRPTW [26, 25] and unless explicitly stated otherwise, we'll henceforth assume that the time windows define the time interval during which the work on each demand has to start.

2.1.2 Field Service Scheduling with Dependencies (FSS-D)

Demands are not necessarily independent of each other. Very often they have relationships, such as the constraint to use the same resource or that one demand should start a specific amount of time after another one has finished. The relationships can be deemed critical, which triggers an all-or-nothing situation where the dependent demand shouldn't get scheduled unless its dependency is scheduled as well. Let us look at a couple of examples:

- Plastering has to be finished before painting, and the wall has to rest for a certain amount of time for the plaster to dry. This is an example of a critical relationship, as there is no point in scheduling the painting job unless plastering has also been scheduled to complete a sufficient amount of time before.
- Specialized tool pick-up in the morning for a job requiring the tool later in the day would also be an example of a critical relationship.
- An example of a non-critical relationship would be a courtesy call confirming an already agreed arrival for an important job that was scheduled and prepared for well in advance.

We'll call this problem *Field Service Scheduling with Dependencies* (FSS-D).

2.1.3 Field Service Scheduling with Long Demands (FSS-L)

Demands often have durations that approach or surpass the duration of any uninterrupted working time interval, yet they still need to be assigned in one piece to a single resource. Different cases cover anything from longer jobs, such as wiring a building that can take several days yet is best done by no more than one or two people, to outstanding paperwork that doesn't have very specific time or travel requirements and can be completed in breaks while waiting for other demands. In practice, such demands have to be seamlessly and dynamically broken into parts and scheduled spanning several working days, taking into account other existing work (when that is allowed) and associated travel, as well as travel from and to home at the start and end of each day. We'll call this variant *Field Service Scheduling with Long Demands* (FSS-L).

2.1.4 Field Service Scheduling with Efficiency (FSS-E)

Finally, resources are usually more or less experienced at working on specific types of demands, and therefore the same demand might take substantially different amounts of time to complete depending on who it is assigned to. This can be modelled as resource efficiency, overall or per skill. Also, travel time might depend on the mode of transport the resource uses (walking, bicycle, public transport, van, car) and the type of terrain (rural or urban area, pedestrian zone, central areas with frequent traffic jams). In such instances, travel times and route optimality become individualized and straightforward implementations of certain population based meta-heuristics become very hard or impossible to use. After all, in the extreme case where a sequence of demands has an unpredictably and inconsistently different objective value for each resource, there is no room for cross-pollination, and the population dissolves into a set of individuals each of which needs to be considered separately. We'll call this variant *Field Service Scheduling with Efficiency* (FSS-E).

2.1.5 Additional Objectives

Most variants of FSS encountered in practice can be covered by the constraints introduced so far. However, in addition to the constraints used, substantial differences in objective functions are also encountered. Let's take a look at a few examples:

- Instances of FSS may include objectives such as scheduling demands to their preferred resources, a preference to schedule demands to the least qualified available resource, a preference to schedule two or more related demands to the same resource, or a preference to keep resources in areas they know well.
- In instances of FSS-TW we often encounter an objective to minimize the risk of lateness by scheduling higher priority demands earlier in their time window, or, less frequently, the opposite objective of scheduling demands in the last part of their SLA interval in order to encourage customers to purchase a higher priority (and more expensive) support package with shorter SLA.
- When several demands may be situated at the same location, it is often very important to minimize the number of repeat visits even when that would generate a schedule with less overall travel because of the overhead associated with each visit or the impression such repeat visits might make on the customer. The objective function would in such cases typically contain a term that would reward consecutive scheduling of same-site demands or, alternatively, penalize repeat site visits.

2.1.6 Dynamic FSS

In addition to the static version of FSS with the full visibility of the data in advance, it is also very common to encounter a dynamic (or real-time) version of the problem, in which new demands (usually in the form of customer calls) or additional operational information (delays in travel or execution of demands, sick leave, missing parts etc.) can enter the system at any time. In this case the schedule needs to be continuously adjusted to accommodate the new information while remaining well optimized. We won't consider these variants of the problem in this chapter.

2.2 Field Service Scheduling and the Vehicle Routing Problem

The problem to which FSS would most commonly be associated and would seem closest to is VRP [26, 15, 25] or some variant thereof; and indeed, when simplified by removing certain constraints and objectives, FSS comes very close to VRP. However, in more typical cases FSS has several properties that sufficiently distinguish it from VRP that the methods commonly used to solve the latter need to be substantially modified in order to be used for the former, or they are not applicable at all. These distinguishing properties are:

- Resources in FSS are substantially (and not just quantitatively) different from each other while vehicles in VRP are typically all the same, or differ only in capacity. In FSS, there is a list of demands that each resource can perform, and there is no general relationship between those lists – they can be completely distinct, partially overlapping or some are subsets of others.
- Resources in FSS have limited availability in time, and are limited in number. In general VRP, it is usually feasible to allocate all work to a single vehicle, while an equivalent is not generally a feasible solution of FSS. Even in Capacitated VRP (CVRP, see [8]), there is usually an assumption that there are always enough vehicles available to accommodate the work. In FSS, the resources are individualistic, their availability is limited, and solutions don't necessarily contain assignments for all demands, making the selection of the best demands to schedule an important algorithmic consideration.

  This difference between VRP and FSS is similar to the difference between minimization and maximization versions of GAP respectively [9]. In the minimization version, the goal is to find the least costly allocation of jobs to machines assuming that they can all be completed, equivalent to VRP's unlimited fleet. In the maximization version, the goal is to find the most profitable allocation of jobs to a fixed set of machines with limited time given individual machine revenues, which is equivalent to FSS's individualized resources with limited availability. Additional discussion about FSS and GAP can be found in section 2.4.
- FSS has a different objective function from VRP. In VRP, the goal is to minimize either travel time, number of vehicles, or some combination of both. In FSS, as a consequence of typically not being able to schedule everything, the goal is to maximize the number of scheduled demands (or revenue, if each demand is assigned its own individual revenue when scheduled) while minimizing travel time. In general, additional revenue and cost components can be added to the objective function.

Other obvious differences between FSS and VRP, such as non-zero duration of demands, time limitation on each individual route or the typically large number of starting points (not only multiple, but many depots) have occasionally been covered in literature in various VRP variants [26].

In extreme and mostly impractical cases, FSS instances can also be instances of VRP. FSS instances where resource capacity is guaranteed to be sufficient to accommodate all demands constitute instances of what is essentially a time-constrained

VRP with vehicle specialization (transformed in a way that VRP travel time includes demand duration). Further on, in cases where any resource can be assigned to any demand, for example in meter reading, we have a time-constrained VRP without even a specialization requirement. However, in practice it is almost never the case that the capacity is guaranteed to be sufficient to assign all demands, and that is where the two problems and the methods for solving them diverge quite substantially.

2.3 Field Service Scheduling and the Job Shop Scheduling Problem

If we are guaranteed enough capacity and disjunctive resource skill-sets, all variants of FSS can be modeled as instances of JSP (or open-shop scheduling for FSS without dependencies) with parallel machines and sequence-dependent setup times [7]. The strength of this qualification and the specialized nature of the JSP problem which is rich enough to model FSS indicates that the two problems are not very strongly related. Still, when faced with FSS-D with complex demand dependencies and relatively little travel time, scheduling algorithms can be hybridized into the FSS solver and can be very helpful in guiding the decisions on the order in which demands should be considered for scheduling.

2.4 Field Service Scheduling and the (Maximum) General Assignment Problem

Max-GAP [14, 19] can be defined as follows: Given a set of machines with time constraint and a set of jobs with possibly different durations and revenues on different machines, find the assignment of jobs to the machines that produces the greatest revenue. There are many parallels between Max-GAP and FSS:

- FSS demands correspond to GAP jobs. FSS resources correspond to GAP machines with resource availability being equivalent to machines' time constraint.
- If a resource doesn't have the skills to work on a demand, the corresponding job's duration on the corresponding machine can be set greater than the machine's available time.
- Resource efficiency in FSS-E can be modeled by different durations of the same job on different machines.
- In both Max-GAP and FSS there is no guarantee that a full assignment will be found.

The most important component of FSS missing in Max-GAP is the concept of job ordering. In other words, everything in FSS that has to do with travel time, demand time windows and precedence constraints cannot be expressed within Max-GAP. Apart from that the two problems converge. In the real-life world of FSS the demand ordering is almost always of crucial importance, and the techniques derived from GAP are of limited practical use (apart from special cases like for example in section 6, where GAP methods are used not for solving the problem itself, but for separately addressing one of its aspects).

3 Strategies for Restricted Candidate Lists within a Generic Framework

Our experience lies in routinely solving the FSS problem for customers from different industries, with widely varied business workflows, constraints and objectives. When faced with such a variety of problems, a fundamental decision needs to be made about how to approach this task:

- One way is to design a separate solution for each individual business case, taking all the problem specifics into account and possibly developing a different implementation for each FSS variant as a result. This may result in a somewhat better solution quality, but it also leads to repeated effort, much longer implementation times, higher cost and lower adaptability to inevitable changes.
- An alternative to this is to set up a generic framework which has sufficient expressiveness to accommodate the variations encountered in the field, and use customization points and optimization workflow modifications to adjust the system to the particular problem at hand. This solution is somewhat less tailored to each particular problem, but it does have a lot more room for adaptability to different problems and a vastly greater ability to accommodate change, both in the model and in the workflow.

In this chapter we show how the second option can be used in conjunction with what we call strategies for RCLs in order to implement a high-performing optimization system for a wide range of FSS problems with greater potential for code reuse.

At the core of the proposed generic framework (which is described in detail in [6]) is a constraint propagation engine that models the FSS in three dimensions: resources (R), demands (D) and time (T). Constraints are implemented as independent components operating on a subset of the three dimensions. R-constraints can for example remove resources that are unavailable for scheduling, DR-constraints might eliminate resource-demand combinations that fail to satisfy the required skills, and DT-constraints might for example remove from consideration the times beyond the end of demand's appointment time.

The objective function is similarly composed of components, called objectives, which define contributions to the complete objective function. Such components for example define revenue for different demands, cost of travel or resource overtime, penalty for lateness or for increased risk of lateness etc. Objectives operate on both heuristic (during schedule building) and evaluative (after the schedule has been built) level, often with different weights and varying parameters.

This constraint propagation engine is generic enough to model the full richness of the FSS encountered in industrial practice, including all variants mentioned in the previous section, and rarely needs to be extended. It produces the candidate lists at every stage of the schedule building process, and offers the evaluation of the current and proposed schedules at any time.

On top of this constraint propagation engine, we propose a selection of meta-heuristic solvers, allowing us to choose the ones that best fit the problem at hand. These solvers may be much less generic than the framework itself, and can make specific assumptions about the instances that they are presented with in order to speed up the convergence; the only condition they need to satisfy is to be able to take advantage of the candidate lists and evaluations provided by the constraint engine, allowing them to be used even for the cases that they weren't originally designed to solve. The solvers that we investigated vary from deterministic local search to GRASP, genetic algorithms and ant colony optimization, each one with its own advantages and disadvantages, but all of them capable of taking advantage of the underlying engine and therefore, successfully or less successfully, solving a large variety of FSS variants. In essence, such approach moves the deployment of the system from the realm of implementation of solvers, algorithms and other logic components with all the risks that carries, to configuration, allowing successful reuse of tested components in new business domains, sometimes entirely outside the scope for which they were originally intended.

In addition to expressiveness, another important practical consideration in industrial applications is performance. Optimization is a game of diminishing returns, and squeezing the last few percent from the solution is far less important than the ability to create solutions of reasonable quality fast enough that the operational needs are satisfied. This is where the generic approach meets a serious challenge, and the need to adapt the solver to the problem at hand starts to look like an excellent idea. On the other hand, giving up the expressiveness of the generic solution is often quickly penalised when new business requirements inevitably appear. Is there a way to keep the generic nature of the solution while still being able to tailor the solver to the particular problem at hand?

We propose a two-pronged approach. First, instead of using the full candidate lists at each step, we use RCLs during most of the optimization process. Of course, the use of RCLs may in some cases, despite due caution, eliminate from consideration parts of the search space that contain better solutions. However, given the size of these potential improvements and the practical constraints imposed on the optimization time, living with such a risk is a price well worth paying given the substantial gain in performance on the other side. The support for RCLs is built into the framework and customizable, allowing the kind and the amount of pruning to be adjusted to suit the problem at hand, both its FSS variant, its size and its required performance. Previously developed components can be reused and configured again, as well as new components written if the case is the first of its kind. Several meta-heuristic solvers such as GRASP [12, 20, 24], tabu search [21] etc. can then take advantage of this flexible infrastructure.

The behavior of RCLs is described using the standard convention based on three parameters: α, β (both with values between 0 and 1) and m (an integer). Parameters α and m determine the size of the RCL by reducing the number of candidates to $\max(\alpha S, m)$, where S is the size of the full candidates list composed of all constraint satisfying variables, α determines the desired proportion of the candidate list we wish to keep, and m determines the minimum number of candidates within the RCL. Both α and m are predetermined and remain constant throughout the optimization process. The third parameter, β, modifies the level of greediness during the optimization run, allowing the selection process to be more or less biased towards greedier options. β is used to determine the proportion of the RCL that is considered for the actual selection, and changes, randomly or otherwise, for each selection step.

For problems with hundreds of resources and thousands of demands over several weeks horizon, the performance improvement gained by this approach is substantial, but not always sufficient, particularly when the time available for optimization is severely limited. In such cases, setting a suitable strategy for further meaningful reduction of the RCL can improve the rate of convergence towards solutions that take the particular considerations of the given problem into account, even though such speed-up might mean a lower-quality result overall. On other occasions, we may have additional information or knowledge that will, if expressed as a reduction strategy on the RCL, be useful in helping the convergence and will in many cases not only reduce the search time, but also improve the quality of results. In this case, implementing a strategy is an efficient and low risk way of putting knowledge into the system without modifying the implementation of the search algorithm.

In addition to specific, very targeted cases where additional information is conveyed, RCL strategies are useful whenever we have a non-local consideration that might influence the scheduling outcome. They often change what is originally a non-local, expensive to evaluate scheduling criterion into something practical and usable that can be verified locally with a much lower performance cost. Let us give a few examples for better understanding:

- There may be an imbalance between skills required by demands and skills that the resources have – some skills may be required by a disproportionally high number of demands. Prioritizing the scheduling of such demands and discouraging the scheduling of resources with those skills can be very helpful to ensure that a high proportion of such demands don't consistently remain unscheduled because their potential resources were already scheduled to demands requiring more common skills. This consideration is obviously not local and the analysis of skills needs to be performed prior to the main optimization.
- When solving instances of FSS-D, it is helpful to evaluate some critical measures of risk for networks of related demands, so that they can be prioritized in scheduling before such scheduling becomes virtually or entirely impossible. Even when the networks become impossible to schedule, it is beneficial to know about that so they can be excluded from the consideration without unnecessary effort being expended on trying to schedule them.

In response to such situations, we propose a second step in our approach – *strategies* for RCL contextualisation and recalculation. Instead of simply calculating the RCL at each step based on the valid moves and their heuristic values given the schedule at the moment, we introduce two additional steps:

- We pre-calculate as much heuristic information as possible for use later in the process. The particular pre-processing might include neighborhood lists for each demand in order to speed up geographical calculations, the ratio of required and available skills in order to identify rare skills and give them appropriate weight, or running a scheduling algorithm to determine the critical parameters for networks of related demands. This calculation is done only once at the start of the optimization, and its results are reused later during the RCL calculation, hence the name strategy.
- We introduce the RCL contexts that are triggered by certain scheduling events or states, to which we respond by dramatically increasing the weight of a specific strategy in the RCL calculation. For example, the scheduling of a demand that is part of a critical network of demands triggers an RCL mode in which other members of the network (and other networks of similar criticality) will be virtually the only candidates in order to allow the network to be realised without the danger of other demands taking their place and preventing the networks from being fully scheduled (and therefore fail as a whole, potentially leaving holes in the schedule). The strength of such contextual boost is of course configurable and depends on the importance of a particular strategy for each particular problem.

In this chapter we describe in detail two such strategies and demonstrate their effectiveness on a large-scale real-life FSS instance, as well as show how their effectiveness is not universal and varies depending on the problem variant at hand.

4 Experimental Setup

This section is devoted to a short description of the experimental setup used in the rest of the chapter. This includes the implementation of a GRASP [12] solver for FSS that we used on top of our strategy-modified RCLs, and a description of the leading real-life problem instance which was used as a basis to generate other problem instances used in experiments described later in the chapter.

4.1 The GRASP Implementation

While we have successfully used various meta-heuristics in order to solve FSS [6], we chose to demonstrate the strategies for RCL in this chapter using our implementation of the GRASP meta-heuristic. The reason for this decision is GRASP's inherent and direct reliance on candidate lists, allowing a fairly straightforward implementation without needing to reproduce the framework we described in section 3. This would allow interested readers to quickly try and test the results of the proposed strategies. Our implementations of genetic algorithms and ant colony optimization generally perform better, but their implementations get very involved and are clearly out of the scope of this chapter.

Following the initialization phase during which candidate lists are populated, strategy initialization performed and the static parts of the fitness of each option pre-computed, our GRASP implementation uses the following two stages:

1. An initial solution is created using a step-by-step constructive algorithm in which selections from the RCL are randomized at each step. A step in this regard consists of a selection of the next demand to be fitted into the partial solution, followed by a selection from possible placements according to the RCL. Once the demand has been incorporated in the partial solution, the RCL is updated and objective function contributions are reevaluated for the remaining demands. Greedier moves (i.e. the ones with more positive immediate effect on the objective function) receive a greater weight during this process. This stage is called a randomized greedy stage.
2. Increasingly more adventurous local variation operators are used to explore the solutions's local neighborhood in the search space, resulting in ever increasing changes to the original solution. The main activity of this stage is composed of deconstruction and reconstruction of an increasing number of steps used to create the initial solution in the first place, with modifications to the decisions that were originally made. For each new solution, deterministic local search or any other relevant local improvement methods are employed in order to reach a local minimum. This stage is called an adaptive stage.

We monitor the effectiveness of this iterative process by maintaining a count of iterations that have gone by without any substantial improvement in schedule quality. After each iteration, we implement either of the following two options:

(a) If a gain in schedule quality has been achieved in one of the recent iterations, randomly remove a subset of pre-defined size from the existing schedule in order to allow the next iteration to continue from there.
(b) If the limit on the number of successive stale iterations has been reached, destroy the entire schedule, allowing the next constructive stage to start afresh.

A full GRASP run consists of multiple iterations of these two stages, in which the best solution found throughout the process is kept as a result of the optimization.

Each stage has its own control loop, allowing for seamless strategy adjustments to the RCL. The strategies are initially selected according to the kind of problem at hand, and can then be further fine-tuned through experimentation. Substantial experimentation we conducted for this chapter using the described implementation showed that careful selection of RCL strategies can result in substantial adaptation of this generic setup to the FSS variant at hand, resulting in a significantly more effective search and faster convergence rates.

4.2 The Leading FSS Problem Instance

In this section we describe the principal FSS instance used in this chapter. The instance in question is based on an actual, real-life instance from the field of telecommunications.

The instance contains 80 resources and 1,700 demands within a single geographical district. This number of demands would typically schedule over two to three days as resources have the capacity to serve around 1,400 demands in two days of work. However, for demonstration purposes of both the results and some of the difficulties encountered, we modified the instance by setting all demands to schedule on a single day.

The business problem features resources which start and finish their workday from their home locations scattered throughout the geography. Resources have different work hours, skills and equipment. Demands differ by their time window, duration (which varies between 30 to 50 minutes), required skills and equipment.

The following constraints need to be satisfied by all valid solutions:

- Demands should be scheduled within their defined time windows and always for their exact duration.
- Demands should only be scheduled to resources having the required skills and equipment.
- The gaps between scheduled demands should reflect the travel time between them.
- Resources can only perform a single demand at any given time.
- Resources can only work within their regular daily work hours.

These constraints are a clear indication that we are dealing with an instance of FSS-TW. None of the additional elements defining FSS-L, FSS-D or FSS-E can be found in these constraints.

The following are the business objectives for the instance:

- Maximize number of scheduled demands - the value of each scheduled demand is constant at 50 points and doesn't vary with the demand's duration.
- Minimize travel time - the cost of each travelled hour is 50 points.
- Maximize the consecutive scheduling of demands situated at the same site – a bonus of 20 points is given for every such occurrence. This objective was used selectively in some of the experiments; see more about this in sections 5 and 6.

The average number of resource candidates per demand in this problem instance is 22 out of 80. This number is an indicator of *workforce specialization* – the less candidate resources there are for each demand, the more specialized the workforce is. We discuss some of the implications of different degrees of workforce specialization in later sections.

5 Constrained Clustering Strategy – When FSS Shifts towards VRP

The first strategy we present becomes relevant when the FSS problem is, oddly enough, simplified – when certain constraints limiting the number of candidate resources for each demand are either removed or play a lesser role. To give a more specific example, when most of the skills are sufficiently common amongst the resources, then most resources can perform most of the demands and skill constraints are effectively relaxed to the point where they don't have a real selective effect. This is where the story of this section begins.

5.1 The Business Case

In certain industries such as the domestic utilities sector, some of the work does not involve a high degree of workforce specialization. For example, in water and gas meter reading and repair, most of the resources are qualified to perform most demands. The ability to read or replace meters is the only relevant skill, thus allowing almost everyone who is trained in performing any work to work on almost anything.

An additional factor that commonly comes into play in such cases is a relatively short demand duration associated with this type of work. This implies that a resource can complete a dozen or more demands per day. When resource capacity is not tight, the optimization problem starts to resemble the classic VRP. Of course, the features that distinguish FSS from VRP, such as limited availability of a fixed number of resources and the inability to always schedule every single demand, are still present. But assuming resource levels are high enough to schedule all demands, the problem starts to resemble VRP very closely.

5.2 Implementing a "Cluster First, Route Second" Strategy

For large FSS instances and when their search space becomes more loosely constrained, we find that the usual optimization setup becomes less effective and the convergence to high quality solutions becomes slower. This introduces the need for a leading strategy to be applied to the RCL.

Clustering is the strategy that becomes effective in the VRP-like domain when DR constraints imply that most resources can perform most of the demands. Our implementation of clustering is not a straightforward one and involves a few methodologies. First, we employ the familiar "cluster first, route second" approach [13] and make use of its wide range of application in the field of VRP solutions. This approach has been comprehensively investigated and widely used in practice [26]. However, in addition to geographical clustering, in FSS it has to take the following considerations into account:

- Despite geographic proximity, certain demands can't be members of the same cluster. For example, if there is a single specialist in the domain that can perform certain demands, the demands that require this resource will all need to be in the cluster with the specialist regardless of their location.
- The balance of skills and capacity between resources and demands also needs to be considered when creating clusters. This is important as clustering represents an additional constraint on demands' scheduling, and such considerations

represent an attempt to at least try to match the demands' probability for getting scheduled when clustering.

Constrained clustering approach [29, 28] is a possible answer here, as it utilizes constraint programming in order to enhance clustering. In our case it incorporates a set of cannot-link and must-link constraints. The linking constraints define a relationship between pairs of demands implying the members cannot or must be associated with the same cluster.

The leading meta-heuristic for clustering is the well-known *k*-means clustering approach [18]. In the iterative clustering algorithm we aim to generate a predefined number (K) of geographically separated subsets of demands in which each demand belongs to a single cluster. The number of generated clusters is constrained to be smaller or equal to the number of resources for obvious reasons. The algorithm iterates through the following steps:

1. Create subsets of demands that minimize the total distance between demands within each of the different subsets while adhering to linking constraints described above.
2. Create a matching between clusters generated in the first step and the resources in the domain while attempting to maximize the predefined fitness function that evaluates the matching between clusters and resources. The fitness function is based on DR compatibility (i.e. the suitability of demands contained in a cluster given the resources assigned to serve this cluster, including the balance of requirements and availabilities in both time and skills), the cluster–resource distance and optionally additional components.
3. The results of the second step are analyzed, and the results of analysis are utilized during the creation and modification of the linking constraints. The constraints are created by identifying demands that are poorly fitted to their clusters (either by not having any scheduling options, requiring the skills that are locally deficient, or there is relative lack of availability in comparison to other clusters), and finding better fitting clusters for them according to fitness criteria similar to the ones used for resources.

The stop conditions of the algorithm are similar to those of the standard *k*-means approach. The algorithm is deemed to have converged when "jump" stabilization occurs and the fitness function no longer changes.

At this stage, the clustering results are used in order to adjust the RCL. We examined adjustments through either a constraint or an objective, and here we present results in the former case.

In the extreme case when K equals the number of resources, we achieve the maximum reduction of the candidate list. The effects of the clustering strategy are also most visible in this case, providing a clear gain in performance, but also introducing an "over-constraining" affect that may negatively influence schedule quality. This in turn invites further examination of the optimal number of resources per cluster which we explore in the next section.

5.3 Experimental Results

We start by comparing the results obtained using the GRASP meta-heuristic without any RCL strategies (referred to as "pure GRASP" later on), with those using clustering as a strategy with different values of K.

The results presented in figure 2 on the following page were obtained on the problem instance described in section 4. As we mentioned before, the average number of candidates per demand in the instance is 22 out of the total of 80.

Clustering was run with several values of K in order to observe the effect of the number of clusters on the quality of results as a function of optimization time. The results clearly indicate that the strategy is beneficial in the short term. We attribute this improvement to shorter candidate lists and therefore faster iterations. For example, with $K = 10$ the average number of candidate resources per demand is less than 8 as opposed to 22 in the unrestricted case. When more time is available for optimization, clustering eventually becomes a hindrance rather than an advantage. Its role in increasing performance becomes less important as search time increases, so its only remaining effective role as an artificial constraint (when compared to the unrestricted case) prevents the exploration of clearly advantageous options that pure GRASP is able to reach. The time the restrictive nature of clustering becomes obvious clearly correlates with the number of clusters K. In our experience, to achieve best results clustering should be gradually relaxed in the later stages of the optimization after its initial boost to faster convergence was consumed.

Next, we survey the impact of workforce specialization on the effectiveness of the clustering strategy. We regard specialization as equivalent to the proportion of resource candidates after applying DR constraints which disqualify part of the resources based on skills, equipment, time intervals and other relevant attributes.

In order to study the transition of a problem instance from less to more specialized, we have to consider an experimental setup very carefully. Using completely different problem instances with different levels of specialization wouldn't serve the purpose well as it would introduce a number of additional variables that would be very hard to control. Therefore, we took a smaller real-life instance with 275 demands and 20 resources[2] which we gradually modified from complete openness where all resources were valid candidates for all demands, to achieve ever higher specialization.

We started in a constellation without any skills, and all twenty resources were feasible candidates for all demands. Then, starting from an exact copy of the first instance, we first randomly selected a single resource per each demand that will not posses skills to perform it, leaving exactly 19 candidates per each demand. Moving on, we removed two candidate resources per demand from the original instance in order to get to 18 candidates per demand, and so on. While we ended up with the

[2] The demands in the instance were set with a priority value ranging between 1 (lowest) to 9 (highest). The value of calls with priority level of 1 was 100 points, and it was set to rise progressively by 50 points for every call of a higher priority, making priority 2 calls worth 150 points, priority 3 200 points and so on leading to the highest priority 9 calls having a value of 500 points. The cost of every travelled hour was set to 50 points.

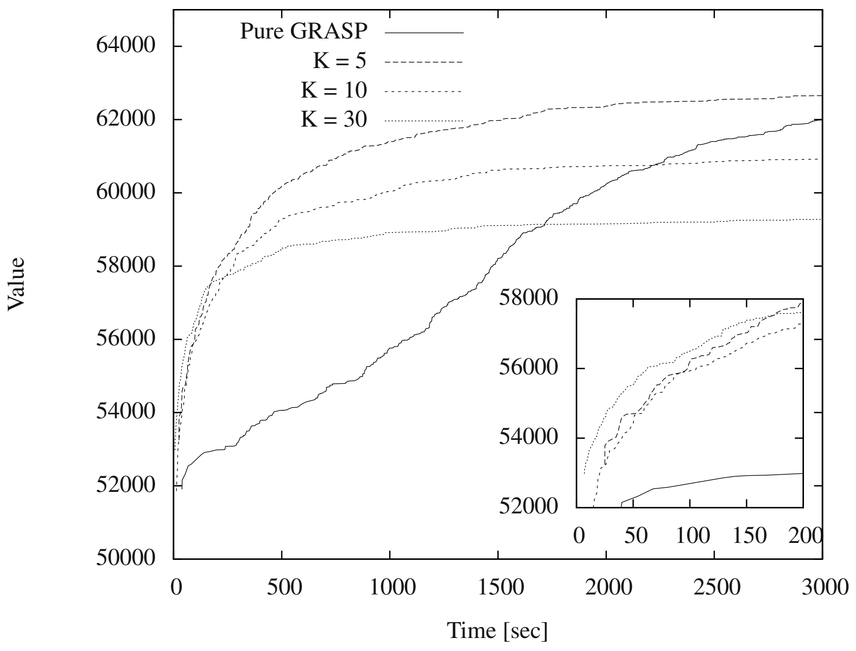
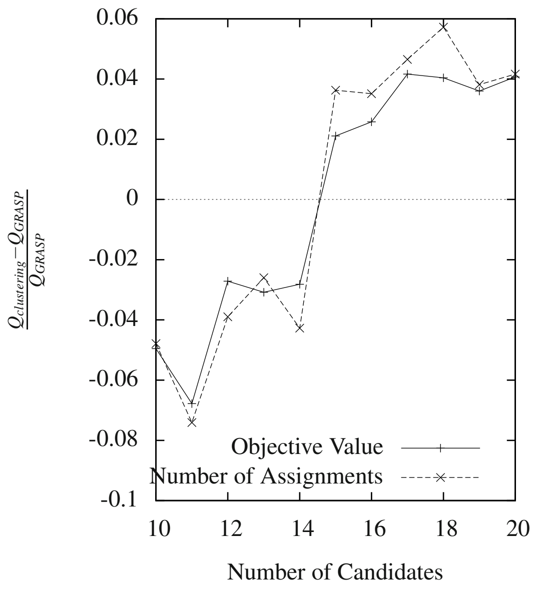


Fig. 2 The figure compares results obtained by pure GRASP with results of clustering at different measures of K, as function of processing time. The inner subfigure zooms-in on first 200 sec of the run. The comparison reveals the benefit of using tight clusters (K = 30) at very short time scales. Results of looser clustering (K = 10 and 5) emerge more slowly but outmatch the tighter clustering results at the medium processing times. As can be seen, at larger processing times pure GRASP will outperform results of any of the clustering strategy settings.

instances that were progressively more artificial, we feel that we successfully isolated resource specialization as the single most important difference between them. We also feel that always starting from the original instance instead of progressively modifying the already modified instances indicates that the results that we see are not the consequences of a particular set of lucky modifications, but have somewhat wider validity.

The results are shown in figure 3 on the next page. We measured schedule quality after a short constant period of optimization with and without clustering for a fixed value of K as a function of resource specialization. As we move from right to left in the graph and the specialization of resources increases, we find that the benefits of using clustering quite suddenly disappear, and the strategy becomes detrimental to the quality of the results. The specialization of resources basically divides the problem space into two regimes: one in which clustering can play a very instrumental role in achieving higher quality results at least in the initial stages of the optimization, and the other in which it shouldn't be used at all because it damages the quality even in the short term. The transition between the two regimes seems to be abrupt and could be described as a phase transition. As our past experiences in the field seem to indicate and further experiments of a similar kind show[3], the state transition is not just present in one instance, but seems to be a common occurrence in FSS. The exact place the phase transition is going to take place is not obvious, implying that care needs to be taken when using clustering in practice.



C	Clustering	No clustering
10	209	219
11	216	232
12	231	240
13	231	237
14	234	244
15	248	239
16	256	247
17	252	246
18	262	247
19	262	256
20	264	256

Fig. 3 On the left side, we see the relative value of using the clustering strategy compared with results achieved without it, as a function of workforce specialization. The units of the *x*-axis show the number of candidate resources per demand, while the *y*-axis represents the relative improvement in quality of clustering over pure GRASP, given by $(Q_{clustering} - Q_{GRASP})/Q_{GRASP}$, where the quality of results is measured as the value of the objective function and as number of created assignments. The result indicates a phase transition when the number of candidate resources falls below the level of 15 resources. The table on the right shows the actual number of scheduled demands with and without clustering as a function of the number of candidate resources per demand (C).

In Beck et al. [5], the authors make an analogous finding regarding the lack of affectiveness of VRP meta-heuristics when the level of fleet specialization crosses some borderline. They state: “When we increased the specialization of the fleet, we discovered that the routing technology failed to produce a solution”. With the closed and packaged algorithm used in their study, Beck et al. did not have the ability

[3] The results will be published separately in the future in a dedicated paper.

to investigate how exactly the quality of the results diminishes as specialization is increased. Luckily, we were in a position to perform such an analysis, as detailed above.

6 Bundling Strategy

In certain business cases, for instance in utility maintenance or meter reading scenarios, several demands are often situated at the same location, and can be performed consecutively by the same resource. Quite often this is in fact the optimal way of scheduling such demands – consecutively and to the same resource, although it's easy to find cases where scheduling two separate resources to visit the same site is actually better, as illustrated in figure 4.

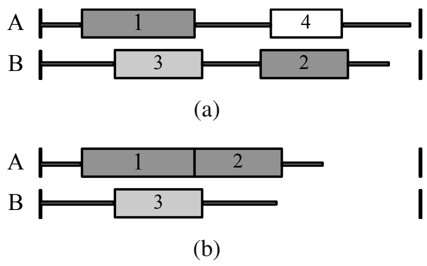


Fig. 4 Scheduling same-site demands consecutively and to the same resource is not always optimal. In the above example, resources A and B both have the skills to work on the same-site demands 1 and 2, but only resource A has the skills to work on demand 4 and only resource B has the skills to work on demand 3. Splitting the same-site demands, as in case (a), will result in more travel, but it will also allow all demands to be scheduled. Scheduling the same-site demands consecutively and to the same resource will always leave at least one demand off the schedule, as illustrated in case (b).

Visiting a site is often associated with overheads that are not immediately apparent, such as signing in when visiting the site, parking and unloading the van, and getting to the right location within the site. All these overheads are largely eliminated for all demands but one when same-site demands are scheduled consecutively and to the same resource, as illustrated in figure 5 on the next page. In such cases, splitting same-site demands between resources or scheduling a resource for repeat site visits is usually strongly discouraged even if the basic objective function actually increases on paper when doing so. The decrease in the actual demand duration can be utilized by dynamic optimization mechanisms to schedule more demands to the same resource once the actual duration of the whole same-site block becomes known. Of course, this is going somewhat beyond the pure FSS as defined here, but in practical terms this is a common situation and therefore of great interest and importance. Repeat site visits in practice also create a psychological impression of perceived sub-optimality by resources carrying out the work and dispatchers working with the schedule, which decreases the trust in the automated schedule optimization. For all these reasons, many organizations choose to place a great deal of weight on minimizing the number of site visits, even at the expense of other business considerations.

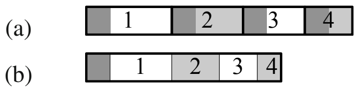


Fig. 5 Realistic demand duration will include not just the actual work that needs to be done, but also overheads such as entering the site or performing a site safety check. If same-site demands are scheduled separately, such as in case (a), each of them will contain these overheads (shown here as a dark shaded area at the start of the demand's duration). However, if they are scheduled consecutively and to the same resource, such overheads will be largely or completely eliminated in practice, as shown in case (b).

6.1 What Is Bundling?

Bundling is a strategy in which same-site demands with suitably overlapping time windows and requiring similar skills and parts are merged into a "bundle" – a single demand that is composed of qualifying same-site demands. Several considerations play important roles in bundling strategy:

Selection defines which demands get to be considered for bundling and which demands can be bundled together. This might be used to prevent bundling together demands that have no overlap in their time windows, create separate bundles from demands requiring different types of skills, or to avoid bundling high-priority or highly specialized demands together with common demands that anyone can work on.

Aggregation defines how the properties of the bundle are calculated from the properties of individual bundled demands. The bundle's required skills would typically be a union of required skills of all bundled demands[4], and the bundle's time window would typically be an intersection of bundled demands' time windows, but we encountered several other properties and aggregation methods in practice.

[4] There is a trade-off to be made between bundling together all same-site demands, or only demands requiring same or commonly associated skills. In the former case, we will generally have fewer bundles, but can also end up with skill-set requirements for some of the bundles that few or none of the resources could meet. The latter case would often result in less demanding bundles with more resources to choose from, but it can also result in more site visits. Our experience tends to favour the latter approach.

Scaling defines how the bundle's duration is derived from the durations of bundled demands. If we want to compare the solutions with and without bundling, such as we are trying to do in this chapter, then the bundle's duration has to be the sum of all bundled demands' durations. However, as illustrated in figure 5 on the preceding page, this is not always realistic. Therefore, the bundle's realistic duration in real-life applications would very often be quite significantly shorter than the sum of bundled demands' durations, and scaling reflects that expectation and tries to model it mathematically.

Partitioning defines how compatible same-site demands will be split into several bundles when there are too many of them to create a single bundle of reasonable duration. Having a single bundle that is longer than any resource's availability would quite obviously not be a good idea. Similarly, it would also not be a good idea to have a four-hour bundle that can only be scheduled after 2PM if all resources stop working at 5PM. Apart from such extreme cases, partitioning would typically be tuned to minimize the amount of artificial constraints[5] imposed on demands in the bundle as a consequence of property aggregation.

Using scaling and permissive aggregation (for instance, an abandonment of certain required skills for bundled demands, or an effective extension of their time intervals when mere presence on-site is sufficient and the bundle's demands don't necessarily need to be performed individually within their time interval) would typically result in a schedule that can not be unbundled and still considered to be a feasible solution of the original problem. Depending on the circumstances, the durations would be too short or some demands could not be performed within their own time window after unbundling.

On the other hand, it is straightforward to construct a bundling procedure that would preserve the feasibility of the solution: no scaling, bundle skills are a union of all bundled demands' skills, bundle time interval is an intersection of all bundled demands' time intervals decreased from the right by the bundle duration, and the bundle duration is no longer than the intersection of time intervals of all bundled demands. The last two points might seem non-obvious, but this is the only way to guarantee that all bundled demands will fit the time window when the bundle is trivially unbundled (unbundled without reordering or any kind of additional logical processing, as illustrated in 6 on the next page). We'll call such bundling *faithful bundling*. We'll only use faithful bundling in our results in order to ensure that the results with and without bundling are directly comparable.

The main effects of faithful bundling are that the total number of demands in the later optimization process is decreased, sometimes quite substantially, and that same-site demands that are bundled together are assured to be scheduled consecutively and to the same resource.

[5] When demands are bundled, the constraints imposed on the bundle are tight enough to ensure that each demand still satisfies its own constraints. If some demand's constraints are tighter, imposing such constraints on the bundle as a whole effectively decreases the scheduling options for other demands in the bundle. We refer to such constraints as artificial constraints or bundling constraints.

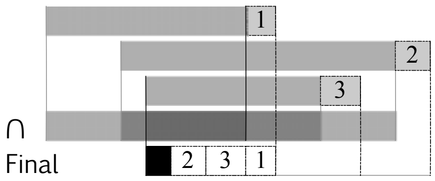


Fig. 6 The first three rows above show three demands' time intervals (darker gray) followed by their duration (the dotted box with a number in it). Since the FSS definition defines the demand's time interval as the time that the demand is allowed to start, the left and right side of this box indicate the latest start and latest finish respectively for each demand. The fourth row shows the intersections of these time intervals, using the intensity of colour to indicate the number of intersecting time intervals. Only the darkest part indicates the intersection of all three demands and therefore the times during which the bundle of all three demands can start. Finally, the fifth row takes into account the fact that the whole bundle needs to finish by the earliest latest finish (in this case the latest finish of demand 1) and that the order of work within the bundle can rarely be guaranteed in practice, further reducing the bundle's time interval to the black bit.

6.2 Bundling as Bin Packing

In case of the most basic FSS and faithful bundling we are really faced with an instance of the classical bin packing problem [16, 1]:

- Each bundle that we create is equivalent to filling a bin in the bin packing problem.
- We are trying to minimize the number of bundles created which is equivalent to minimizing the number of bins that we use.
- Durations of bundled demands are equivalent to item sizes in bin packing.
- The duration of each bundle is limited by an external parameter, as having demands that are too long can make it very hard to schedule them. This is equivalent to the limited bin size, which is constant in the classical bin packing problem. The ultimate limit here would of course be the length of the longest availability interval amongst resources minus the travel time required to get to the bundle and back from it. In practice the limit would be set lower than that.

In a more realistic case of FSS-TW, still with faithful bundling, the situation becomes substantially more complex. The maximum bundle size now changes as demands are added to it because it is limited by the size of the intersection of all bundled demands' time intervals. In addition to that, in order to determine whether a certain demand can be added to a bundle, it is not enough to look just at its duration, but we also have to consider its effect on the bundle's maximum duration once it's added to it. These two considerations are equivalent to bins getting smaller as we add certain items to them, and the items that are packed also being picky about the combinations in which they can be packed together.

In the process of bundling, each demand's scheduling options decrease in more ways than one. In FSS, the list of their candidate resources can be decreased by the skill requirements of their co-bundled demands. In FSS-TW, their time window can decrease quite dramatically because of the additional constraints imposed by time window aggregation in faithful bundling. The combination of decreased time window size and increased duration can also eliminate certain resources that would be valid candidates for each and every demand in the bundle. In light of this, the minimization of imposed constraints is an important consideration for any realistic bundling strategy. When viewed as a bin packing problem, this represents an important secondary objective in addition to decreasing the number of bundles.

6.3 Experimental Results

The bundling strategy serves as a pre-processor for optimization, essentially modifying the FSS instance by forcing bundles as hard constraints over the RCL. The practical implication is that the demands that have been bundled together will be seen as a single unified demand within the candidate list, and scheduled as a group.

The anticipated outcome of focusing on scheduling same-site demands consecutively and to the same resource is that the overall number of site visits will be minimized, and this becomes an important measure of schedule quality under these circumstances. The general FSS objectives such as maximization of the number of scheduled demands and the minimization of travel cost also remain important, but their relative weight is decreased to a certain degree.

It is also remarkably interesting to observe the effect of bundling on the number of demands that fit into a schedule as bundles become progressively larger, individually require more capacity and become harder to schedule, increasing the risk that all demands bundled within them will remain unscheduled.

Observing these two effects of bundling was the main focus of our research as presented here. The results were obtained on the problem instance introduced in section 4. We performed a series of short schedule optimization runs without bundling (equivalent to maximum bundle size of 1) and with bundling for various values of the maximum bundle size. Bundling was faithful in all cases, and was used as a hard constraint on RCL. After performing the initial measurement on this result, we unbundled all bundles that remained unscheduled, and performed an additional short optimization run trying to fit now individual demands into the existing schedule. The results are shown in figures 7 and 8.

Let us first look at the main declared focus of the bundling strategy – the reduction of the number of site visits. As we see in figure 7, the strategy is very effective on this front. The number of site visits decreases quite dramatically as soon as bundling is introduced, and continues to decrease as bundling becomes more ambitious by increasing the maximum bundle size. A similar, although somewhat less pronounced effect is seen when bundling is subsequently relaxed.

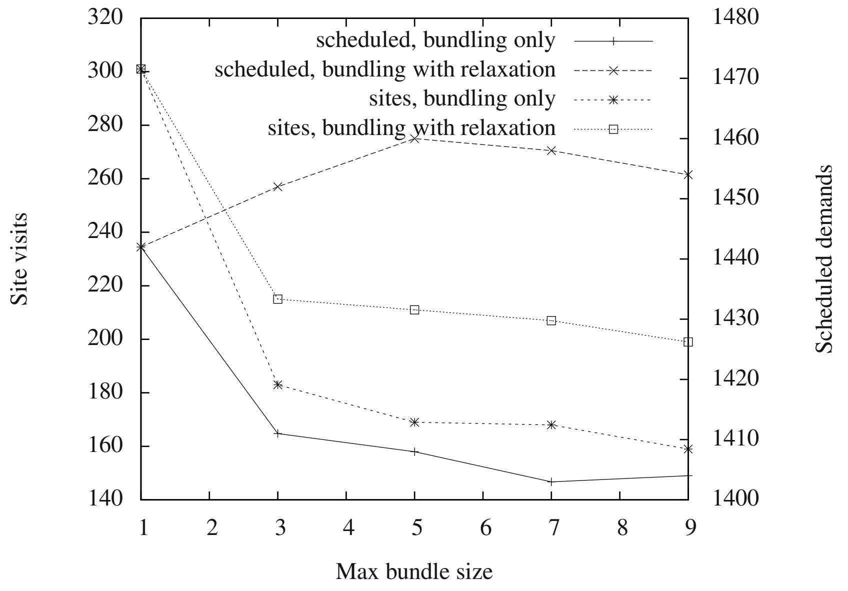


Fig. 7 The figure depicts how the number of site visits and the number of scheduled demands changes as we change the maximum size of the bundles created by the bundling strategy (1 means no bundling at all). We use the strategy as hard strategy ("bundling only") and relaxed for the last round of scheduling in order to allow the demands whose bundles didn't fit into the schedule to be scheduled individually ("bundling with relaxation").

The story of the number of scheduled demands is also interesting and indicates that hard bundling does have a cost. The number of scheduled demands[6] decreases as soon as we introduce bundling, indicating that longer resulting demands are significantly harder to schedule. The trend becomes less pronounced and loses significance as the maximum bundle size increases, and coincides with how well the longer bundles fit the schedule, which seems to become the predominant factor in the end. This can be explained as a consequence of the fact that there aren't that many sites in the instance that can actually take advantage of the longer maximum bundle size.

When we introduce relaxation after hard bundling, the number of scheduled demands consistently increases when compared with the result without bundling. The increase is greater for medium-sized bundles, indicating that such bundling might actually have an independent heuristic value which will need to be investigated further in the future. As bundle sizes become longer, the improvement over the case without bundling remains, but decreases somewhat. Not being able to fit the large bundles during the initial optimization results in those gaps being filled by

[6] All demands bring the same revenue in this instance, so the number of scheduled demands is proportional to the total revenue the solution brings.

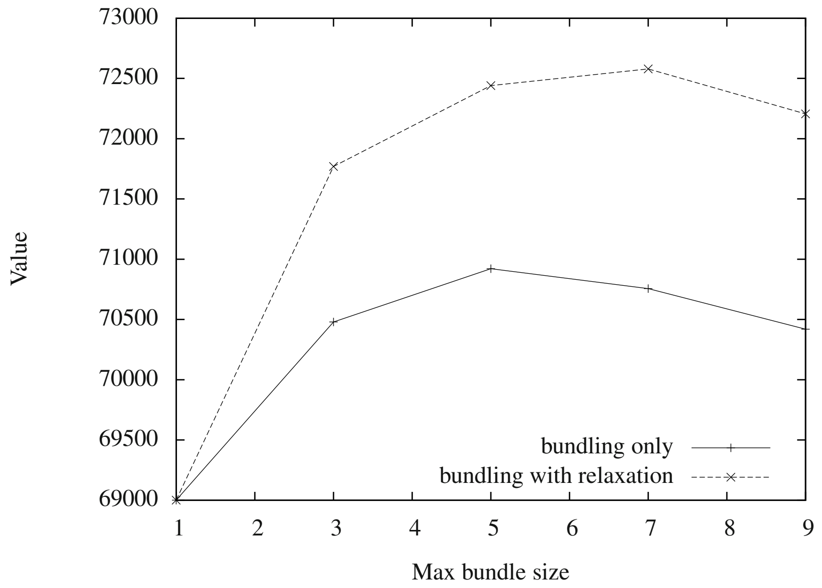


Fig. 8 The figure depicts how the objective value changes as we change the maximum size of the bundles created by the bundling strategy (1 means no bundling at all). The results with and without relaxation correspond to the results shown in figure 7 on the previous page. The increase in objective value for the "bundling only" case despite the reduction in number of scheduled demands can be explained by a high cost associated with return visits in the objective value calculation.

other demands of lesser geographical suitability, and the same-site demands can't be squeezed into the schedule after the unbundling to quite the same degree.

Figure 8 shows the objective function for the same set of experiments. The objective function includes a strong preference for scheduling same-site demands consecutively and to the same resource. The strategy is clearly effective in achieving that goal, and produces superior results for all values of maximum bundle size. However, the figure also indicates that there is a tipping point, and that the reduction in the number of site visits eventually becomes insufficient to compensate for the reduction in revenue due to the inability to schedule more same-site demands that are stuck in non-fitting bundles.

7 Conclusions

In the scope of this chapter, we defined and described the FSS problem and several of its variants. We then introduced the concept of RCL strategies as a part of a wider, unified meta-heuristic approach. We demonstrated the benefits of two such strategies on an actual FSS instance with real-life characteristics and using a GRASP implementation. The specific strategies, clustering and bundling, are both relevant to the challenges faced by the actual service organizations.

The presented results open room for considerable amount of future research. We plan to conduct further analysis of the attributes contributing to the phase transition identified in relation to the clustering strategy described in section 5, in particular when levels of workforce specialization are intensified. This will likely require generating a considerable number of different problem instances and deducing the shape of the phase transition though proper averaging over results. The ultimate goal here would be to achieve an approximation formula for the location of a phase transition based on the attributes of a given problem, and to better understand the factors contributing to its existence.

The main practical implication of this research would be the ability to make an automated decision whether to use the strategy or not based on the properties of the instance at hand, which would clearly be very beneficial in a commercial FSS system. This is even more true in scenarios involving real-time optimization where the need to quickly incorporate the new information into the existing solution is even more pressing.

Acknowledgements. The authors would like to stress that the contents of this chapter are not a description of the algorithms used by ClickSoftware Ltd., the commercial software vendor with which all authors are or have been associated, while gratefully acknowledging the help, support and contributions which our research received from ClickSoftware management and from our colleagues. In particular, we would like to thank our colleague Israel Beniaminy for his enriching advice and valuable criticism which contributed significantly to the maturity of this chapter.

References

1. Albers, S., Mitzenmacher, M.: Average-case analyses of first fit and random fit bin packing. Random Struct. Algorithms 16(3), 240–259 (2000)
2. Christopher Beck, J., Prosser, P., Selensky, E.: Graph transformations for the vehicle routing and job shop scheduling problems. In: Corradini, A., Ehrig, H., Kreowski, H.-J., Rozenberg, G. (eds.) ICGT 2002. LNCS, vol. 2505, Springer, Heidelberg (2002)
3. Beck, J.C., Prosser, P., Selensky, E.: On the Reformulation of Vehicle Routing Problems and Scheduling Problems. In: Koenig, S., Holte, R.C. (eds.) SARA 2002. LNCS (LNAI), vol. 2371, pp. 282–289. Springer, Heidelberg (2002)
4. Christopher Beck, J., Prosser, P., Selensky, E.: A case study of mutual routing-scheduling reformulation. J. Scheduling 9(5), 469–491 (2006)
5. Beck, J.C., Prosser, P., Selensky, E.: Vehicle routing and job shop scheduling: What's the difference? In: ICAPS, pp. 267–276 (2003)

6. Beniaminy, I., Yellin, D., Zahavi, U., Zerdin, M.: When the rubber meets the road: Bio-inspired field service scheduling in the real world. In: Pereira, F.B., Tavares, J. (eds.) Bio-inspired Algorithms for the Vehicle Routing Problem, pp. 191–213. Springer, Heidelberg (2009)
7. Bertsimas, D., Gamarnik, D.: Asymptotically optimal algorithms for job shop scheduling and packet routing. Journal of Algorithms, 296–318 (1999)
8. Borgulya, I.: An algorithm for the capacitated vehicle routing problem with route balancing. Central European Journal of Operations Research 16(4), 331–343 (2008)
9. Chekuri, C., Khanna, S.: A PTAS for the multiple knapsack problem. In: SODA 2000: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, pp. 213–222 (2000)
10. Davis, L.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
11. Dorigo, M., Stützle, T.: Ant Colony Optimization (Bradford Books). The MIT Press, Cambridge (2004)
12. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
13. Fisher, M.L., Jaikumar, R.: A generalized assignment heuristic for vehicle routing. Networks 11(2), 109–124 (1981)
14. Fleischer, L., Goemans, M.X., Mirrokni, V.S., Sviridenko, M.: Tight approximation algorithms for maximum general assignment problems. In: SODA 2006: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 611–620 (2006)
15. Gambardella, L.C., Taillard, E., Agazzi, G.: MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. Technical report, IDSIA, Lugano, Switzerland (1999)
16. Garey, M.R., Graham, R.L., Ullman, J.D.: An analysis of some packing algorithms. Combinatorial Algorithms, 39–47 (1973)
17. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
18. Hartigan, J.A.: Clustering Algorithms. John Wiley & Sons, Inc., New York (1975)
19. Nutov, Z., Beniaminy, I., Yuster, R.: A (1-1/e)-approximation algorithm for the generalized assignment problem. Oper. Res. Lett. 34(3), 283–288 (2006)
20. Resende, M.G.C.: Metaheuristic hybridization with GRASP. In: Chen, Z.-L., Raghavan, S. (eds.) Tutorials in Operations Research. Inst. for Mgmt Sci. and O.R. INFORMS (2008)
21. Santos, H.G., Ochi, L.S., Souza, M.J.F.: A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. J. Exp. Algorithmics 10, 2–9 (2005)
22. Spieksma, F.: On the approximabilty of an interval scheduling problem. Journal of Scheduling 2, 215–227 (1999)
23. Stein, R., Dhar, V.: Satisfying customers: Intelligently scheduling high volume service requests. AI Expert 12, 20–27 (1994)
24. El-Ghazali, T.: Metaheuristics: From Design to Implementation. Wiley Publishing, Chichester (2009)
25. Tan, K.C., Lee, L.H., Zhu, K.Q., Ou, K.: Heuristic methods for vehicle routing problem with time windows. Artificial Intelligence in Engineering 15 (3), 281–295 (2001)
26. Toth, P., Vigo, D. (eds.): The vehicle routing problem. Society for Industrial and Applied Mathematics (2001)

27. Vigoroso, M.W.: Field service optimization part 2: Synchronizing supply and demand in right time. Aberdeen Group Benchmark Report (2005)
28. Wagstaff, K., Basu, S., Davidson, I.: When is constrained clustering beneficial, and why? In: AAAI (2006)
29. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained k-means clustering with background knowledge. In: ICML, pp. 577–584 (2001)
30. Weigel, D., Cao, B.: Applying GIS and OR techniques to solve sears technician-dispatching and home delivery problems. Interfaces 29 (1), 113–130 (1999)

Framework for Integrating Optimization and Heuristic Models for Solving Planning and Scheduling Problem in a Resin Manufacturing Plant

Mohamed K. Omar and Yasothei Suppiah

Abstract. This chapter introduces a three-level hierarchical production planning and scheduling approach for a multi-product batch processing resin plant in South East Asia. The approach integrates optimization and heuristic models to determine aggregate plans, master production schedule, number of batches that need to be scheduled and finally sequencing of jobs with the objective to minimize the total weighted tardiness. At the top level of the hierarchy, our approach deploys a mixed-integer linear programming model to solve the aggregate plans where setups occur. At the second level a weighted integer goal programming model is developed to disaggregate the aggregate plans and provides an optimal number of monthly batches to be sequenced in the presence of setups activities. At the third level of the hierarchy, a job-sequencing model is developed that combines two heuristics approaches that aims to minimize the total weighted tardiness. Real industrial data is used to test and validate the proposed approach. The results indicate that the approach is capable of dealing with a full range of the products produced by the resin plant.

Keywords: hierarchical production planning, sequencing, batching, heuristics, weighted tardiness.

Mohamed K. Omar
Nottingham University Business School
Jalan Broga-43500 Semenyih, Selangor Darul Ehsan. Malaysia
Fax: +603-89248019, Tel: +603-89248697
email: mkhaledomar@gmail.com

Yasothei Suppiah
Faculty of Engineering and Technology, Multimedia University- Melaka Campus.
Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia
Fax : +606-231 6552, Tel : +606-252 3814
email: yasothei.suppiah@mmu.edu.my

1 Introduction

This chapter deals with a planning and scheduling problem in a multi-product single machine batch process environment. Planning in a production organization is about making decisions based on the current status of the organization and is aimed at achieving certain future goals. Production planning which deals with a longer time horizon, determines the expected inventory levels, as well as the workforce and other resources necessary to implement the production plans.

Scheduling on the other hand, deals with a shorter time horizons, coupling individual products with individual productive resources with the emphasis often being on the lower level decisions such as sequencing of operations. Decisions made at the planning level have a strong influence on scheduling which makes both of them closely related. In general, planning and scheduling refer to the strategies of allocating equipment and utility or manpower resources over time to execute processing tasks required to manufacture one or several products.

In the resin manufacturing environment, batch processing is dominant and obtaining optimal scheduling of the products in the batch processing machines is of great concern in productivity and on-time delivery management. Batch processing means the processing of an integrated, non-divisible units of production such as volume of resins, polymer or plastic. Batch operations are used to provide a wide variety of products and it is prevalent across a wide spectrum of process industry. In such an industry, products are often grouped into incompatible product families, where an intensive setup is incurred, whenever production changes from one product family to another.

Typically, in the resin production environment, the planning and scheduling task starts by considering a set of orders where each order specifies the product and the amount to be manufactured as well as the promised due date. The most important task of the planner is the so-called batching of orders, (see Mendez et al. [1]). Batching of orders is the process of transforming customers' product orders into sets of batches to be planned and subsequently assigned due date. This process is commonly practised in the industry such as this, since a batch is frequently shared by several orders with the earliest one determining the batch due date. Moreover, while the planner is carrying out this task, his/her objective is to minimize as much as possible the setups between products that are generated from incompatible families.

Despite the use of batch operations in many industries, their planning and scheduling have not been fully studied. This research identifies and addresses these characteristics and aims to pursue the development of a framework to solve a batching and sequencing problem for a realistically sized industrial case that can be easily implemented with inexpensive and readily available software. A three-tiered Hierarchical Production Planning (HPP) framework for single-stage, single machine, multi product batch plants with restricted batch sizes that originated from incompatible product families is developed. The framework integrates optimization and heuristic models for solving production planning and scheduling in a resin manufacturing plant.

This chapter is organized as follows. In the next section, the literature review is presented and followed by description of the production environment. The methodology, hierarchical framework and model formulations are then presented. Finally, model validation procedure, results and discussions and conclusions and future research are presented.

2 Literature Review

A rigorous mathematical analysis of HPP is found in the pioneering work of Hax and Meal [2] and Gabbay [3]. Since then, many theoretical works on the topic have followed (see Bitran and Tirupati [4]).

Many HPP solutions to industrial applications have been reported in the literature, such as the tile industry (Liberator and Miller [5]), steel manufacturing (Mackulak et al. [6]), milk powder manufacturing (Rutten [7]), paper industry (Bowers and Agarwal [8]), fibre industry (Qiu and Burch [9]), paint industry (Venkataraman and Smith [10]) , household chemicals (Das et. al [11]) and resin production (Omar and Teo [12]).

The literature on solution techniques used for solving the different levels of the HPP is vast, and we do not attempt to cover it all here. However, interested readers are referred to comprehensive reviews provided by Mckay et al. [13] and Okuda [14]. A brief literature review on disaggregate production planning and short-term sequencing models is provided here.

2.1 Disaggregation Production Planning

Disaggregation is a process of translating the aggregate plans into feasible detailed plans. The effectiveness of the aggregate plans depends on the existence of consistency and sound disaggregation. Without this linkage, decisions made at the aggregate level cannot be translated into cost savings at the shop floor. Disaggregation works within the boundaries established by the aggregate plan. It translates the objectives and goals as set by the aggregate plan into a workable program suitable for practical implementation. Many approaches for disaggregate plans have been reported in the literature. Bitran and Tirupati [4] presented a model that treated the disaggregate plan as a knapsack problem. Oliff [15] suggested a mixed integer programming formulation to solve the disaggregate problem for multi processor, multi product with conditional setups. Leong et al. [16] presented a weighted goal programming model that disaggregate the aggregate plans to lot-sizes and line assignment by product and group and finished goods inventories by product.

Omar and Teo [12] reported on a three-level HPP for a single-stage, identical parallel machines in the process industry with batch size restricted production environment that involves setup considerations. The authors reported that their proposed model can provide optimal solution to the production planning and scheduling problem considered, however, they conclude that their proposed model cannot solve a planning problem when the products involved exceeds 18 products originated from 5 incompatible families.

2.2 Short-Term Scheduling and Sequencing

Enormous solutions have been proposed for machine scheduling problems, and we do not attempt to cover it all here. However, interested readers are referred to the recent reviews by Toppan et al. [17] and Allahverdi et al. [18]. However, we will provide a brief review related to our work. Tardiness, defined as positive lateness of a job, incurs if a job is completed after its due date. In the weighted case, each job's tardiness is multiplied by a positive weight. The objective is to find a sequence of jobs that minimizes the total weighted tardiness in a single machine which is NP-hard in the strong sense (Lenstra et al. [19]). Adding the characteristics of jobs originated from incompatible families increases the difficulty of the problem of minimizing the total weighted tardiness in a single machine. Many practical industrial situations require the explicit consideration of setups and the development of appropriate scheduling tools. Among the reported cases, Pinedo [20] describes a manufacturing plant making paper bags where setups are required when the type of bag changes. A similar situation was observed in the plastic industry by Das et al. [21]. The aluminium industry has a casting operation where setups, mainly affecting the holding furnaces are required between the castings of different alloys (see Gravel et al. [22]).

Previous research done in the case of incompatible job families had been focused mostly on single machine batch problems. Fanti et al. [23] developed a heuristic that aims to minimize the makespan of jobs on a multi-product batch processing machine. Dobson and Nambimodom [24] considered the problem of minimizing the mean weighted flow time and provided an integer programming formulation to solve the problem. Azizoglu and Webster [25] described a branch and bound procedure to minimize total weighted completion time with arbitrary job sizes. Their procedure returns optimal solutions to problems of up to 25 jobs. Most recently, Perez et al. [26] developed and tested several heuristics to minimize the total weighted tardiness on single machine with incompatible job families. Their tests consistently show that the heuristics that uses Apparent Tardiness Cost (ATC) rule to form batches, combined with Decomposition heuristics (DH) to sequence jobs, perform better than other heuristics tested, except ATC combined with Dynamic Programming algorithms (DP). Their tests show that ATC-DH and ATC-DP results are close.

The literature is also not extensive either for single machine scheduling problems with sequence-dependent setups, where the objective is to meet delivery dates or to reduce tardiness. However, Lee et al. [27] have proposed the Apparent Tardiness Cost with Setups (ATCS) dispatching rule for minimizing total weighted tardiness. Among other authors who have treated the problem, we find Rubin and Ragatz [28] developed a genetic algorithm method while Tan and Narasimhan [29] used simulated annealing as a solution procedure. Tan et al.[30]

presented a comparison of four approaches for solving scheduling problem of single machine with the objective to minimize total tardiness in a sequence dependant setup environment. Their experiment results suggested that simulated annealing and random-start pairwise interchange are viable solution techniques that can yield good solutions to a large combinatorial problem and that the genetic algorithm had the worst performance.

3 Production Environment

This research is motivated by a planning and scheduling problem encountered in a multi-product batch operations of a chemical firm in South East Asia. This firm produces a variety of resin intermediates, which are used for coatings. It is a joint venture company of a leading global industry for high solids and waterborne coating formulations and powder, together with traditional solvent borne technology. Customers of this company include leading suppliers of automotive OEM coatings, vehicle refinishes, coatings for plastics, industrial wood finishes, metal and protective coatings. As a member of the global resins manufacturers, the resins company is able to leverage on knowledge and experience for its research and development works, production methodology and processes from other members of the group worldwide.

The resin manufacturing plant has two production lines and the major production reactions include alkylation, acyliction, aminotion, leading to the production of about 100 finished products.

The resin manufacturing plant operates on three shifts, and each production year has 358 days. Working capacity is around 742 tons and 633 tons per month for production line one and two respectively. The operation in each production line is a reaction process, where the chemical reaction takes a place in a reactor; mixing where chemicals are mixed in a thinning tank; filtering where purities are control to meet customers' specifications and packaging. Reaction is the bottleneck operation, hence the working capacities estimation are based on the reaction process. Demand of the finished products is considered to be high and, therefore, cannot be satisfied from production runs, since some of the available capacity is consumed for setups, so the firm allows backorder practices. Owing to storage limitations, the firm does not practice safety stock policy, and allows inventory for restricted fixed period of time. The workforce involved on the production is very limited and the plant management does not practice workforce variation policies.

When the demand estimates for the following year are ready, the marketing department passes these estimations to the production department to prepare the operating budget for the following year. Batching of orders process starts when the production planner receives customers' order due date. The ultimate objective of this process is to meet the customers' due date and minimize setups activities.

When considering large orders, the planner will consider resource family/production line dedication policy: In its simplest form, this policy that aims to utilize the available plant capacity, the planner usually adopts the following dedication policy: Families 1, 2 and 3 that comprises of 15 end products are to be produced in production line 1. The rest, 17 families, that consists of 85 products, are to be produced in production line 2.

The management of the company has become aware that the current manual planning procedure, which is adopted, is inadequate for coping with future planning problems. The management is also concerned for some time about the lack of formal integrated framework for production planning that facilitates coordination and collaboration among different managerial levels. The aim of this case study, therefore, is to develop for this company an integrated framework that can be easily implemented with inexpensive and readily available software.

In the following section, our methodology and the details of the developed mathematical models are presented.

4 Methodology

The HPP presented in this chapter (Figure 1) integrates family planning, item disaggregation planning and job sequencing scheduling into a complete planning and scheduling system.

In our proposed HPP, family demand is the input to the Aggregate Production Planning (APP) model. Since the setup is a major concern in the resin-manufacturing environment, the APP was formulated as a mixed-integer linear programming model. The objective of the APP model is to minimize total production, setup, inventory and backorder costs in a fixed workforce size, industrial environment.

The second level of the HPP is the Disaggregation Production Planning (DPP) as shown in Figure 1. The DPP receives several inputs. One of the inputs is the monthly item demand, which results from the disaggregation of family demand. Another input is the optimal outputs from the APP model which includes aggregate (family) production, inventory and backorder levels. These levels service as the parameters for the goal constraints of the DPP. Additional input to the DPP are the minimum and maximum batch-size for individual products and the man-hours consumed for setup activities.

The DPP problem was formulated as a weighted integer goal programming model. The rational for our choice is the fact that multiple objectives usually exist at this level of planning and the capability of goal programming to guarantee the consistency of disaggregation of aggregate feasible solutions. The objective of the DPP is to minimize the excess of production, inventory and backorder level targets set by the APP model. Moreover, the DPP model, while considering the minimum and maximum batch-size requirements, converts monthly optimal production levels into a monthly optimal number of batches. Determining the optimal number of batches is an important process, since companies in the process industry develop their scheduling in terms of batches to schedule rather than product orders to fill. The optimal output of DPP consists of number of batches, batch sizes, item production, inventory and backorder levels.

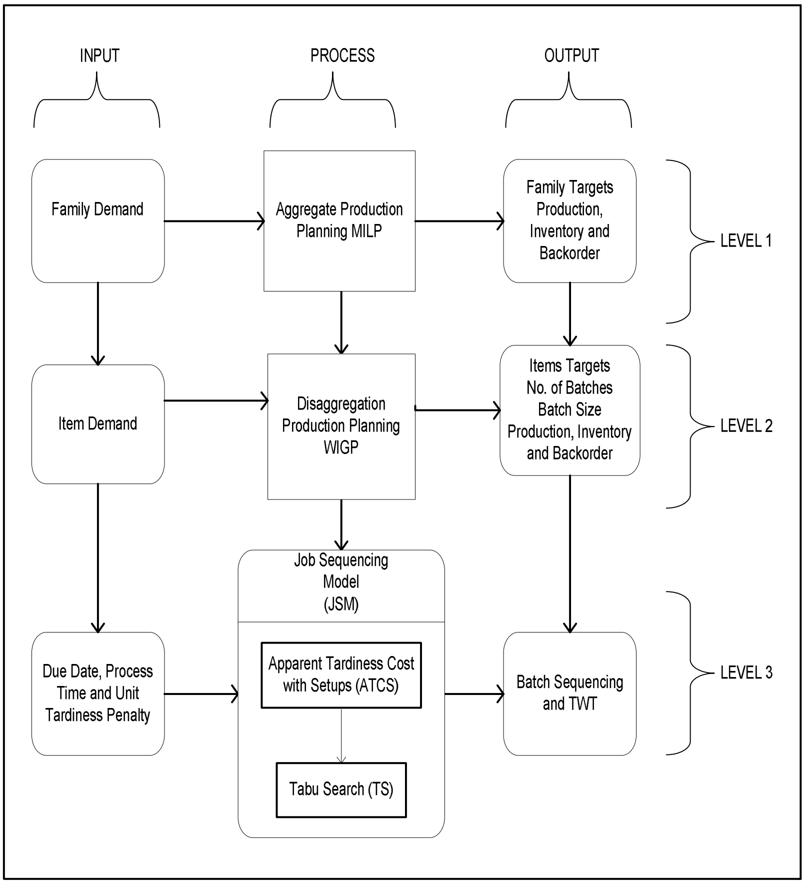

INPUT
PROCESS
OUTPUT
Family Demand
Aggregate Production Planning MILP
Family Targets Production, Inventory and Backorder
LEVEL 1
Item Demand
Disaggregation Production Planning WIGP
Items Targets No. of Batches Batch Size Production, Inventory and Backorder
LEVEL 2
Job Sequencing Model (JSM)
Due Date, Process Time and Unit Tardiness Penalty
Apparent Tardiness Cost with Setups (ATCS)
Tabu Search (TS)
Batch Sequencing and TWT
LEVEL 3

Fig. 1 Proposed hierarchical production planning framework

As can be seen from Figure 1, the third level of our proposed HPP is the job sequencing model (JSM). The input required by the JSM includes the number of batches (jobs) to be scheduled for each product, the product family each job belongs to, the process time, setup time, unit tardiness penalty and due dates. The JSM then determines the sequence of the batches to be processed to meet customer's due dates. In modeling the JSM, we have used an existing dispatching rule (Apparent Tardiness Cost with Setups-ATCS) for single machine with sequence dependent-setups. The ATCS seems to be a good choice since it has been reported in the literature to be one of the most successful dispatching heuristic used in the industry. Rather than just completely relying on the ATCS sequencing results, we decided to implement the suggestions in the literature in which the ATCS is combined with simulated annealing or Tabu-search to improve the sequencing results.

In our case, a Tabu-search (TS) was developed which takes the initial results provided by the ATCS and tries to generate a better solution.

The methodology described here presents a systematic approach that combines optimization and heuristic models to provide operational decision support for planning, scheduling and sequencing of jobs (batches) that originated from incompatible product families in the process industry. The proposed approach can provide important answers to questions that usually posed by the decision maker, such as the following questions: Which product is to be made in which period, how many batches and what is the batch size of each product to be produced per period and what is the exact sequence of orders that meet the customer's due date?

Next, the chapter will introduce the planning, scheduling and sequencing models that have been developed.

4.1 Aggregate Production Planning Model (APP)

A mixed-integer linear programming formulation is proposed for the APP. Model parameters, decision variables and formulations are presented next.

Indices

t	: period: 1,…, T.
i	: product family: 1,…,N.
l	: production line: 1,..L.

Parameters

Z_{it} : Unit production cost for product family i (excluding labour) in period t.

V_{it} Production setup cost for product family i in period t.

H_{it} : Unit inventory holding cost for product family i in period t.

CB_{it} : Unit backorder cost for product family i in period t.

CR_t : Manpower cost in period t.

D_{itl} : Demand for product family i in line l in period t.

SC_{tl} :Maximum available storage capacity in line l in period t.

Q_{tl} : Capacity available for production line l in period t.

$P_{il}^{\min}$:Minimum batch size for product family i in line l.

A_{il} :Unit process time for product family i (man-hour/ units) in line l.

G_{il} :Production setup time required for product family i in line l.

TR_{tl} :Total regular time available in period t in line l.

M_{il} :Upper bound on production of family i in line l in period t.

Decision Variables

x_{itl} : Production level of product family i in line l in period t.

h_{itl} :Inventory level of product family i in line l in period t.

b_{itl} :Backorder level of product family *i* in line *l* in period *t*.

ϕ_{itl} : Binary setup variable for product family *i* in line *l* in period *t*.

s_{tl} : Time consumed in setup activities in line *l* in period *t*, $s_{tl} = \sum_{i=1}^{N} G_{il}\phi_{itl}$.

w_{tl} : Time consumed in production activities in line *l* in period *t*, $w_{tl} = \sum_{i=1}^{N} A_{il}x_{itl}$.

$$Min\{ \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{l=1}^{L}(Z_{it}x_{itl} + V_{it}\phi_{itl}) + \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{l=1}^{L}(H_{it}h_{itl} + CB_{it}b_{itl}) + \sum_{t=1}^{T}\sum_{l=1}^{L}CR_t(w_{tl} + s_{tl}) \} \tag{1}$$

Subject to:

$$x_{itl} + h_{i,t-1,l} - b_{i,t-1,l} - h_{itl} + b_{itl} = D_{itl} \quad , \forall i,t,l \tag{2}$$

$$\sum_{i=1}^{N} h_{itl} \le SC_{tl} \quad , \forall t,l \tag{3}$$

$$\sum_{i=1}^{N} x_{itl} \le Q_{tl} \quad , \forall t,l \tag{4}$$

$$x_{itl} \ge P_{il}^{\min}\phi_{itl} \quad , \forall i,t,l \tag{5}$$

$$x_{itl} \le M_{il}\phi_{itl} \quad , \forall i,t,l \tag{6}$$

$$\sum_{i=1}^{N}(A_{il}x_{itl}) + \sum_{i=1}^{N}(G_{il}\phi_{itl}) \le TR_{tl} \quad , \forall t,l \tag{7}$$

$$x_{itl}, h_{itl}, b_{itl}, s_{tl}, w_{tl} \ge 0 \tag{8}$$

$$\phi_{itl} \in \{0,1\} \tag{9}$$

In the above formulation, equation (1) represents the objective function, which is to minimize the sum of production, setup, inventory, backorder, and workforce costs. Equation (2) is the demand, inventory and backorders constraint relationship. Equations (3) and (4) state the storage and capacity limitation. Equations (5), (6) and (9) enforce a minimum batch size requirement for each product family in each production line in each planning period. Equation (7) states, that the total labour capacity for each product family, in each planning period and production line is sufficient for both production and setup activities. Equation (8) is a non-negativity constraint.

4.2 Disaggregate Production Planning Model (DPP)

An integer-weighted goal-programming model is proposed for the DPP. The indices, parameters and decision variables are presented next.

Indices

t : period: 1,…, *T*.

i : product family: 1,…,*N*.

l : production line: 1,..*L*.
k : item: 1,…,*K*.

Inputs from the Aggregate Production Planning Model:
x_{itl} : Target production level of product family *i* in line *l* in period *t*.
h_{itl} : Target inventory level of product family *i* in line *l* in period *t*.
b_{itl} : Target backorder level of product family *i* in line *l* in period *t*.
s_{tl} : Man hour consumed in setup activities in line *l* in period *t*.

Parameters
Q_{tl} : Production capacity available for line *l* in period *t*.
D_{kitl} : Demand for item *k* of product family *i* in line *l* in period *t*.
SC_{tl} :Maximum storage capacity in line *l* in period *t*.
TR_{tl} :Total regular time available in line *l* in period *t*.
$P_{ikl}^{\min}$:Minimum batch size of item *k* of product family *i* in line *l*.
$P_{ikl}^{\max}$:Maximum batch size of item *k* of product family *i* in line *l*.
HB_{kil} : Processing hours per batch of item *k* of product family *i* in line *l*.

Decision Variables
x_{kitl} : Production level of item *k* of product family *i* in line *l* in period *t*.
h_{kitl} : Inventory level of item *k* of product family *i* in line *l* in period *t*.
b_{kitl} : Backorder level of item *k* of product family *i* in line *l* in period *t*.
d_{itl}^{1+} : Over production of product family *i* in line *l* in period *t*.
d_{itl}^{1-} : Under achievement of production levels of product family *i* in line *l* in period *t*.
d_{itl}^{2+} : Inventory excess of product family *i* in line *l* in period *t*.
d_{itl}^{2-} : Under achievement of inventory level of product family *i* in line *l* in period *t*.
d_{itl}^{3+} : Positive deviation of backorder level of product family *i* in line *l* in period *t*.
d_{itl}^{3-} : Negative deviation of backorder level of product family *i* in line *l* in period *t*.
η_{kitl} : Integer variable denoting the number of batches of item *k* of product family *i* produced in line *l* in period *t*.

$$Min\ \ W_1\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{l=1}^{L} d_{itl}^{1+} + W_2\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{l=1}^{L} d_{itl}^{2+} + W_3\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{l=1}^{L} d_{itl}^{3+} \qquad (10)$$

Subject to:

Goal Constraints:

$$\sum_{k=1}^{K} x_{kitl} + d_{itl}^{1-} - d_{itl}^{1+} = x_{itl}\ , \forall i,t,l \qquad (11)$$

$$\sum_{k=1}^{K} h_{kitl} + d_{itl}^{2-} - d_{itl}^{2+} = h_{itl}\ , \forall i,t,l \qquad (12)$$

$$\sum_{k=1}^{K} b_{kitl} + d_{itl}^{3-} - d_{itl}^{3+} = b_{itl} \text{ , } \forall i,t,l \tag{13}$$

System Constraints:

$$x_{kitl} + h_{k,i,t-1,l} - b_{k,i,t-1,l} - h_{kitl} + b_{kitl} = D_{kitl} \text{ , } \forall k,t,l \tag{14}$$

$$\sum_{k=1}^{K} \sum_{i=1}^{N} h_{kitl} \le SC_{tl} \text{ , } \forall t,l \tag{15}$$

$$\sum_{k=1}^{K} \sum_{i=1}^{N} x_{kitl} \le Q_{tl} \text{ , } \forall t,l \tag{16}$$

$$x_{kitl} \ge P_{kil}^{\min} \eta_{kitl} \text{ , } \forall i,k,t \, and \, l \tag{17}$$

$$x_{kitl} \le P_{kil}^{\max} \eta_{kitl} \text{ , } \forall i,k,t \, and \, l \tag{18}$$

$$\sum_{i=1}^{N} \sum_{k=1}^{K} (HB_{kil} \, \eta_{kitl}) + s_{tl} \le TR_{tl} \quad \forall t,l \tag{19}$$

$$x_{kitl}, h_{kitl}, b_{kitl}, \eta_{kitl}, d_{itl}^{1+}, d_{itl}^{1-}, d_{itl}^{2+}, d_{itl}^{2-}, d_{itl}^{3+}, d_{itl}^{3-} \ge 0 \tag{20}$$

Equation (10) presents the objective function that aims to minimize the excess of production, inventory and backorder target levels set by the aggregate production planning model. The relative importance of the goals is addressed by assigned weights W_1, W_2 and W_3 in each goal, in the objective function. Equations (11), (12) and (13) present the model production, inventory and backorder goal constraints. Equation (14) is the demand, inventory and backorders constraint relationship. Equation (15) enforces the available storage capacity and equation (16) presents production capacity limitations for each production line, in each planning period. Equations (17) and (18) determine the optimal number of batches and ensure that the monthly production quantities of the end items are within minimum and maximum batch size. Equation (19) enforces that the total batch processing time and the setup time incurred do not exceed total available time. Equation (20) is the non-negativity constraints.

4.3 Job Sequencing Model (JSM)

The JSM uses the idea of combining the ATCS and TS in order to solve a sequencing problem that involve batches (jobs) which are originated from incompatible product families in a single machine environment. The objective of the JSM is to find a sequence of jobs that minimizes the total weighted tardiness. The detail of the JSM is presented next.

4.3.1 Composite Dispatching Rule

The Apparent Tardiness Cost (ATC) heuristic is a composite dispatching rule developed by Lee et al. [27] that combines the Weighted Shortest Processing time (WSPT) and the Minimum Slack (MS) rule. Under the ATC rule, jobs are scheduled one at a time, that is, every time a machine becomes free, a ranking index is completed for each remaining job. The job with the highest-ranking index is then selected to be processed next. Several generalizations of the ATC rule have been developed. Such generalization, the Apparent Tardiness Cost with setups (ATCS) rule that has been designed to minimize the total weighted tardiness of jobs which are subject to sequence dependent setup times. The ATC and the ATCS heuristics are guaranteed to always produce a feasible schedule. The ATCS rule combines the Weighted Shortest Processing time (WSPT) rule, the Minimum Slack (MS) rule and the Shortest Setup Time (SST) rule in a single ranking index. The rule calculates the index of job j at time t when job l has completed its processing on the machine as

$$I_j(t,l) = \frac{w_j}{p_j}\exp\left(-\frac{\max(d_j - p_j - t, 0)}{K_1 \overline{p}}\right) \times \exp\left(-\frac{s_{lj}}{K_2 \overline{s}}\right) \text{ where}$$

w_j = unit tardiness penalty

p_j = processing time of job j

d_j = due date of job j

$\overline{p}$ = average process time

$\overline{s}$ = average setup time

$$\tau = 1 - \frac{\sum_{j=1}^{n} d_j}{n\hat{C}_{\max}}$$

$$R = \frac{d_{\max} - d_{\min}}{\hat{C}_{\max}}$$

$d_{\max}$ = maximum value of the due dates

$d_{\min}$ = minimum value of due date

$\hat{C}_{\max}$ = estimated makespan

$$\eta = \frac{\overline{s}}{\overline{p}}$$

$K_1 = 4.5 + R$ for $R \le 0.5$; $K_1 = 6.0 - 2R$ for $R \ge 0.5$;

$$K_2 = \frac{\tau}{2\sqrt{\eta}}$$

s_{lj} = setup time incurred when job j is processed after job l

A sequence of jobs is generated by selecting the job with the highest ranking index to be processed next. Once the sequence is obtained, the total weighted tardiness $\sum_{j=1}^{n} w_j T_j$ is calculated where

T_j = tardiness of job j = $\max\{0, C_j - d_j\}$

C_j = completion time of job j

The ATCS rule in this phase gives a sequence which provides an initial seed or solution for the Tabu Search.

4.3.2 Tabu Search

Tabu Search (TS), a technique for solving combinatorial optimization problems, is basically a strategy to overcome the local optimality. This is done by surpassing local optimality, by a strategy of penalizing certain moves during a number of iterations. TS basic step starts from a feasible solution *S* to the problem we are dealing with and then moves to another feasible solution $S^{\surd}$ belonging to the neighbourhood of S that optimizes an objective function over a set of neighbouring solutions. In order to prevent cycling, the procedure stores data concerning a certain number of recent moves, which are considered "tabu" moves, namely, prohibited. Readers interested to learn more about TS are referred to Glover [32, 33]. In this chapter we propose a TS heuristic that receives the sequencing results obtained from the ATCS and tries to improve the sequencing results. Our TS heuristic is summarized as follows:

The tabu search procedure is summarized as follows:

1. Set length of tabu list.
2. Set number of iterations.
3. Set initial tabu list as zero.
4. Let $S_1 =$ initial sequence from ATCS.
5. Set $S_0 =$ total weighted tardiness value from ATCS (TS_1) as best objective value from sequence S_1.
6. For $k = 2,...n;$ If the move from $S_{k-1} \to S_k$ is not prohibited by any mutation on the tabu list, select a candidate schedule S_k from the neighbourhood of S_{k-1}, which gives the minimum value for the total weighted tardiness value: (TS_k).
7. Update S_0 if $TS_k < TS_{k-1}$.
8. Update tabu list.
9. Increment *k* by 1.
10. Repeat steps 6-9 until maximum iteration.

4.3.3 An MILP Formulation

In this section, a mixed integer linear programming model for a single machine with the objective to minimize the total weighted tardiness is presented. The optimal results obtained from this model will be compared with the heuristic results.
Notations

$$X_{ijk} = \begin{cases} 1 & \text{if job j from family i is placed in position k} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{iik} = \begin{cases} 1 & \text{if setup } s_i \text{ is needed before a job at position k} \\ 0 & \text{otherwise} \end{cases}$$

m = number of families
n_i = number of jobs in family i.
n = total number of jobs
d_{ij} = due date of jth job in family i.
p_{ij} = processing time of jth job in family i.
w_{ij} = weight of jth job in family i.
C_k = completion time of the job at position k.
T_{ijk} = tardiness of the jth job in family i at position k.
s_i = family setup time of family i.

Formulation
Objective Function:

$$Min \sum_{i=1}^{m}\sum_{j=1}^{n_i}\sum_{k=1}^{n} w_{ij}T_{ijk} \tag{21}$$

Subject to

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} X_{ijk} = 1 \ , \ k = 1,2,...n \tag{22}$$

$$\sum_{k=1}^{n} X_{ijk} = 1 \ , \ i = 1,2,...m; \ j = 1,2,..,n_i \tag{23}$$

$$\sum_{j=1}^{n_i} X_{ijk} + \sum_{j=1}^{n_i} \sum_{p\in\{1,2,...m\}-\{i\}} X_{pj(k-1)} - Y_{ik} \le 1 \ , \ i = 1,2,...m; \ k = 1,2,..,n \tag{24}$$

$$C_1 = \sum_{i=1}^{m}\sum_{j=1}^{n_i} p_{ij}X_{ij1} \tag{25}$$

$$C_k = C_{k-1} + \sum_{i=1}^{m} s_i Y_{ik} + \sum_{i=1}^{m}\sum_{j=1}^{n_i} p_{ij}X_{ijk} \ , \ k = 2,3,...n \tag{26}$$

$$C_k - \sum_{i=1}^{m}\sum_{j=1}^{n_i} d_{ij}X_{ijk} \le \sum_{i=1}^{m}\sum_{j=1}^{n_i} T_{ijk}X_{ijk} \ , \ k = 1,2,...n \tag{27}$$

$$C_k \ge 0 \ , \ k = 1,2,...n \tag{28}$$

$$T_{ijk} \ge 0 \ , \ i = 1,2,...m; \ j = 1,2,...,n_i, \ k = 1,2,..,n \tag{29}$$

The objective function (21) of the model is to minimize the total weighted tardiness of the problem. Constraint (22) and (23) state that each position can be occupied by only one job and each job can be processed only once. Constraint (24) checks whether or not the preceding job and the following job are from the same family. If so, there is no setup time between them. Otherwise, a family setup time of the job in position k exists. Constraint (25) states the completion time of the job in the first position. Constraint (26) calculates the completion time from the second position to the last position of the sequence. Constraint (27) determines the tardiness values for all positions. Constraints (28) and (29) gives the non-negativity constraints.

5 Testing and Validation

The model's testing and validation are now illustrated using data taken from the company. In this research, Microsoft Excel is used to import and export data needed to execute all developed models for the three levels. The APP and DPP were developed using OPL Studio version 3.6 and solved using CPLEX version 8. The JSM is developed using Visual Basic 6.0. All models were executed using a personal computer with Pentium IV 2.80 GHZ processor. Our proposed testing and validation methodology is carried out in the following manner:

(1) Generating random demand: A crucial input to both the APP and DPP is the families and product demand. Using 17 months sales history of 100 products that originated from 20 product families, demand is generated using two parameters in the demand distribution as proposed by Xie et al. [35]. Demand variation (DV), represents the variability of the total demand and product-mix variation (MV), represents the variability in the proportion of the demand for each of the 100 products items in the total demand for two production lines. Three levels of DV factor are used. DV is set at 10% and 20% and 40% of the average total demand for three levels, respectively, representing the low, medium and high levels of variations in the normal random noise component of the total 100 products. The magnitudes of the noise component for MV are also varied at three levels. MV is set at 10%, 20% and 40% of the average proportion of individual demand for the three levels, respectively, representing the low, medium and high levels of variations in the normal random noise component of the total 100 products. The demand generation process which is based on real data obtained from the resin manufacturing firm, results in the development of 9 different demand variations and product-mix variations for each production line and was generated for 60 periods (months) with each period recorded after 4000 simulation runs.

(2) Validating the APP and the DPP levels: One of our objectives is to use a demand generation function that allows us to ensure that both the APP and DPP models are capable of developing optimal plans for the different demands and product-mix variations. It is worth mentioning that our intention in adopting this method of demand generation is different from that stated by Xie et al. [35]. Another objective is to test whether the DPP model is capable of converting production levels into optimal number of batches that can be processed.

(3) Validating the performance of the combined heuristic (ATCS + TS): An important question that requires an answer is this: How good are the quality of the schedules generated by the ATCS and the combined heuristic? In industrial application, heuristics were compared against other heuristics which may lead to a wrong judgement or conclusion about the quality of the schedules.

(see Ovacik and Uzsoy [34]). The authors developed an integer programming model that can answer the above important question. It is known that integer programming models can only deal with small size problems; therefore we have created a problem that consists of 10 jobs that originated from 2, 3 and 4 incompatible product families. The optimal results obtained from the integer programming model are compared with the results obtained from the ATCS and combined heuristic.

(4) Robustness issues: Once we were quite comfortable with our approach of the demand generation at (1), it is important to determine the behaviour of the solutions found by the proposed models. The first issue of concern would be to investigate the capability of the APP model to provide optimal solutions and how far in terms of the planning horizon the planner might consider. The second issue is to investigate the ability of the DPP model to convert finish product quantities into workable number of batches the planner may process. The third issue that need to be examined is the ability for the combined heuristic to provide optimal solutions and reasonable solution to the sequencing problem faced by the resin firm. The various demand scenarios mentioned above are used for investigating the robustness of the first and the second issues. For the third issue, we decided to use 20, 30, 40, 50 and 60 jobs with each job consisting of a single batch of a product. Families were made to vary between 10, 15 and 20 leading to the development of 15 sets. Every set was run 5 times using randomly generated due dates that lies between 1.0 and 7.0 (we assumed that each month has 30 working days and each week has 7 working days). The whole process, made available 75 cases ready for evaluation. The tardiness penalty for every job is assigned a weight 1, 2 or 3 according to the product family. The number of iterations of the TS is fixed at 500 and the tabu list is allowed a maximum of 5 pairs of jobs.

6 Results and Discussions

Turning to our experimental results, in which demand generated from the 9 scenarios for 60 time horizons, indicate that the APP model can provide optimal solution**.** However, since the APP is formulated as a mixed-integer linear programming problem, obtaining optimal solutions for longer time horizons would be a challenge to the model due to the increased number of integers. As for the DPP model, our investigations indicate that reliable optimal solutions are only possible if the planning horizon does not exceed 3 months, beyond that, computational time will increase exponentially as the number of integer variables increase. Consequently, the decision maker may not be able to obtain results in real time to be of any use for implementation purposes. However, this problem should not be considered as a drawback since in practice, companies use a rolling horizon planning methodology at this level of planning and typically will not exceed 3 months.

To demonstrate the capability of the DPP model to convert monthly production levels into workable optimal number of batches, the authors run the DPP model for 3 months time horizon using the developed 9 demand generating scenarios function. The summary of all computational results are shown in Table 1. For example, a close look to the first row and first column in Table 1 indicates that for the demand generation function that consist of DV 0.1 and MV 0.1, the planner needs to make available in January, 62 batches which is equivalent to 742000 kg from production line 1 in order to meet the demand for that particular month.

Table 1 Summary of monthly batches developed by the DPP

Demand Scenario	**January**		**February**		**March**	
	L1	**L2**	**L1**	**L2**	**L1**	**L2**
DV=0.1,MV=0.1	62	76	63	74	64	28
DV=0.1,MV=0.2	64	76	64	70	52	30
DV=0.1,,MV=0.4	64	75	64	65	40	19
DV=0.2,MV=0.1	61	75	62	75	61	36
DV=0.2,MV=0.2	63	71	60	72	59	69
DV=0.2,MV=0.4	63	74	65	67	56	26
DV=0.4,MV=0.1	64	77	62	61	61	51
DV=0.4,MV=0.2	61	72	62	72	33	19
DV=0.4,,MV=0.4	64	77	60	28	57	42

The authors developed Table 2 to explain the advantage of converting production levels into batches to meet variety of customers. In Table 2, we considered the result obtained in January for production line one (742000kg), where the planner only knows that 742000kg is produced out of 15 products. While the planner by using our developed batching method will have all the details shown in Table 2 columns 2 and 3, and obviously making the batching process more efficient. Returning to Table 1, it could be seen that the DPP model successfully converted production levels (quantities) into optimal number of batches making the batching of orders process job more efficient.

To examine how good are the schedules generated by the ATCS and the combined heuristic, the total weighted tardiness results obtained from using the ATCS and the combined heuristic are compared with the optimal total weighted tardiness (TWT) results obtained from using the MILP model. The Relative Error percentage, $\left(RE\% = \frac{TWT_{HEU} - TWT_{OPT}}{TWT_{OPT}} * 100 \right)$ is used to determine the performance of the ATCS and the combined heuristic.

The results of the performance of the ATCS and the combined heuristic are shown in Table 3. The ATCS is at most about 4.68% from the optimal solution whereas the combined heuristic provided a better solution which is about 1.17% away from the optimal solution. Moreover, the ATCS and the combined heuristic provided the same value as the MILP model for the total weighted tardiness for

Table 2 Development of Batches

Without batching of orders	With batching of orders	
742000kg which consists of 15 products	Products	No. of batches
	1	13
	2	9
	3	5
	4	6
	5	1
	6	4
	7	2
	8	2
	9	1
	10	5
	11	3
	12	1
	13	2
	14	2
	15	6
	Total	62

Table 3 Comparison of MILP, ATCS and (ATCS+TS)

	10 jobs originated from 2 families		
	MILP	**ATCS**	**(ATCS+TS)**
TWT	344	344	344
CPU Time(s)	53.17	5.94	11.16
RE(%)		0.00%	0.00%
	10 jobs originated from 3 families		
	MILP	**ATCS**	**(ATCS+TS)**
TWT	344	344	344
CPU Time(s)	73.99	9.17	11.27
RE(%)		0.00%	0.00%
	10 jobs originated from 4 families		
	MILP	**ATCS**	**(ATCS+TS)**
TWT	342	358	346
CPU Time(s)	95.05	10.35	11.27
RE(%)		4.68%	1.17%

the instance where 10 jobs originating from 2 and 3 families. This has shown that the combined heuristic can provide an optimal solution. Furthermore, our findings indicate that the approach that calls for combining the ATCS and TS to provide reasonably good solutions for jobs that originated from incompatible product families works.

Table 4, shows the summary of 15 problems set tested. In Table 4, each row starting from column 3 represents the computational average results based on 5 set problems. The performance measures shown in table 4 are as follows. Column (3) and (4) are, respectively, the total weighted tardiness (TWT) results of the ATCS and the combined heuristic (ATCS +TS). Column (5) represents the average CPU time in seconds consumed to solve the problem by the combined heuristic. The average number of setups that resulted from applying combined heuristic is shown in column (6).

Table 4 Summary of the 15 problem sets average computation results

No. of jobs	No. of families	TWT of ATCS	TWT of (ATCS+TS)	CPU time(s) of (ATCS+TS)	Average no. of setups
20	10	46.40	46.30	14.57	9
	15	113.20	113.16	15.72	14
	20	118.76	187.70	19.26	19
30	10	115.07	114.92	25.15	9
	15	216.10	216.06	28.26	14
	20	349.75	348.26	32.82	19
40	10	203.13	202.88	40.66	9
	15	331.21	330.41	42.68	14
	20	510.27	510.05	48.09	20
50	10	311.22	311.03	50.38	10
	15	505.41	504.19	58.82	15
	20	672.10	671.68	59.78	20
60	10	394.38	413.44	62.71	10
	15	607.87	606.73	67.40	15
	20	810.09	808.75	73.20	20

From the result of our experiments, we can make the following observations:

(1) Even though integer programming models cannot provide optimal solutions for industrial scheduling problems, they are a very important tool in testing the scheduling heuristics.

(2) As expected, the number of jobs (batches) and number of incompatible families have a significant impact on both the objective function (minimizing the total weighted tardiness value) and the CPU time consumed to obtain a solution. As it could be seen from Table 4, as the number of job increases, the

average total weighted tardiness and CPU time increases as well. It is worth noting that there is no significant difference in the CPU time consumed for the ATCS alone or the combined heuristic of ATCS and TS.

(3) ATCS and combined heuristic attempts to sequence the jobs within the same product family to be sequenced consecutively in order to reduce the setups. For example, jobs involving 10 families, the optimal number of setups is 9 times. This has been achieved for 20, 30 and 40 jobs whereas when the number of jobs increases to 50 and 60, the average number of setups is 10 indicating a minimum increment.

(4) For all the 75 cases tested (15 scenarios run for 5 times), no significant improvement of TWT value of the combined heuristic(ATCS+TS) over ATCS was found. This issue was investigated in order to find the reason behind this. Our findings indicate that based on our input data, the proposed heuristic attempts to improve the results obtained from the ATCS by producing a different sequence in which some of the jobs from the same family position are exchanged. It seems that the tightness of an instance is a critical factor in deciding the efficiency of the heuristic for the TWT problem. This is true since the proposed heuristic provides a significant improvement when used to solve the problem described in Pinedo [20] page 350. In this case, the authors compared the results obtained by Pinedo using ATCS and the results for the same problem by using the combined heuristic ATCS+TS. The study indicates that while ATCS provided a sequence that resulted in total weighted tardiness of 440, the combined heuristic ATCS+TS provided a sequence with a total weighted tardiness of 408.

7 Conclusions and Future Research

This research was initiated due to a planning and scheduling decisions problem encountered at a resins firm located in South East Asia. A three hierarchical framework is proposed to address this problem, which extends the literature of the hierarchical system and takes into consideration the shortcomings of the job scheduling and sequencing literature.

Although several hierarchical production planning (HPP) systems have been developed, there is strong evidence that the industrial implementations are minimal. The main reasons were a lack of communication between designers and users and the intrinsic difficulties of quantitative models in production planning. As many HPP systems focused only on the disaggregation of aggregate planning, the author realised the importance of integrating jobs sequencing decisions into production planning. Hence, in this research, a three-level hierarchical framework, which integrates production planning and job sequencing decisions, is proposed.

The performance of the integrated approach has been tested in two ways. First, the APP and DPP models were tested to ensure that they are capable of developing optimal solutions for the range of products tested. Secondly, the 75 cases of job sequencing problems were analyzed to show that the heuristic model for the JSM is capable of obtaining solution in a reasonable time frame. Heuristic method is preferred, as the integer programming cannot solve large scheduling problems as stated in the literature. The capability of the heuristic for large problems is tested by conducting experiments with many combinations of jobs and families. The heuristic model showed that it can provide a good sequence in terms of minimizing setup time in a reasonable computational time. Our proposed models assumed deterministic demand and market conditions and therefore an obvious extension would be to consider stochastic conditions at the APP and DPP levels. An opportunity for further research is the use of different heuristics models for the suggested methodology such as genetic algorithm or simulated annealing and considers probabilistic nature of job arrivals.

References

[1] Mendez, C.A., Henning, G.P., Cerda, J.: Optimal scheduling of batches plants satisfying multiple product orders with different due-dates. Computers and Chemical Engineering 24, 2223–2245 (2000)

[2] Hax, A.C., Meal, H.C.: Hierarchical integration of production planning and scheduling. In: Geisler, M. (ed.) Logistics. TIMS Studies in Management Science, vol. 1. North-Holland, American Elsevier, New York (1975)

[3] Gabbay, H.: A Hierarchical Approach to Production Planning. Technical Report No. 120, Operation Research Center, MIT, USA (1975)

[4] Bitran, G.R., Tirupati, D.: Hierarchical production planning. In: Graves, S.C., Rinnooy Kan, A.H.G., Zipkin, P.H. (eds.) Logistics of Production and Inventory, pp. 523–567. North-Holland, Amsterdam (1993)

[5] Liberator, J.M., Miller, T.: A hierarchical production planning system. Interface 15, 1–11 (1985)

[6] Mackulak, G.T., Moodie, C.L., Williams, J.T.: Computerized hierarchical production control in steel manufacturing. International Journal of Production Research 18, 455–465 (1980)

[7] Rutten, W.G.M.M.: Hierarchical mathematical programming for operational planning in a process industry. European Journal of Operational Research 64, 363–369 (1993)

[8] Bowers, M.R., Agarwal, A.: Hierarchical production planning: scheduling in the apparel industry. International Journal of Clothing Science and Technology 5, 36–43 (1993)

[9] Qiu, M.M., Burch, E.E.: Hierarchical production planning and scheduling in a multi-product, multi-machine environment. International Journal of Production Research 35, 3023–3042 (1997)

[10] Venkataraman, R., Smith, S.B.: Disaggregation to a rolling horizon master production schedule with minimum batch-size production restrictions. International Journal of Production Research 6, 1517–1537 (1996)

[11] Das, B.P., Richard, J.G., Shah, N., Macchietto, S.: An investigation on integration of aggregate production planning, master production scheduling and short-term production scheduling of batch process operations through a common data model. Computers and Chemical Engineering 24, 625–1631 (2000)

[12] Omar, M.K., Teo, S.C.: Hierarchical production planning and scheduling in multiproduct, batch process environment. International Journal of Production Research 45, 1029–1047 (2007)

[13] McKay, K.N., Safayeni, F., Buzacott, J.A.: A Review of hierarchical production planning and its applicability to modern manufacturing. Production Planning and Control 6, 384–394 (1995)

[14] Okuda, K.: Hierarchical structure in manufacturing systems: a literature survey. International Journal of Manufacturing Technology Management 3, 210–224 (2001)

[15] Oliff, M.D.: Disaggregate planning for parallel processors. IIE Transactions 19, 215–221 (1987)

[16] Leong, G.K., Oliff, M.D., Markland, R.E.: Improved hierarchical production-planning model. Journal of Operations Management 8, 90–114 (1989)

[17] Toppan, S., Joanne, M.S., Parthasarati, D.: Static scheduling research to minimize weighted and unweighted tardiness: A state of-the-art survey. International Journal of Production Economics 8, 1–12 (2003)

[18] Allahverdi, A., Ng, C.T., Cheng, T.C.E., Mikhail, Y.: A survey of scheduling problems with setup times or costs. European Journal of Operational Research 187, 985–1032 (2008)

[19] Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P.: Complexity of machine scheduling problems. Annals of Decision Mathematics 1, 342–362 (1997)

[20] Pinedo, M.: Scheduling theory, algorithms and systems, 2nd edn. Prentice Hall, New Jersey (2002)

[21] Das, S.R., Gupta, J.N.D., Khumawala, B.M.: A saving index heuristic algorithm for flowshope scheduling with sequence dependent set-up times. Journal of the Operational Research Society 46, 1365–1373 (1995)

[22] Gravel, M., Price, W., Gagnec, C.: Scheduling jobs in a Alcan aluminium factory using a genetic algorithm. International Journal of Production Research 38, 3031–3041 (2000)

[23] Fanti, M.P., Maione, B., Piscitelli, G., Turchiano, B.: Heuristic scheduling of jobs on a multi- product batch processing machine. International Journal of Production Research 34, 3263–3286 (1996)

[24] Dobson, G., Nambimadom, R.S.: The batch loading and scheduling problem. Operations Research 49, 52–65 (2001)

[25] Azizoglu, M., Webster, S.: Scheduling a batch process machine with incompatible job families. Computers and Industrial Engineering 39, 325–335 (2001)

[26] Perez, I.C., Fowler, J.W., Carlyle, W.M.: Minimizing total weighted tardiness on a single batch process machine with incompatible job families. Computers and Operations Research 32, 327–341 (2005)

[27] Lee, Y.H., Bhaskaran, K., Pinedo, M.: A heuristic to minimize the total weighted tardiness with sequence-dependent setups. IIE Transactions 29, 45–52 (1997)

[28] Rubin, P.A., Ragatz, G.L.: Scheduling in a sequence dependent setup environment with genetic search. Computers and Operations Research 22, 85–99 (1995)

[29] Tan, K.C., Narasimhan, R.: Minimizing tardiness on a single processor with sequencedependent setup times: a simulated annealing approach. Omega 25, 619–634 (1997)

[30] Tan, K.C., Narasimhan, R., Rubin, P.A., Ragatz, G.L.: A comparison of four methods for minimizing total tardiness on a single processor with sequence-dependent setup times. Omega 28, 313–326 (2000)

[31] Cicirello, V.A.: Weighted Tardiness Scheduling with Sequence-Dependent Setups: A Benchmark Library, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (2004), `http://www.cs.drexel.edu/~cicirello/benchmarks/wtsbenchmarks.pdf`

[32] Glover, F.: Tabu search-Part I. *ORSA* Journal on Computing 1, 190–206 (1989)
[33] Glover, F.: Tabu search-Part II. *ORSA* Journal on Computing 2, 4–32 (1990)
[34] Ovacik, T.M., Uzsoy, R.: Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times. International Journal of Production Research 32, 1243–1263 (1994)
[35] Xie, J., Zahao, X., Lee, T.S.: Freezing the master production schedule under single resource constrain and demand uncertainty. International Journal of Production Economics 83, 65–85 (2003)
[36] Vollmann, T.E., Berry, W.L., Whybork, D.L.: Manufacturing planning and control systems, 3rd edn. Homewood, Irwin (1992)

Evolutionary Algorithms in the Optimal Sizing of Analog Circuits

Esteban Tlelo-Cuautle, Ivick Guerra-Gómez, Luis Gerardo de la Fraga, Georgina Flores-Becerra, Said Polanco-Martagón, Mourad Fakhfakh, Carlos Alberto Reyes-García, Gustavo Rodríguez-Gómez, and Gerardo Reyes-Salgado

Abstract. Analog signal processing applications such as filter design and oscillators, require the use of different kinds of amplifiers, namely: voltage follower, current conveyors, operational amplifiers and current feedback operational amplifiers. To improve the performances on these applications, it is very much needed to optimize the behavior of the amplifiers. That way, this work shows their optimization by applying two evolutionary algorithms: the Non-Sorting Genetic Algorithm (NSGA-II), and the Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D). The analog circuits are sized taking into account design constraints, and linking HSPICE like circuit simulator to evaluate their electrical characteristics. Additionally, we show that differential evolution (DE) enhances the convergence to the Pareto front and controls the evolution of the objectives among different runs. DE also preserves the same time efficiency and increases the dominance on NSGA-II and MOEA/D compared with the one point crossover genetic operator.

1 Introduction

Analog signal processing applications require the use of different kinds of analog electronic devices. For instance, in [1] an analysis of the state-of-the-art of active

Esteban Tlelo-Cuautle · Ivick Guerra-Gómez · Carlos Alberto Reyes-García · Gustavo Rodríguez-Gómez
INAOE, Mexico,
e-mail: etlelo@inaoep.mx

Luis Gerardo de la Fraga
CINVESTAV, Computer Science Department, Mexico

Georgina Flores-Becerra · Said Polanco-Martagón
Instituto Tecnológico de Puebla, Mexico

Mourad Fakhfakh
University of Sfax, Tunisia

Gerardo Reyes-Salgado
CENIDET, Mexico

devices is introduced, where a methodology to generate new ones is proposed. An important thing is that the majority of the active devices, already known or new ones, can be designed by using four unity-gain cells (UGCs), namely: voltage follower, voltage mirror, current follower, and current mirror [2,3]. The UGCs can be synthesized automatically as already shown in [4]. Furthermore, they can be evolved to design more complex devices, for example: current conveyors (CCs) [5], and current feedback operational amplifiers (CFOAs) [3]. The behavior of UGCs, CCs and CFOAs can be modeled by applying symbolic analysis [6,7], to speed-up time simulation. Other circuit synthesis approaches can be found in [8,9,10,11,12,13,14], all of them apply evolutionary or swarm intelligence in the optimization process.

Some applications of UGCs, CCs and CFOAs include sinusoidal and chaotic oscillators [15,16,17,18,19,20]. However, to improve the performances of these applications, it is very much needed to optimize the behavior of the active devices. Although some optimization approaches have been already presented in [21,22,23,24,25,26,27], the analog circuit optimization problem is yet unsolved. This is due to the fact that analog circuit design automation tools are not developed at the same speed of technology [28]; analog circuit design remains dependent on the designer's experience.

Besides, the actual tendency is to apply evolutionary algorithms (EAs) [29,30,31,32,33,34,35,36], combined with intelligent techniques [10,11,12,21,22,23,37,38], to solve the sizing optimization problem in analog circuit design including nanoscale CMOS devices [39]. That way, this work shows the performances of applying the Non-Sorting Genetic Algorithm (NSGA-II) [29], and the Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) [31], to automatically size analog circuits such as UGCs, CCs and CFOAs taking into account design constraints, and linking HSPICE like simulator to evaluate electrical characteristics. The EAs described herein are evaluated by applying ZDT1–4 test functions from [30]. An important point is that the performance of the proposed optimization system depends on the generation of solutions to the optimization problem (sizing analog integrated circuits), or in other words, our system performance depends on the generation of the true Pareto front [21].

In Sect. 2 the design characteristics of the analog circuits UGC, CC, and CFOA are shown. This section also summarizes the sizing problem. In Sect. 3 the main characteristics of NSGA-II and MOEA/D are presented, and these algorithms are tested by applying the functions given in [30]. In Sect. 3.4, the genetic operators one-point crossover and differential evolution (DE) are described. In Sect. 4, the optimization results of applying NSGA-II and MOEA/D, with and without DE, to the analog circuits design are shown. Finally, in Sect. 5 we summarize the main conclusions and future lines for research.

2 Analog Circuit Sizing

Nowadays, progresses made in the VLSI technology are leading to the full integration of mixed analog/digital circuits [14]. Even though the analog part presents a small portion of the entire circuit, its design is a very complicated task that generally relies on the experience of the skilled designer. Despite its importance, analog design automation still lags behind that of digital circuits. Besides, analog circuit design is a hard and tedious work due to the large number of parameters, constraints and performances that the designer has to handle [8, 9, 10, 11, 12, 13, 21, 22, 23, 24, 25, 26, 27, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41].

As highlighted in [11, 12, 21, 22, 26, 40], since analog designs are becoming more and more complex, there is a pressing need for analog circuit design automation (ADA), to meet the time to market constraints. In this manner, this chapter shows the characteristics of two multiobjective EA's: NSGA-II and MOEA/D to contribute to solve the sizing problem of analog integrated circuits [41]. Basically, an EA searches for the optimal width (W) and length (L) of the MOSFETs to accomplish target specifications. Furthermore, it is shown that NSGA-II and MOEA/D give much better results than the classically used statistic based approaches [23, 27, 31]. However, a very important set of major problems seems to be still open; mainly problems related to the number of parameters, the numbers and nature of constraints and objective functions [14, 31, 32, 33, 34, 35], etc. Added to that the fact that almost all circuit design problems comprise conflicting performance trade-offs among electrical characteristics, i.e. improving one performance leads to the degradation of another one; the use of multiobjective approaches is unavoidable [22]. Furthermore, due to the increasing complexity of the considered circuits and technology, developed and handled performance models [7], as well as constraints, suffer from approximations that the designer is forced to adopt [14].

The circuits to be sized by applying the EAs NSGA-II and MOEA/D are the voltage follower shown in Fig. 1, the positive-type second generation current conveyor (CCII$^+$) shown in Fig. 2, and the current-feedback operational amplifier (CFOA) shown in Fig. 3. As one can see, the CCII$^+$ is designed from the evolution of the VF, and the CFOA is designed from the cascade connection between the CCII$^+$ and the VF [3]. This systematic design approach is quite interesting to show that the optimized sizes of the VF can change when it is evolved to design the CCII$^+$, and when it is used to design the CFOA. In the following section is shown the encoding of these circuits to be optimized by using standard CMOS technology of 0.18 μm, $V_{ss} = -V_{dd} = 2$ V, $I_{ref} = 50$ μA and a load capacitor of 1 pF.

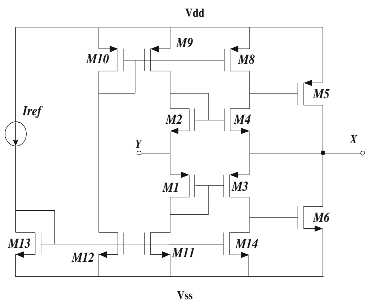


Fig. 1 Voltage Follower (VF)

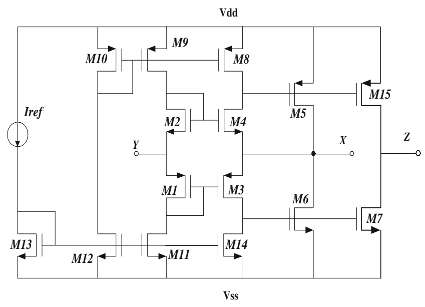


Fig. 2 Current Conveyor (CCII$^+$)

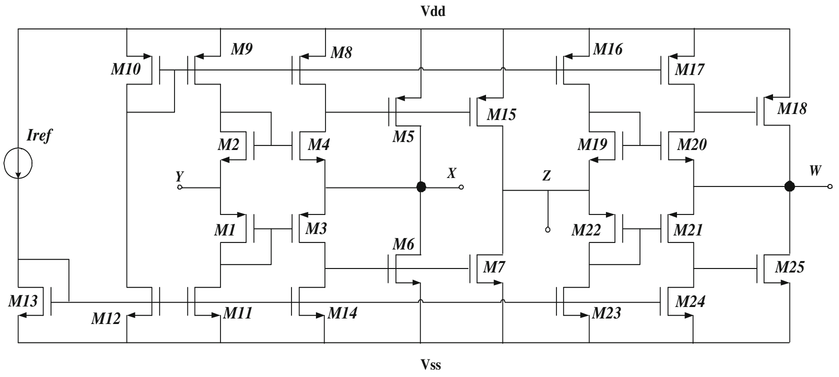


Fig. 3 Current-Feedback Operational Amplifier (CFOA)

3 Evolutionary Algorithms

Evolutionary algorithms are used to solve multiobjective optimization problems, and these solvers are incorporated into tools for analog circuit design. In this manner, changing the number of objectives or variables, and even with different magnitudes, is very easy to perform [22, 34, 35, 36]. Additionally, constraints can be taken into account under user defined limits, between design parameters and electrical characteristics [37, 38]. In this manner, this chapter shows the characteristics of the two selected EAs: NSGA-II and MOEA/D, for the optimization of the three analog circuits described in Sect. 2. Both EAs are tested by applying the functions given in [30].

3.1 Multi-Objective Optimization

A Multi-objective Optimization Problem (MOP) can be formally defined as the problem of finding the vector: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$ which satisfies the k inequality constraints:

$$g_i(\mathbf{x}) < 0; \text{ for } i = 1, 2, \ldots k$$

the p equality constraints

$$h_j(\mathbf{x}) = 0; \text{ for } j = 1, 2, \ldots p$$

and minimizes the vector function

$$\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$$
$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^{\mathrm{T}}$$

In other words, we aim to determine from among the set S of all vectors (points) which satisfy the constraints those that yield the optimum values for all the m objective functions simultaneously. The constraints define the feasible region S and any point $\mathbf{x}$ in the feasible region is called a feasible point.

Pareto dominance is formally defined as follows: A vector $\mathbf{u} \in \mathbb{R}^m$ is said to dominate a vector $\mathbf{v} \in \mathbb{R}^m$, $\mathbf{u} \prec \mathbf{v}$, if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e., $\forall i \in \{1,\ldots m\}$, $u_i \leq v_i \wedge \exists i \in \{1,\ldots m\}$: $u_i < v_i$, (assuming minimization). In order to say that a solution dominates another one, this one needs to be strictly better in at least one objective, and not worse in any of them.

The formal definition of Pareto optimality is provided next: A solution $\mathbf{x}_u \in S$ (where S is the feasible region) is said to be Pareto optimal if and only if there is no $x_v \in S$ for which $\mathbf{v} = \mathbf{f}(\mathbf{x}_v) \in \mathbb{R}^m$ dominates $\mathbf{u} = \mathbf{f}(\mathbf{x}_u) \in \mathbb{R}^m$, where m is the number of objectives. In words, this definition says that $\mathbf{x}_u$ is Pareto optimal if there exists no feasible vector $\mathbf{x}_v$ which would decrease some objective without causing a simultaneous increase in at least one other objective (assuming minimization). This definition does not provide us a single solution (in decision variable space), but a set of solutions which form the so-called Pareto Optimal Set or Pareto Front. All the vectors that correspond to the solutions included in the Pareto Front are non-dominated.

3.2 NSGA-II

This algorithm approximates the Pareto Front of a multi-objective optimization problem (MOP) by sorting and ranking all solutions in order to choose the better solutions to make new offspring. This means, by ranking all the population in different Pareto subfronts that it will be possible to know which solutions show better performance. In this algorithm a form to choose the best solution between two solutions, in the same subfront preserving diversity, is contemplated; in this form it is possible to select the best part of a population without losing diversity. NSGA-II is based on three main issues : *Fast Ranking Function* , *Crowding Distance Assignment* and elitism. These procedures, and the fact that constraints can be added easily, ensure that the solutions are feasible.

At the beginning it is necessary to randomly initialize the parameters and start by building two populations (P_o and Q_o) of size N, from random values into a feasible region. The NSGA-II procedure in each generation consists of rebuilding the actual population (R_t) from the two original populations (P_t and Q_t) then the new size of the population will be N, where t represents the actual generation.

Through a nondominated sorting, all solutions in R_t are ranked and classified in a family of subfronts. In the next step, it is necessary to create from R_t (previously ranked and ordered by subfront number) a new offspring (P_{t+1}), the objective is to choose, from a population of size $2N$, the N solutions which belong to the first subfronts. In this manner, the last subfront could have more than the necessary individuals; therefore ($i_{distance}$) measure is used to identify the best solutions, a process that preserving elitism. And for preserving diversity, the solutions that are far of the rest are selected; this is possible simply by modifying a little bit the concept of Pareto dominance as follows:

$$i \prec_n j \text{ if } (i_{\text{rank}} < j_{\text{rank}})\text{or } \left[(i_{\text{rank}} = j_{\text{rank}}) \text{ and } (i_{\text{distance}} > j_{\text{distance}})\right] \tag{1}$$

Algorithm 1. NSGA-II

```
 1: $P_0 \leftarrow$ random, $Q_0 \leftarrow$ random                                   //initialize
 2: $t \leftarrow 0$
 3: Evaluate($P_0$), Evaluate($Q_0$)
 4: **repeat**
 5:     $i \leftarrow 1$
 6:     $P_{t+1} \leftarrow \emptyset$
 7:     $R_t \leftarrow P_t \cup Q_t$                  //combine parent and offspring population
 8:     evaluate($R_t$)
 9:     $F \leftarrow$fast-ranking($R_t$)     //$F = (F_1, F_2, ...)$ , all nondominated fronts of $R_t$
10:     **while** $|P_{t+1}| + |F_i| \leq N$ **do**
11:         $P_{t+1} \leftarrow P_{t+1} \cup F_i$          //include $i$th nondominated front in the parent pop
12:         $i \leftarrow i+1$                    //include $i^{th}$ nondominated front in the parent pop
13:     **end while**
14:     crowding-distance-assignment($F_i$)  //calculate crowding-distance in last $F_i$ to achieve $N$
15:     sort($F_i$,$\prec_n$)                          //sort in descending order using $\prec_n$
16:     $P_{t+1} \leftarrow P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$     //choose the first $(N - |P_{t+1}|)$ elements of $F_i$
17:     $Q_{t+1} \leftarrow$ make-new-pop($P_{t+1}$)          //use crossover and mutation to create it
18: **until** stop criteria
```

3.2.1 Fast Ranking Function

This procedure is responsible to rank each solution into a subfront, and it starts by selecting the nondominated solutions among the current population (R_t). This first group of solutions will be labeled as the solutions into the first subfront (F_1) and are separated from R_t. For the remaining solutions in R_t the nondominated solutions are selected again, but this time they are labeled into the second subfront (F_2) and separated from R_t like the solutions in (F_1) were separated before. This procedure continues until all solutions in R_t are ranked into a subfront.

The procedure uses a counter for each solution. Such counter allows to know how many solutions dominate to other solutions (n_p where p is the *p-solution*). In the same way, there is a set which contains all the solutions dominated for each solution (all solutions in S_p are dominated by *p-solution*). First, the solutions with counter equal to zero are taken, and to each reminded solution are decreased its counter in one. In this way, the next subfront is composed by the remaining solutions with counter equal to zero. This continues until all solutions have been ranked.

3.2.2 Crowding Distance Assignment

This is the second procedure to select those solutions which generate the offspring, and it makes sense when it is necessary to choose the last members of the population

Algorithm 2. Fast Ranking Function Algorithm

```
1: for each $p \in P$ do
2:     $S_p \leftarrow \emptyset$
3:     $n_p \leftarrow 0$
4:     for each $q \in P$ do
5:         if $(p \prec q)$ then
6:             $S_p \leftarrow S_p \cup \{q\}$
7:         else if $(q \prec p)$ then
8:             $n_p \leftarrow n_p + 1$
9:         end if
10:        if $n_p \leftarrow 0$ then
11:            $p_{rank} \leftarrow 1$
12:            $F_1 \leftarrow F_1 \cup \{p\}$
13:        end if
14:    end for
15: end for
16: i $\leftarrow 0$
17: while $F_i \neq \emptyset$ do
18:     $Q \leftarrow \emptyset$
19:     for each $p \in F_i$ do
20:         for each $q \in S_p$ do
21:             $n_q \leftarrow n_q - 1$
22:             if $n_q \leftarrow 0$ then
23:                 $q_{rank} \leftarrow i + 1$
24:                 $Q \leftarrow Q \cup \{q\}$
25:             end if
26:         end for
27:     end for
28:     $i \leftarrow i + 1$
29:     $F_i \leftarrow Q$
30: end while
```

P_{t+1} into a subfront. In this manner, the crowding distance allows a measure to chose members into the same subfront. The main idea is to perform a density estimation named *crowding distance* ($i_{distance}$) by sorting in ascending order the solutions for each objective function, then for each objective it is first selected the smallest and largest limit found and an infinite value is assigned to their crowding distances.

Algorithm 3. Crowding Distance Assignment

```
1: $l \leftarrow |T|$                                  //number of solutions in $|T|$
2: for each $i$ do
3:     set $T[i]_{distance} \leftarrow 0$                  //initializing
4:     for each objective $j$ do
5:         $T \leftarrow$ sort$(T, j)$                //sort according each objective, $1 \leq j \leq m$
6:         $T[1]_{distance} \leftarrow T[l]_{distance} \leftarrow \infty$ //boundaries are always selected for all other points
7:         for $i \leftarrow 2$ to ($l$-1) do
8:             $T[i]_{distance} \leftarrow T[i]_{distance} + (T[i+1] \cdot j - T[i- \quad 1] \cdot j)/(f_j^{max} - f_j^{min})$
9:         end for
10:    end for
11: end for
```

Next, for the rest of the functions values, their crowding distance are calculated as follows: with the sorted objective values take the immediately maximum and minimum function value of each objective, and the difference of these two values is normalized (using the global maximum and global minimum of the current function). Finally, the main procedure of NSGA-II could be summarized in Algorithm 1.

3.3 MOEA/D

The basic idea of MOEA/D is the decomposition of a multiobjective problem in scalar optimization subproblems by a *weights vector*. This vector associates a weight ($\boldsymbol{\lambda}$) for each subproblem which is considered as a single individual in the population and it is going to try to improve itself and to its nearby (*neighborhoods*).

Algorithm 4. MOEA/D pseudocode

```
1:  $t = 1$ , $P$=random(N,n) , set $EP = \emptyset$ , $T$
2:  build an uniform spread of N weight vectors $\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_N$
3:  for $i = 1, 2, \dots, N$ do
4:      $B(i) = \{\boldsymbol{\lambda}_{i_1}, \dots, \boldsymbol{\lambda}_{i_T}\}$
5:  end for
6:  evaluate population $P$
7:  save best solutions in EP
8:  repeat
9:      for $i = 1, 2, \dots, N$ do
10:         randomly select parents from $B(i)$
11:         generate new individual $\mathbf{y}$
12:         for each $\ell \in B(i)$ do
13:             if $g(\mathbf{y} \mid \boldsymbol{\lambda}_\ell, \mathbf{Z}^*) \leq g(\mathbf{x}^\ell \mid \boldsymbol{\lambda}_\ell, \mathbf{Z}^*)$ then
14:                 $\mathbf{x}^\ell = \mathbf{y}$
15:                 $\mathbf{f}^\ell(\mathbf{x}^\ell) = \mathbf{f}(\mathbf{y})$
16:             end if
17:         end for
18:     end for
19:     all vectors dominated by $\mathbf{f}(\mathbf{y})$ are removed from EP
20: until stop criteria
```

After the initialization of the parameters the first step in MOEA/D is related to find the N spread weight vectors (to each individual $\mathbf{x}_{i,\ 0 \leq i \leq N}$ corresponds one $\boldsymbol{\lambda}_i = \{\lambda_i^1, \lambda_i^2, \dots, \lambda_i^m\}$ where m is the number of objective functions). One way can be by using a parameter H in a sequence as described by (2):

$$\left\{ \frac{0}{H}, \frac{1}{H}, \dots, \frac{H}{H} \right\} \tag{2}$$

Therefore, for $m = 2$, $N = H + 1$, with $m > 2$ the number of such vectors is given by (3):

$$N = C_{H+m-1}^{m-1} \tag{3}$$

Now for each individual in the population corresponds one $\boldsymbol{\lambda}_i$. Therefore, it is possible to define a number (T) of neighborhoods for each $\boldsymbol{\lambda}_i$ and they will be saved in B.

In each generation there is a population of N points $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N \in X$ where $\mathbf{x}^i$ is the current solution to the i_{th} subproblem and there are $\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^N$, where $\mathbf{f}^i$ is the $\mathbf{f}$-value of $\mathbf{x}^i$.

In the procedure it is necessary to generate a new individual $\mathbf{y}$ which will be compared with its neighborhood by applying a decomposition approach ($g[\mathbf{x}^i \mid \boldsymbol{\lambda}_i, \mathbf{Z}^*]$) such as the *Tchebycheff Approach* and each neighbor worse than this new individual will be replaced by it in an external population (EP) which is used to store non-dominated solutions.

In the Tchebycheff Approach, the scalar optimization problem is described by (4), where $\mathbf{Z}^* = [z_1^*, z_2^*, \ldots, z_m^*]^{\mathrm{T}}$ is the vector of the best objective functions values found. Algorithm 4 shows the steps performed by MOEA/D :

$$g(\mathbf{x}^i \mid \boldsymbol{\lambda}_i, \mathbf{Z}^*) = \max\{\lambda_i^j |f_j(\mathbf{x}^i) - z_j^*|\}_{\substack{1 \le i \le N \\ 1 \le j \le m}} \tag{4}$$

3.4 Genetic Operators: Differential Evolution

Evolutionary algorithms have been tested by including the Differential Evolution (DE) operator to improve convergence and diversity [34,36]. Such operator consists of randomly choosing three solutions: $\mathbf{x}_a, \mathbf{x}_b$ and $\mathbf{x}_c$ from the population X. A new solution $\mathbf{x}_{new} = x_{new}^1, x_{new}^2, \ldots, x_{new}^n$ is generated in the following way [42]:

$$\mathbf{x}_{\text{new}}^k = \begin{cases} x_a^k + R \cdot (x_b^k - x_c^k) & \text{if rand()} < C, \\ x_a^k & \text{otherwise,} \end{cases} \qquad \text{for } k = 1, 2, \ldots, n. \tag{5}$$

Where R is a constant factor which controls the amplification of the differential variation, C is the cross-over probability and rand() is a function that returns a random real number in the interval $[0, 1)$. In this work we set R and C to constant values, recommended values for these constants are $R \in [0.5, 1.0]$, $C \in [0.8, 1.0]$ [43]. In [44] these parameters are self-adapted by including them inside the optimization problem.

Iorio and Li [45] propose the Nondominated Sorting Differential Evolution (NSDE). This approach is a simple modification of the NSGA-II. The only difference between this approach and the NSGA-II is the method for generating new individuals. The NSGA-II uses a real-coded crossover and mutation operator, but in the NSDE, these operators are replaced with the DE's operators. New candidates are generated using the DE. On the other hand, Qingfu and Li [46] use DE as genetic operator in MOEA/D. In our proposal, we include DE into NSGA-II and MOEA/D to show its suitability in sizing analog circuits. Furthermore, the results shown in the next section, provide diversity and good behavior of the generations to find optimal solutions.

3.5 Behavior of NSGA-II and MOEA/D on Test Functions ZDT

Zitzler *et al.* [30], provided a comparison of various evolutionary approaches to multiobjective optimization using six carefully chosen test functions, each one involves a particular feature, that is known to cause difficulty in the evolutionary optimization process, mainly in converging to the Pareto-optimal front. In Table 2, the four test functions used in this work are shown, where $ZDT1$, $ZDT2$, $ZDT3$ and $ZDT4$ are the name of the selected functions, n is the number of variables used for the functions, and bounds are the maximum and minimum search limits for each variable. For the four selected functions the goal is to minimize the functions given. The test function $ZDT1$ has a convex Pareto-optimal front, $ZDT2$ is the nonconvex contrapart to $ZDT1$. $ZDT3$ represents the discreteness feature, its Pareto-optimal front consist of several noncontiguous convex parts. $ZDT4$ contains 21^9 local Pareto-optimal fronts then it tests the EA's ability to deal with multimodality.

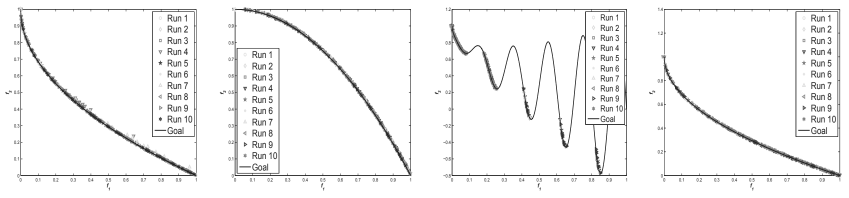
The test functions were evaluated (in a dual processor 1.2GHz, RAM 2GHz) over ten runs with a population size of 100 along 250 generations, by using DE and single point crossover (SPC) as genetic operators. Figures 4 - 7 depict the behavior of each method with each genetic operator. Table 2 shows the maximum value (MAX), minimum (MIN), average value (AVG), standard deviation (STD), average error (ERR) between the objective values found and the goal, and finally, the average run time (TMR). It is possible to see how DE preserves or improves the SPC performance in most cases, but for other ones, the average error worse, as in NSGA-II for ZDT3. However by comparing Figures 4(c) and 5(c) (or Figures 6(c) and 7(c)), DE notably

Table 1 Test Functions

Function	n	Bounds	Functions	Optimal Sol.
$ZDT1$	15	$x_i \in [0,1]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1 - \sqrt{\frac{x_1}{g(\mathbf{x})}}\ ]$ $g(\mathbf{x}) = 1 + \frac{9}{n-1}\ \sum_{i=2}^{n} x_i$	$x_1 \in [0,1]$ $x_i = 0,$ $i = 2,\ldots,n$
$ZDT2$	15	$x_i \in [0,1]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1 - \ (\frac{x_1}{g(\mathbf{x})})^2\ ]$ $g(\mathbf{x}) = 1 + \frac{9}{n-1}\ \sum_{i=2}^{n} x_i$	$x_1 \in [0,1]$ $x_i = 0,$ $i = 2,\ldots,n$
$ZDT3$	15	$x_i \in [0,1]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1 - \sqrt{\frac{x_1}{g(\mathbf{x})}} - \frac{x_1}{g(\mathbf{x})}\sin(10\pi x)]$ $g(\mathbf{x}) = 1 + \frac{9}{n-1}\ \sum_{i=2}^{n} x_i$	$x_1 \in [0,1]$ $x_i = 0,$ $i = 2,\ldots,n$
$ZDT4$	15	$x_1 \in [0,1]$ $x_{2,\ldots,n} \in [-5,5]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1 - \sqrt{\frac{x_1}{g(\mathbf{x})}}\ ]$ $g(\mathbf{x}) = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10\cos(4\pi x_i)]$	$x_1 \in [0,1]$ $x_i = 0,$ $i = 2,\ldots,n$

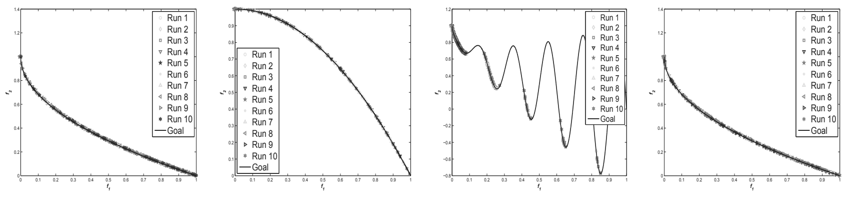
improves the convergence to the goal. In the same way, DE improves the convergence for MOEA/D for ZDT4 as Figures 6(d) and 7(d) show.

In this manner, we choose DE as genetic operator for this work due to its capability to improve the convergence by preserving the run time performance.



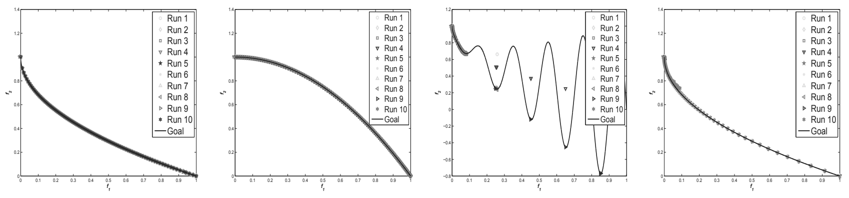
(a) ZDT1 (b) ZDT2 (c) ZDT3 (d) ZDT4

Fig. 4 NSGA-II with single point crossover for ZDT functions



(a) ZDT1 (b) ZDT2 (c) ZDT3 (d) ZDT4

Fig. 5 NSGA-II with differential evolution for ZDT functions



(a) ZDT1 (b) ZDT2 (c) ZDT3 (d) ZDT4

Fig. 6 MOEA/D with single point crossover for ZDT functions

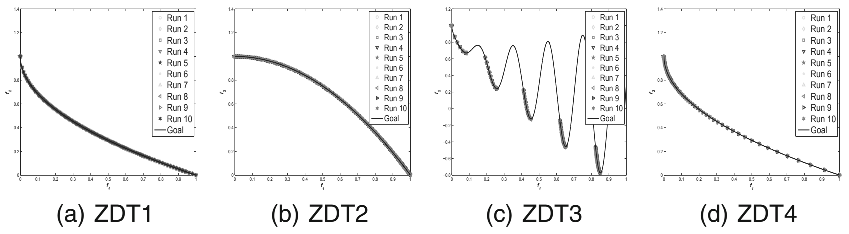


Fig. 7 MOEA/D with differential evolution for ZDT functions

Table 2 Results on Test Functions

Function	Measure	NSGA-II				MOEA/D			
		SPC		DE		SPC		DE	
		f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
*ZDT*1	MAX	0.9977	1.0000	0.9991	1.0028	1.0000	1.0058	0.9998	1.0023
	MIN	0	1.450E-3	0	1.699E-3	0	1.992E-4	0	2.826E-4
	AVG	0.4517	0.3971	0.5546	0.2916	0.4190	0.3973	0.4187	0.3975
	STD	0.3098	0.2909	0.2762	0.2215	0.2741	0.2353	0.2739	0.2351
	ERR	0.0022325		0.0021727		0.00030632		0.00029017	
	TMR	1.439s		1.372s		0.041s		0.042s	
*ZDT*2	MAX	0.9998	1.0000	0.9947	1.0000	1.0000	1.0000	1.0000	1.0000
	MIN	0	4.697E-3	0	1.086E-2	0	2.978E-4	0	5.051E-4
	AVG	0.6024	0.5726	0.1813	0.8825	0.5685	0.5823	0.5685	0.5823
	STD	0.2621	0.2983	0.2912	0.2280	0.3081	0.3302	0.3081	0.3301
	ERR	0.002427		3.3345E-5		0.00016512		0.00016875	
	TMR	1.392s		1.382s		0.0299s		0.0293s	
*ZDT*3	MAX	0.8517	1.0033	0.8514	1.0025	0.8519	1.0000	0.8531	1.0016
	MIN	0	-7.687E-1	0	-7.687E-1	0	-7.690E-1	0	-7.727E-1
	AVG	0.2121	0.4673	0.1411	0.5880	0.1667	0.5559	0.5197	-1.261E-1
	STD	0.2367	0.4541	0.1843	0.3646	0.2411	0.4098	0.2793	0.5003
	ERR	0.00062655		0.00066736		0.0035069		0.00042582	
	TMR	1.430s		1.364s		0.0296s		0.0289s	
*ZDT*4	MAX	0.9998	1.0013	0.9997	1.0003	1.0000	1.0000	1.0000	1.0000
	MIN	0	2.016E-3	0	2.179E-3	0	2.285E-7	0	0
	AVG	0.4803	0.3764	0.4719	0.3843	0.2299	0.7009	0.3611	0.5568
	STD	0.3100	0.2983	0.3197	0.3009	0.3796	0.3742	0.4293	0.4060
	ERR	0.0015544		0.0014316		0.00082932		0.0003195	
	TMR	1.377s		1.370s		0.664s		0.642s	

4 Sizing Optimization Results

The three circuits, depicted in Figures 1, 2 and 3, were optimized and the results are shown in this section. Each one was optimized with different number of variables, objectives, population size and number of generations.

Both optimization systems are performed with MATLAB and the circuit simulations are made with HSPICE by modifying each transistor width (W_i) and length

(L), and recollecting results from the output listing. Table 3 shows an example of the HSPICE directions to perform an AC and DC analysis and the measurements of gain, bandwidth and offset in voltage mode [33, 34].

Table 3 Measurements library example

```
.LIB MEASLIB                ⋆ Library name
 .AC dec 100 100 1G         ⋆ Executing an AC Analysis
 .TF V(X) VIN               ⋆ Calculating DC parameters
 .NET V(X) VIN              ⋆ Calculating AC parameters
 .MEAS AC GAIN MAX Vdb(X) FROM=100 TO=100    ⋆Calculating Gain in dB
 .MEAS AC BW TRIG vdb(X) at=100 TARG vdb(X) VAL='GAIN-3' cross=1    ⋆Calculating f−3dB

 .DC VIN 1.5 -1.5 .01              ⋆Executing a DC Analysis
 .MEAS DC OFFSET TRIG V(x) at=0 targ v(x) val=0 cross=1  ⋆Calculating offset
.ENDL MEASLIB
```

4.1 Voltage Follower (VF)

The VF depicted in Fig. 1 is encoded with seven design variables: transistors lengths (L) and widths (W_i), where i represents a specific transistor (or transistors which share the same width) of the circuit, as Table 4 shows.

Table 4 VF encoding

gene	Design Variable	Encoding Transistors
x_1	L	$M_1, \dots M_{14}$
x_2	W_1	$M_{11}, M_{12}, M_{13}, M_{14}$
x_3	W_2	M_8, M_9, M_{10}
x_4	W_3	M_6
x_5	W_4	M_5
x_6	W_5	M_1, M_3
x_7	W_6	M_2, M_4

The optimization problem for this circuit is expressed as:

$$\begin{aligned} &\text{minimize } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), f_4(\mathbf{x}), f_5(\mathbf{x})]^{\mathrm{T}} \\ &\text{subject to } \ h_k(\mathbf{x}) \geq 0\ , \ \ k = 1, \dots, 14, \\ &\text{where } \ \mathbf{x} \in X. \end{aligned} \tag{6}$$

where $X : \mathbb{R}^7 \mid 0.36\ \mu m \leq X_i \leq 80\ \mu m$, $i = 1, 2, \dots, 5$, is the decision space for the variables $\mathbf{x} = [x_1, \dots, x_7]^{\mathrm{T}}$. $\mathbf{f}(\mathbf{x})$ is the vector formed by five objectives:

- $f_1(\mathbf{x})$ = 1 - Voltage gain (From Y port to X port).
- $f_2(\mathbf{x})$ = Voltage offset (Between Y port and X port).
- $f_3(\mathbf{x})$ = 1 / Voltage band width (From Y port to X port).
- $f_4(\mathbf{x})$ = 1/ Input resistance (Y port).
- $f_5(\mathbf{x})$ = Output resistance (X port).

Finally, $h_k(\mathbf{x}),\ k = 1,\ldots,14$ are performance constraints, in our experiments we include the saturation condition in all transistors as constraints [24]. Then this circuit was optimized along 126 generations over 10 runs, $H = 5$ was used, and according to equation 3 the population size is 126. For DE, C=1 and R=0.5 were used.

Table 5 NSGA-II optimization measurements for VF

Measure	Gain ($\frac{V}{V}$)	Offset (V)	BW (Hz)	Rout (Ω)	Rin (Ω)
MAX	0.9910	2.739E-3	9.991E+8	3.1486	2.077E+5
MIN	0.9866	1.951E-6	4.837E+8	0.5203	1.011E+5
AVG	0.9887	9.702E-4	8.811E+8	0.8073	1.616E+5
STD	6.404E-4	5.272E-4	1.294E+8	0.2783	2.276E+4

Table 6 MOEA/D optimization measurements for VF

Measure	Gain ($\frac{V}{V}$)	Offset (V)	BW (Hz)	Rout (Ω)	Rin (Ω)
MAX	0.9914	4.145E-3	9.997E+8	18.3272	2.088E+5
MIN	0.9836	5.172E-7	3.004E+8	0.5030	6.183E+4
AVG	0.9893	1.469E-3	7.334E+8	2.6302	1.673E+5
STD	1.287E-3	1.136E-3	2.265E+8	3.4790	3.288E+4

4.2 *Positive Second Generation Current Conveyor (CCII$^+$)*

The CCII$^+$ depicted in Fig. 2 is encoded with nine design variables: transistors lengths (L) and widths (W_i), where i represents a specific transistor (or transistors which share the same width) of the circuit, as Table 9 shows.

The optimization problem for this circuit is expressed as:

$$\begin{aligned} &\text{minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{10}(\mathbf{x}))^T \\ &\text{subject to } \ h_k(\mathbf{x}) \geq 0\ ,\ \ k = 1,\ldots,15, \\ &\text{where } \ \mathbf{x} \in X. \end{aligned} \tag{7}$$

where $X : \mathbb{R}^9 \mid 0.36\ \mu m \leq X_i \leq 80\ \mu m$, $i = 1,2,\ldots,9$, is the decision space for the variables $\mathbf{x} = (x_1,\ldots,x_9)$. $\mathbf{f}(\mathbf{x})$ is the vector formed by ten objectives:

- $f_1(\mathbf{x})$ = 1 - Voltage gain (From Y port to X port).
- $f_2(\mathbf{x})$ = Voltage offset (Between Y port and X port).

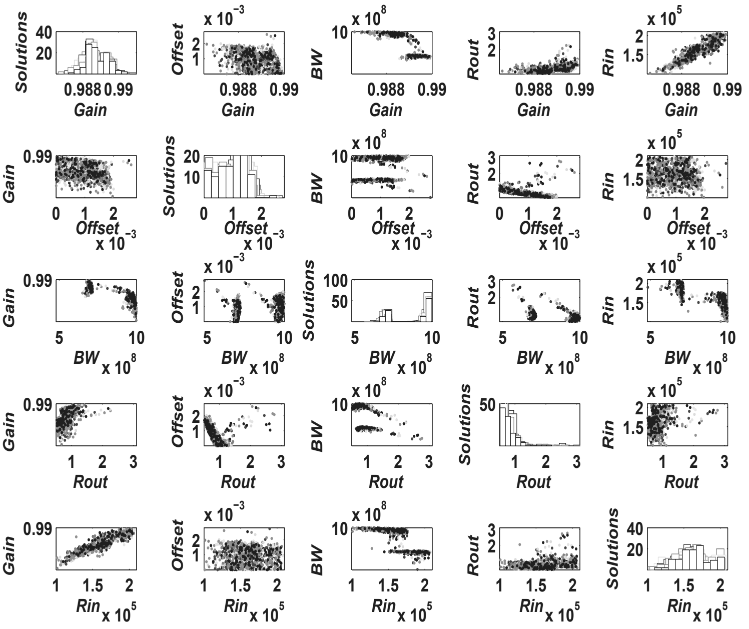


Fig. 8 NSGA-II optimization for VF

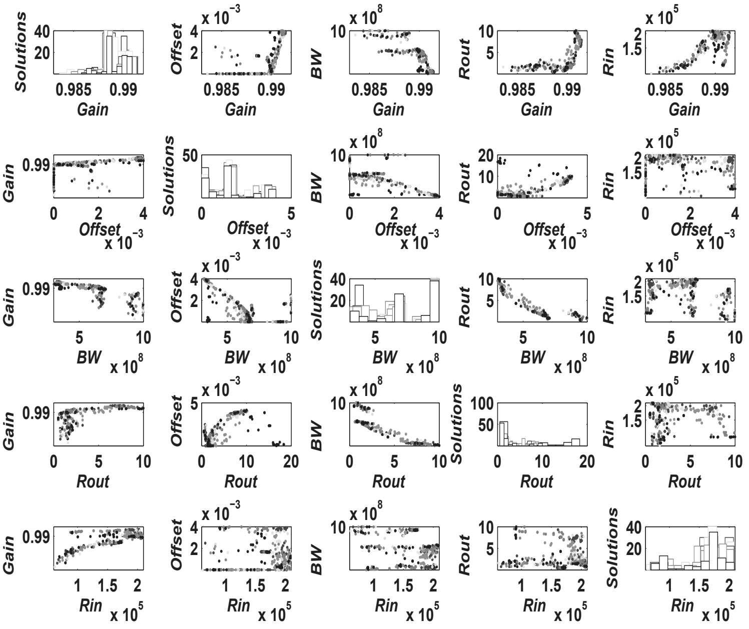


Fig. 9 MOEA/D optimization for VF

Table 7 NSGA-II best objective results for VF

	Best Objective for:				
	Gain	Offset	BW	Rout	Rin
Gain	**0.991**	0.987	0.987	0.989	0.990
Offset	2.74E-3	**1.95E-6**	1.20E-3	1.69E-3	9.41E-4
BW	4.84E+8	9.54E+8	**9.99E+8**	9.48E+8	6.99E+8
Rout	3.149	1.114	0.750	**0.520**	0.835
Rin	1.90E+5	1.01E+5	1.16E+5	1.72E+5	**2.08E+5**
Variable Values for the Best Values:					
L	7.20E-7	5.40E-7	5.40E-7	5.40E-7	7.20E-7
W_1	2.11E-6	1.31E-5	4.86E-6	7.93E-7	1.03E-6
W_2	5.94E-6	5.41E-6	1.16E-5	1.68E-6	3.58E-6
W_3	3.15E-6	2.53E-5	4.93E-5	7.97E-5	4.77E-5
W_4	7.76E-5	6.98E-5	7.84E-5	7.89E-5	7.17E-5
W_5	7.72E-5	6.54E-5	7.41E-5	7.35E-5	7.96E-5
W_6	6.25E-5	6.09E-5	2.87E-5	5.68E-5	6.93E-5

Table 8 MOEA/D best objective results for VF

	Best Objective for:				
	Gain	Offset	BW	Rout	Rin
Gain	**0.991**	0.988	0.985	0.989	0.990
Offset	3.91E-3	**5.17E-7**	2.38E-3	1.65E-3	1.37E-3
BW	3.25E+8	9.38E+8	**1.00E+9**	9.45E+8	6.92E+8
Rout	7.798	1.236	0.985	**0.503**	0.738
Rin	2.03E+5	1.29E+5	8.29E+4	1.71E+5	**2.09E+5**
Variable Values for the Best Values:					
L	7.20E-7	5.40E-7	5.40E-7	5.40E-7	7.20E-7
W_1	1.23E-6	2.80E-6	3.52E-5	1.19E-6	1.06E-6
W_2	4.61E-6	1.27E-5	3.96E-6	1.89E-6	2.88E-6
W_3	7.29E-7	2.07E-5	5.41E-5	8.00E-5	6.31E-5
W_4	7.23E-5	5.94E-5	4.77E-5	8.00E-5	7.00E-5
W_5	7.25E-5	7.64E-5	7.13E-5	7.98E-5	7.97E-5
W_6	6.20E-5	5.73E-5	4.41E-5	7.69E-5	7.73E-5

- $f_3(\mathbf{x})$ = 1 / Voltage band width (From Y port to X port).
- $f_4(\mathbf{x})$ = 1/ Input resistance (Y port).
- $f_5(\mathbf{x})$ = Output resistance (X port).
- $f_6(\mathbf{x})$ = 1 - Current gain (From X port to Z port).
- $f_7(\mathbf{x})$ = Current offset (Between X port and Z port).
- $f_8(\mathbf{x})$ = 1 / Current band width (From X port to Z port).
- $f_9(\mathbf{x})$ = Input resistance (X port).
- $f_{10}(\mathbf{x})$ = 1/Output resistance (Z port).

Table 9 CCII$^+$ encoding

gene	Design Variable	Encoding Transistors
x_1	L	$M_1, \ldots M_{14}$
x_2	W_1	$M_{11}, M_{12}, M_{13}, M_{14}$
x_3	W_2	M_8, M_9, M_{10}
x_4	W_3	M_6
x_5	W_4	M_5
x_6	W_5	M_1, M_3
x_7	W_6	M_2, M_4
x_8	W_7	M_{15}
x_9	W_8	M_7

Finally, $h_k(\mathbf{x}),\ k = 1,\ldots,15$ are performance constraints, in our experiments we include the saturation condition in all transistors as constraints [24]. Then this circuit was optimized along 111 generations over 10 runs, with a population size of 111. For DE, C=1 and R=0.5 were used.

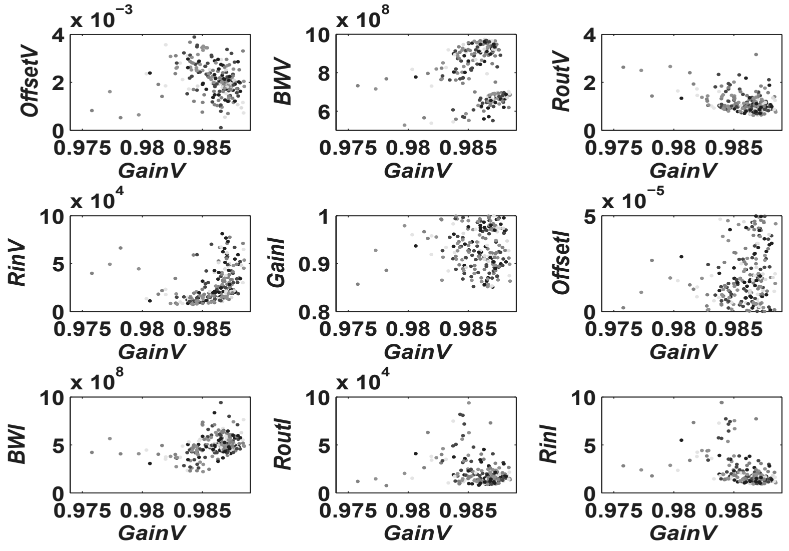


Fig. 10 NSGA-II voltage gain (GainV) optimization for CCII$^+$

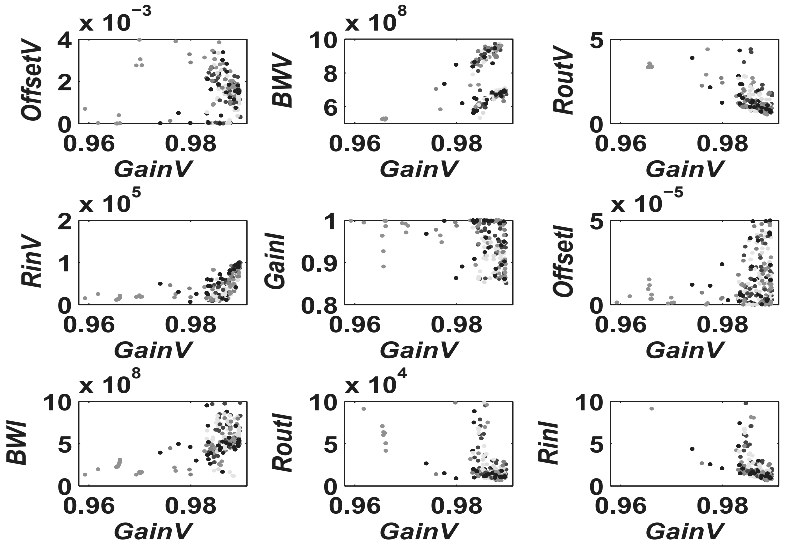


Fig. 11 MOEA/D voltage gain (GainV) optimization for CCII^{+}

Table 10 NSGA-II optimization measurements for CCII^{+}

	Measure	GainV ($\frac{V}{V}$)	OffsetV (V)	BWV (Hz)	RoutV (Ω)	RinV (Ω)
VOLTAGE (Y-X)	MAX	0.9885	4.836E-3	9.733E+8	3.1576	8.115E+4
	MIN	0.9758	1.105E-4	5.268E+8	0.6239	7.393E+3
	AVG	0.9858	2.159E-3	7.913E+8	1.1510	2.783E+4
	STD	1.947E-3	7.562E-4	1.308E+8	0.4134	1.788E+4
CURRENT (X-Z)	Measure	GainI ($\frac{I}{I}$)	OffsetI (A)	BWI (Hz)	RoutI (Ω)	RinI (Ω)
	MAX	0.9999	4.988E-5	9.425E+8	1.456E+5	13.8628
	MIN	0.8513	1.972E-8	2.218E+8	7.758E+3	0.8994
	AVG	0.9302	1.878E-5	4.917E+8	2.504E+4	2.5182
	STD	4.140E-2	1.402E-5	1.130E+8	2.406E+4	2.4096

4.3 Current Feedback Operational Amplifier (CFOA)

The CFOA depicted in Fig. 3 is encoded with fifteen design variables: transistors lengths (L) and widths (W_i), where i represents a specific transistor (or transistors which share the same width) of the circuit, as Table 14 shows.

The optimization problem for this circuit is expressed as:

Table 11 MOEA/D optimization measurements for CCII$^+$

	Measure	GainV ($\frac{V}{V}$)	OffsetV (V)	BWV (Hz)	RoutV (Ω)	RinV (Ω)
VOLTAGE (Y-X)	MAX	0.9897	6.129E-3	9.734E+8	20.8758	1.011E+5
	MIN	0.9592	7.773E-7	2.650E+8	0.5303	6.552E+3
	AVG	0.9856	1.548E-3	7.268E+8	1.7078	5.037E+4
	STD	5.076E-3	1.108E-3	1.551E+8	2.1357	2.873E+4
CURRENT (X-Z)	Measure	GainI ($\frac{I}{I}$)	OffsetI (A)	BWI (Hz)	RoutI (Ω)	RinI (Ω)
	MAX	1.0000	4.994E-5	9.832E+8	5.142E+5	65.3312
	MIN	0.8514	6.135e-010	1.175E+8	6.641E+3	0.6405
	AVG	0.9605	1.421E-5	5.175E+8	3.251E+4	4.2372
	STD	4.911E-2	1.392E-5	1.680E+8	5.645E+4	8.3924

Table 12 NSGA-II best objective results for CCII$^+$

	Best Objective for:									
	GainV	OffsetV	BWV	RoutV	RinV	GainI	OffsetI	BWI	RoutI	RinI
GainV	**0.988**	0.987	0.986	0.987	0.987	0.987	0.988	0.987	0.986	0.987
OffsetV	1.83E-3	**1.11E-4**	1.92E-3	2.01E-3	2.29E-3	2.09E-3	2.27E-3	1.11E-4	4.41E-3	1.98E-3
BWV	6.86E+8	9.54E+8	**9.73E+8**	9.35E+8	6.16E+8	6.37E+8	9.19E+8	9.54E+8	7.00E+8	9.65E+8
RoutV	0.754	1.232	0.680	**0.624**	1.084	1.012	0.682	1.232	1.609	0.627
RinV	5.38E+4	1.94E+4	1.38E+4	1.50E+4	**8.12E+4**	4.73E+4	4.15E+4	1.94E+4	1.54E+4	2.13E+4
GainI	0.864	0.863	0.916	0.868	0.871	**1.000**	0.895	0.863	0.888	0.915
OffsetI	4.58E-5	2.81E-5	9.36E-6	4.32E-6	3.36E-6	2.67E-6	**1.97E-8**	2.81E-5	4.34E-6	3.57E-5
BWI	4.87E+8	9.43E+8	5.01E+8	5.05E+8	4.52E+8	4.11E+8	5.27E+8	**9.43E+8**	2.46E+8	4.81E+8
RoutI	1.38E+4	1.98E+4	9.83E+3	8.98E+3	1.89E+4	1.80E+4	1.00E+4	1.98E+4	**1.46E+5**	9.19E+3
RinI	1.047	1.772	0.995	0.948	1.682	1.545	0.972	1.772	13.555	**0.899**
	Variable Values for the Best Values:									
L	7.20E-7	5.40E-7	5.40E-7	5.40E-7	7.20E-7	7.20E-7	5.40E-7	5.40E-7	5.40E-7	5.40E-7
W_1	1.88E-6	3.78E-6	5.69E-6	5.52E-6	8.33E-7	1.93E-6	1.83E-6	3.78E-6	4.26E-6	3.60E-6
W_2	4.85E-5	4.68E-5	5.44E-5	5.26E-5	5.30E-5	5.21E-5	3.95E-5	4.68E-5	4.36E-5	4.45E-5
W_3	7.85E-5	2.33E-5	7.01E-5	7.97E-5	7.53E-5	6.97E-5	7.89E-5	2.33E-5	7.08E-5	7.60E-5
W_4	7.59E-5	7.72E-5	8.00E-5	7.41E-5	7.33E-5	7.31E-5	5.79E-5	7.72E-5	2.84E-5	7.94E-5
W_5	7.32E-5	7.91E-5	7.46E-5	6.84E-5	5.08E-5	5.86E-5	6.36E-5	7.91E-5	7.94E-5	7.43E-5
W_6	3.44E-5	1.64E-5	2.84E-5	7.72E-5	8.58E-6	1.02E-5	5.65E-5	1.64E-5	3.33E-6	2.96E-5
W_7	6.63E-5	6.64E-5	7.77E-5	6.92E-5	6.55E-5	7.51E-5	5.51E-5	6.64E-5	2.80E-5	7.63E-5
W_8	7.44E-5	2.19E-5	7.25E-5	7.85E-5	7.04E-5	7.52E-5	7.93E-5	2.19E-5	6.66E-5	7.92E-5

$$\begin{aligned}
&\text{minimize } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{12}(\mathbf{x})]^{\mathrm{T}} \\
&\text{subject to } \; h_k(\mathbf{x}) \geq 0\;, \;\; k = 1 \ldots p, \\
&\text{where } \; \mathbf{x} \in X.
\end{aligned} \tag{8}$$

where $X : \mathbb{R}^15 \mid 0.36\ \mu m \leq X_i \leq 80\ \mu m$, $i = 1, 2, \ldots, 15$, is the decision space for the variables $\mathbf{x} = [x_1, \ldots, x_{15}]^{\mathrm{T}}$. $\mathbf{f}(\mathbf{x})$ is the vector formed by twelve objectives:

- $f_1(\mathbf{x})$ = 1 - Voltage gain (From Y port to X port).
- $f_2(\mathbf{x})$ = Voltage offset (Between Y port and X port).

Table 13 MOEA/D best objective results for CCII$^+$

	Best Objective for:									
	GainV	OffsetV	BWV	RoutV	RinV	GainI	OffsetI	BWI	RoutI	RinI
GainV	**0.990**	0.986	0.988	0.989	0.990	0.985	0.986	0.990	0.986	0.989
OffsetV	3.09E-4	**7.77E-7**	1.46E-3	1.74E-3	1.13E-3	3.05E-3	2.85E-3	3.36E-4	5.83E-3	1.74E-3
BWV	6.74E+8	6.49E+8	**9.73E+8**	9.61E+8	6.90E+8	8.43E+8	8.94E+8	6.74E+8	3.71E+8	9.61E+8
RoutV	1.678	2.072	0.737	**0.530**	1.166	1.069	1.043	1.813	20.876	0.530
RinV	9.98E+4	1.94E+4	7.66E+4	8.29E+4	**1.01E+5**	1.73E+4	2.09E+4	9.87E+4	2.09E+4	8.29E+4
GainI	0.851	0.908	1.000	0.867	0.947	**1.000**	0.958	0.965	0.935	0.867
OffsetI	8.43E-6	3.08E-5	2.40E-5	4.50E-7	1.86E-5	6.87E-6	**6.14e-010**	2.99E-6	1.35E-7	4.50E-7
BWI	8.30E+8	7.40E+8	5.83E+8	5.27E+8	6.90E+8	4.46E+8	5.18E+8	**9.83E+8**	1.68E+8	5.27E+8
RoutI	1.95E+4	3.78E+4	8.77E+3	6.64E+3	1.45E+4	1.92E+4	1.40E+4	2.56E+4	**5.14E+5**	6.64E+3
RinI	1.862	3.589	0.947	0.640	1.390	2.041	1.554	2.017	65.331	**0.640**
	Variable Values for the Best Values:									
L	7.20E-7	7.20E-7	5.40E-7	5.40E-7	7.20E-7	5.40E-7	5.40E-7	7.20E-7	5.40E-7	5.40E-7
W_1	7.28E-7	5.00E-6	6.80E-7	6.96E-7	7.25E-7	3.82E-6	3.67E-6	7.28E-7	2.98E-6	6.96E-7
W_2	6.88E-6	7.75E-5	1.41E-5	4.74E-6	6.92E-6	3.31E-5	7.20E-5	6.88E-6	7.18E-5	4.74E-6
W_3	1.95E-5	1.81E-5	5.30E-5	7.86E-5	3.73E-5	7.19E-5	6.36E-5	1.82E-5	1.75E-5	7.86E-5
W_4	4.67E-5	7.49E-5	6.74E-5	7.87E-5	5.14E-5	5.10E-5	3.70E-5	4.41E-5	1.03E-6	7.87E-5
W_5	7.42E-5	7.02E-5	7.65E-5	7.99E-5	7.95E-5	4.24E-5	5.88E-5	7.38E-5	6.00E-5	7.99E-5
W_6	4.78E-5	9.08E-6	3.45E-5	4.82E-5	5.60E-5	6.66E-6	3.57E-5	3.88E-5	2.93E-6	4.82E-5
W_7	5.60E-5	6.33E-5	7.25E-5	7.69E-5	5.75E-5	5.32E-5	4.01E-5	4.36E-5	9.25E-6	7.69E-5
W_8	2.37E-5	1.78E-5	6.03E-5	7.94E-5	4.21E-5	7.81E-5	7.30E-5	1.85E-5	1.56E-5	7.94E-5

Table 14 CFOA encoding

gene	Design Variable	Encoding Transistors
x_1	L	$M_1, \dots M_{25}$
x_2	W_1	$M_{11}, M_{12}, M_{13}, M_{14}$
x_3	W_2	M_8, M_9, M_{10}
x_4	W_3	M_6
x_5	W_4	M_5
x_6	W_5	M_1, M_3
x_7	W_6	M_2, M_4
x_8	W_7	M_{15}
x_9	W_8	M_7
x_{10}	W_9	M_{23}, M_{24}
x_{11}	W_{10}	M_{16} ,M_{17}
x_{12}	W_{11}	M_{21} ,M_{22}
x_{13}	W_{12}	M_{19} ,M_{20}
x_{14}	W_{13}	M_{18}
x_{15}	W_{14}	M_{25}

- $f_3(\mathbf{x})$ = 1 / Voltage band width (From Y port to X port).
- $f_4(\mathbf{x})$ = Output resistance (X port).
- $f_5(\mathbf{x})$ = 1 - Current gain (From X port to Z port).
- $f_6(\mathbf{x})$ = Current offset (Between X port and Z port).

- $f_7(\mathbf{x})$ = 1 / Current band width (From X port to Z port).
- $f_8(\mathbf{x})$ = 1/ Output resistance (Z port).
- $f_9(\mathbf{x})$ = 1 - Voltage gain (From Z port to W port).
- $f_{10}(\mathbf{x})$ = Voltage offset (Between Z port and W port).
- $f_{11}(\mathbf{x})$ = 1 / Voltage band width (From Z port to W port).
- $f_{12}(\mathbf{x})$ = Output resistance (W port).

Finally, $h_k(\mathbf{x}),\ k = 1 \ldots p$ are performance constraints, in our experiments we include the saturation condition in all transistors as constraints [24]. Then this circuit was optimized along 168 generations over 10 runs, and the population size of 168. For DE, C=1 and R=0.5 were selected.

Table 15 NSGA-II optimization measurements for CFOA

	Measure	GainV ($\frac{V}{V}$)	OffsetV (V)	BWV (Hz)	RoutV (Ω)
VOLTAGE (Y-X)	MAX	0.9880	6.098E-3	6.303E+8	7.7824
	MIN	0.9707	4.868E-4	3.190E+8	0.7915
	AVG	0.9853	2.495E-3	4.871E+8	1.5373
	STD	2.002E-3	8.009E-4	5.247E+7	0.7503
CURRENT (X-Z)	Measure	GainI ($\frac{I}{I}$)	OffsetI (A)	BWI (Hz)	RoutI (Ω)
	MAX	0.9986	4.976E-5	9.996E+8	8.646E+4
	MIN	0.6017	2.111E-7	1.248E+8	2.760E+3
	AVG	0.8513	1.851E-5	8.087E+8	1.581E+4
	STD	0.1001	1.350E-5	1.522E+8	9.532E+3
VOLTAGE (Z-W)	Measure	GainW ($\frac{V}{V}$)	OffsetW (V)	BWW (Hz)	RoutW (Ω)
	MAX	0.9890	7.348E-3	9.893E+8	21.2337
	MIN	0.9754	7.205E-6	3.219E+8	0.8430
	AVG	0.9841	1.744E-3	7.084E+8	3.2157
	STD	2.249E-3	1.312E-3	1.186E+8	2.5300

4.4 Discussion of Results

Before discussing the results, it is important to remember that the variable values of the three problems, represent the physical dimensions of each codified transistor. Smaller values for the dimensions L and W, are required in integrated circuit design to reduce the silicon area.

For the VF, although Tables 5 and 6 show similar results, actually, MOEA/D presents the best performance for all the objectives, but only for offset, MOEA/D significantly improves the NSGA-II offset result. While MOEA/D has better results, it is possible to see how the average results for offset, band width and output

Table 16 MOEA/D optimization measurements for CFOA

	Measure	GainV ($\frac{V}{V}$)	OffsetV (V)	BWV (Hz)	RoutV (Ω)
VOLTAGE (Y-X)	MAX	0.9893	6.984E-3	6.641E+8	50.2381
	MIN	0.9344	1.211E-5	2.268E+8	0.6119
	AVG	0.9851	2.236E-3	4.595E+8	1.8835
	STD	5.695E-3	1.167E-3	6.571E+7	2.9437
CURRENT (X-Z)	Measure	GainI ($\frac{I}{I}$)	OffsetI (A)	BWI (Hz)	RoutI (Ω)
	MAX	1.0000	4.920E-5	9.999E+8	2.809E+5
	MIN	0.6022	2.168E-9	1.011E+8	5.708E+3
	AVG	0.8997	1.135E-5	7.743E+8	2.535E+4
	STD	0.1044	1.061E-5	2.242E+8	2.892E+4
VOLTAGE (Z-W)	Measure	GainW ($\frac{V}{V}$)	OffsetW (V)	BWW (Hz)	RoutW (Ω)
	MAX	0.9899	7.899E-3	9.994E+8	65.7684
	MIN	0.9275	7.635E-7	2.730E+8	0.4185
	AVG	0.9851	1.573E-3	6.507E+8	3.4540
	STD	7.186E-3	1.181E-3	1.659E+8	6.6713

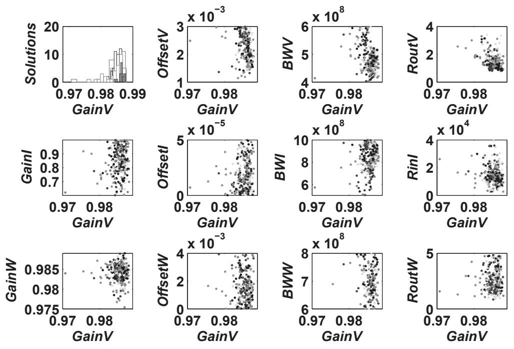


Fig. 12 NSGA-II voltage gain (GainV) optimization for CFOA

resistance, are better in NSGA-II than MOEA/D. Figure 9 and the standard deviation of MOEA/D output resistance, shows an asymmetric behavior in the solution set.

Tables 7 and 8 list the variable values for each best objective. Sometimes, the W_i's values are similar, but the L values are the same, which indicates that both methods found similar Pareto fronts to achieve their best performances.

For the CCII$^+$, the behavior continues in both algorithms: MOEA/D exhibits the best results for the offset (in voltage and current mode); the improvement is most remarkable for the voltage input resistance (Tables 10 and 16). Now the average is closer for both methods although the standard deviation is better in NSGA-II

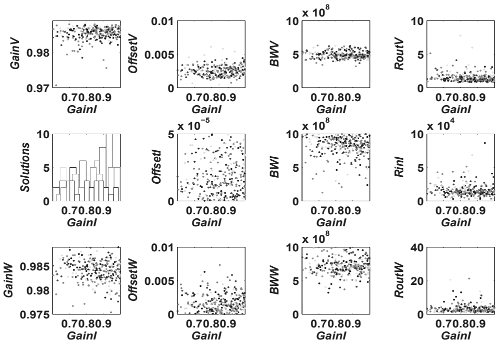


Fig. 13 NSGA-II current gain (GainI) optimization for CFOA

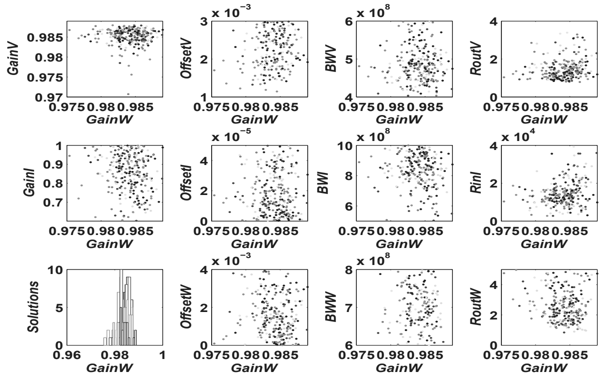


Fig. 14 NSGA-II voltage gain (GainW) optimization for CFOA

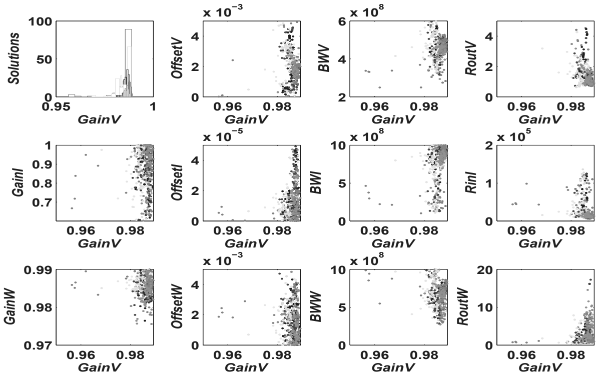


Fig. 15 MOEA/D voltage gain (GainV) optimization for CFOA

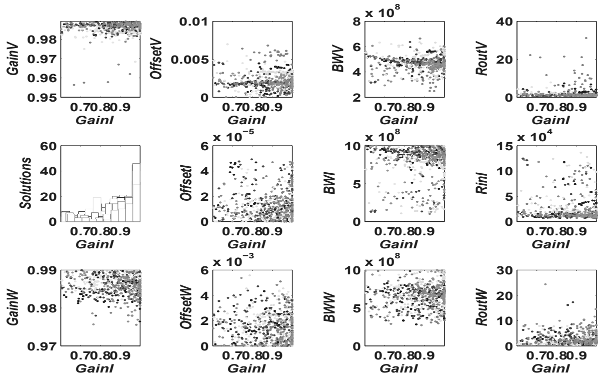


Fig. 16 MOEA/D current gain (GainI) optimization for CFOA

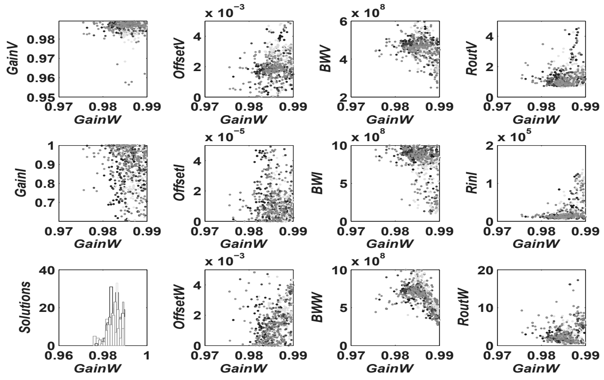


Fig. 17 MOEA/D voltage gain (GainW) optimization for CFOA

than MOEA/D. Tables 12 and 13 have small differences in the variables, although for offset values, *L* is different. In this point, if we compare Tables 12 and 7, the variable values are similar in *L*, however if we compare Tables 13 and 8, MOEA/D tried to change the L value to handle more objectives achieving the best results.

Finally, for the CFOA, the behavior is similar as for the CCII$^+$, MOEA/D has the best performance (in optimal and average objective values) but there is a large improvement only for the offset in voltage and current. Regarding to the standard deviation, MOEA/D improves its values but for output resistances presents an asymmetric behavior. This time, the variables for both algorithms changed to handle the large number of objectives and variables. For the first time, both methods found smaller values of *L* (MOEA/D for BWV, ROUTV and ROUTW, NSGA-II for OFFSETI).

If we focus on the behavior of the EAs, on the one hand, MOEA/D shows better performance because besides finding best objectives values than NSGA-II, its

Table 17 NSGA-II best objective results for CFOA

	Best Objective for:											
	GainV	OffsetV	BWV	RoutV	GainI	OffsetI	BWI	RoutI	GainW	OffsetW	BWW	RoutW
GainV	**0.988**	0.984	0.984	0.987	0.987	0.981	0.985	0.983	0.987	0.980	0.985	0.985
OffsetV	2.13E-3	**4.87E-4**	3.50E-3	1.67E-3	1.99E-3	3.93E-3	2.96E-3	2.91E-3	1.92E-3	2.81E-3	3.41E-3	3.41E-3
BWV	4.51E+8	5.91E+8	**6.30E+8**	4.77E+8	4.81E+8	5.94E+8	5.91E+8	4.97E+8	4.75E+8	4.97E+8	6.25E+8	6.25E+8
RoutV	1.862	3.393	2.368	**0.791**	1.144	1.855	1.189	3.134	1.190	1.920	1.205	1.205
GainI	0.698	0.847	0.692	0.787	**0.999**	0.761	0.978	0.943	0.988	0.729	0.884	0.884
OffsetI	2.40E-5	1.40E-6	1.34E-5	4.80E-5	1.02E-5	**2.11E-7**	1.54E-5	3.50E-6	3.07E-5	2.38E-5	3.61E-5	3.61E-5
BWI	8.53E+8	8.42E+8	8.10E+8	9.15E+8	8.61E+8	9.56E+8	**1.00E+9**	5.36E+8	8.74E+8	4.43E+8	9.76E+8	9.76E+8
RoutI	9.40E+3	4.48E+4	4.03E+4	7.75E+3	1.51E+4	2.99E+4	1.56E+4	**8.65E+4**	9.88E+3	8.76E+3	1.31E+4	1.31E+4
GainW	0.985	0.985	0.986	0.986	0.985	0.985	0.983	0.987	**0.989**	0.985	0.984	0.984
OffsetW	1.79E-3	4.83E-4	1.14E-3	9.11E-4	1.69E-3	6.32E-3	3.88E-3	7.98E-4	3.07E-3	**7.20E-6**	1.81E-3	1.81E-3
BWW	7.42E+8	9.39E+8	8.63E+8	5.56E+8	6.51E+8	3.84E+8	9.21E+8	4.91E+8	4.82E+8	9.60E+8	**9.89E+8**	9.89E+8
RoutW	1.180	3.078	2.196	6.213	3.563	8.694	2.573	1.730	4.726	2.134	0.843	**0.843**
Variable Values for the Best Values:												
L	7.20E-7	5.40E-7	5.40E-7	7.20E-7	7.20E-7	3.60E-7	5.40E-7	5.40E-7	7.20E-7	5.40E-7	5.40E-7	5.40E-7
W_1	3.39E-6	4.14E-6	7.11E-6	7.54E-6	5.03E-6	8.29E-6	7.69E-6	1.04E-5	7.78E-6	7.26E-6	7.90E-6	7.90E-6
W_2	4.94E-5	5.43E-5	7.15E-5	5.88E-5	5.96E-5	6.68E-5	7.64E-5	6.34E-5	5.20E-5	7.60E-5	7.15E-5	7.15E-5
W_3	3.66E-5	1.01E-5	3.50E-5	7.52E-5	5.67E-5	3.27E-5	5.48E-5	2.24E-5	5.71E-5	4.43E-5	6.11E-5	6.11E-5
W_4	2.72E-5	4.23E-5	2.47E-5	7.91E-5	5.51E-5	2.33E-5	4.54E-5	2.42E-5	4.91E-5	4.92E-5	3.83E-5	3.83E-5
W_5	6.28E-5	7.98E-5	5.54E-5	6.22E-5	6.60E-5	4.77E-5	7.51E-5	5.48E-5	5.11E-5	2.76E-5	7.94E-5	7.94E-5
W_6	4.92E-5	4.16E-6	5.78E-6	6.01E-5	2.51E-5	6.27E-6	1.10E-5	5.58E-6	5.99E-5	4.14E-6	1.11E-5	1.11E-5
W_7	4.73E-5	8.21E-6	2.52E-5	5.86E-5	5.64E-5	4.28E-5	5.39E-5	2.55E-5	5.79E-5	5.87E-5	5.49E-5	5.49E-5
W_8	4.19E-5	4.05E-5	1.79E-5	7.23E-5	5.92E-5	3.05E-5	4.80E-5	2.42E-5	5.60E-5	6.37E-5	3.65E-5	3.65E-5
W_9	1.24E-5	6.62E-6	8.50E-6	4.23E-5	1.31E-5	7.81E-7	1.82E-5	7.81E-7	3.78E-5	1.24E-5	1.62E-5	1.62E-5
W_{10}	1.32E-5	6.80E-5	6.00E-5	2.99E-5	3.37E-5	1.07E-5	7.23E-5	1.52E-5	6.39E-5	7.35E-5	4.57E-5	4.57E-5
W_{11}	7.97E-5	5.20E-5	5.30E-5	6.42E-5	5.81E-5	2.72E-5	6.44E-5	2.44E-5	7.88E-5	7.18E-5	6.07E-5	6.07E-5
W_{12}	5.28E-6	1.59E-5	5.84E-5	1.59E-5	4.97E-5	4.93E-6	1.89E-5	6.04E-6	7.24E-5	3.11E-5	7.09E-5	7.09E-5
W_{13}	6.62E-5	4.00E-5	5.12E-5	3.22E-5	1.93E-5	4.23E-5	1.62E-5	6.92E-5	6.46E-5	4.47E-5	6.42E-5	6.42E-5
W_{14}	7.55E-5	1.04E-5	1.14E-5	4.90E-6	2.30E-5	4.59E-7	4.22E-5	1.16E-5	2.08E-6	1.60E-5	7.19E-5	7.19E-5

Table 18 MOEA/D best objective results for CFOA

	Best Objective for:											
	GainV	OffsetV	BWV	RoutV	GainI	OffsetI	BWI	RoutI	GainW	OffsetW	BWW	RoutW
GainV	**0.989**	0.956	0.982	0.983	0.987	0.983	0.989	0.985	0.988	0.987	0.984	0.983
OffsetV	2.00E-3	**1.21E-5**	6.81E-3	3.24E-3	1.75E-3	3.01E-3	1.28E-3	3.97E-3	1.49E-3	1.63E-3	2.95E-3	3.38E-3
BWV	4.97E+8	3.37E+8	**6.64E+8**	5.06E+8	4.52E+8	5.66E+8	5.08E+8	4.27E+8	4.49E+8	4.89E+8	5.33E+8	5.17E+8
RoutV	0.901	22.216	4.412	**0.612**	1.020	0.953	1.628	3.738	1.192	1.210	0.987	0.841
GainI	0.927	0.667	0.877	0.955	**1.000**	0.965	0.966	0.635	0.706	0.970	0.963	0.843
OffsetI	1.51E-5	5.34E-6	6.33E-6	1.04E-5	1.07E-5	**2.17E-9**	1.98E-5	6.08E-8	2.35E-6	1.84E-5	1.44E-5	8.90E-6
BWI	9.55E+8	4.63E+8	6.34E+8	9.98E+8	8.64E+8	1.00E+9	**1.00E+9**	1.71E+8	8.39E+8	9.16E+8	7.43E+8	7.15E+8
RoutI	1.27E+4	4.40E+4	4.17E+4	1.33E+4	1.18E+4	1.17E+4	2.44E+4	**2.81E+5**	1.20E+4	1.50E+4	1.64E+4	2.51E+4
GainW	0.987	0.986	0.985	0.947	0.989	0.984	0.989	0.988	**0.990**	0.986	0.982	0.986
OffsetW	1.09E-3	1.87E-3	2.77E-3	1.07E-3	3.61E-3	6.32E-4	1.86E-3	2.47E-3	3.98E-3	**7.63E-7**	1.42E-3	2.53E-3
BWW	8.10E+8	9.86E+8	9.76E+8	8.14E+8	3.35E+8	9.23E+8	5.45E+8	3.21E+8	3.36E+8	6.76E+8	**9.99E+8**	7.69E+8
RoutW	0.978	0.780	0.473	14.968	5.670	3.193	0.648	3.438	5.413	3.020	0.961	**0.418**
Variable Values for the Best Values:												
L	7.20E-7	5.40E-7	3.60E-7	3.60E-7	7.20E-7	5.40E-7	7.20E-7	7.20E-7	7.20E-7	7.20E-7	5.40E-7	3.60E-7
W_1	9.98E-7	1.87E-6	2.83E-6	3.66E-6	1.31E-5	9.96E-6	3.22E-6	8.09E-6	6.02E-6	1.01E-5	7.44E-6	1.28E-6
W_2	1.25E-5	7.20E-5	6.36E-5	7.55E-5	7.76E-5	7.68E-5	4.72E-5	7.23E-5	5.30E-5	7.66E-5	7.36E-5	4.68E-5
W_3	6.77E-5	6.89E-5	6.10E-5	7.86E-5	5.84E-5	7.96E-5	2.99E-5	3.64E-5	4.54E-5	4.96E-5	6.86E-5	6.30E-5
W_4	5.14E-5	2.89E-6	2.75E-6	8.00E-5	6.30E-5	5.92E-5	3.90E-5	2.03E-5	5.65E-5	5.59E-5	6.53E-5	6.27E-5
W_5	7.73E-5	2.85E-6	2.38E-5	7.30E-5	7.37E-5	4.18E-5	7.71E-5	6.06E-5	7.05E-5	6.18E-5	6.49E-5	5.68E-5
W_6	5.78E-5	4.54E-5	1.31E-5	4.77E-6	6.96E-5	1.26E-5	6.21E-5	3.79E-6	4.04E-5	5.33E-5	6.97E-6	2.45E-6
W_7	6.29E-5	4.96E-5	6.00E-5	7.90E-5	5.84E-5	7.80E-5	2.91E-5	2.50E-5	5.86E-5	4.86E-5	7.27E-5	7.66E-5
W_8	4.96E-5	2.38E-6	2.41E-6	7.99E-5	6.97E-5	5.92E-5	3.77E-5	1.32E-5	7.76E-5	5.56E-5	7.07E-5	7.71E-5
W_9	2.99E-6	2.76E-6	9.13E-7	5.59E-6	4.36E-5	2.01E-5	1.32E-6	1.13E-6	1.78E-5	1.65E-5	1.51E-5	3.65E-7
W_{10}	1.94E-5	7.72E-5	3.45E-5	7.63E-5	1.90E-5	6.34E-5	2.03E-5	1.58E-5	3.18E-5	7.04E-5	7.65E-5	1.16E-5
W_{11}	7.03E-5	6.35E-5	7.37E-5	6.58E-6	5.29E-5	6.37E-5	7.83E-5	6.09E-5	6.32E-5	7.20E-5	3.59E-5	7.81E-5
W_{12}	7.94E-5	1.57E-5	9.23E-6	5.99E-7	2.23E-5	1.57E-5	4.68E-5	3.97E-6	6.79E-5	1.80E-5	5.53E-5	1.01E-5
W_{13}	6.96E-5	7.77E-5	7.95E-5	6.51E-5	6.28E-5	3.05E-5	7.52E-5	7.54E-5	7.51E-5	4.43E-5	6.80E-5	7.90E-5
W_{14}	6.76E-5	6.87E-5	7.66E-5	6.84E-6	1.14E-6	1.39E-5	7.63E-5	3.80E-6	8.21E-7	1.40E-5	7.19E-5	7.98E-5

diversity feature tries to explore the whole search space, finding wider range of values in the objectives values compared with NSGA-II. If we compare the MOEA/D average objective values with theirs standard deviations, it is possible to see how a large number of the solutions are concentrated around the average and a few are exploring promising areas. Also, as we can see, it takes less execution time than NSGA-II to conclude the optimization task. On the other hand, NSGA-II has more symmetry in the solutions set, it avoids exploring areas that are far from the search space and increasing the probability of finding a solution without the need to make a large number of runs.

All these observations, confirm the difficulty for the development of a generic framework for analog circuit optimization, so that an analog designer should test an integrated circuit design with all available optimization tools to select the best optimal solution. This is a very difficult task. Other authors have suggested fuzzy-sets to automate the selection process [39],[40].

5 Conclusion

This work shows the usefulness of EA's in electronic design automation by using two algorithms named as NSGA-II and MOEA/D, which have the capability to handle a multi-objective optimization problem, with two or more objectives and taking into count also constraints.

First, we discussed about the selection of DE as genetic operator by using four synthetic ZDT functions and exposing the performance of both methods. Then we conclude that DE improves the convergence by diminishing the error of each point with the real goal and in most cases improving or preserving the runtime of EA.

For circuit optimization, the objectives to optimize were the voltage or current performances, the variables were the transistor dimensions and the constraints were the saturation condition for each transistor in the circuit.

Afterwards, the first circuit optimization was performed on a VF by optimizing five objectives in voltage mode and five design variables. For this case, MOEA/D found the best results in each objective, albeit NSGA-II found the best average results. In both methods, the relation W/L are similar, this is a signal that both methods find the optimal results in the same region of the search space.

The second circuit optimization was performed on a CCII$^+$, which works with five objectives in voltage mode, and five objectives in current mode, by handling nine design variables. This time the region of the searching space is different for both methods, however, MOEA/D improves its average performance over NSGA-II, but this latter exhibits more symmetry denoted by its standard deviation.

Finally, a CFOA was optimized and the difference was that both EA's found the smallest W/L relation, which is preferable in circuit design, but those values were in different objectives for each method.

In general, for the three circuits, both EA's found closer optimized results, in most cases MOEA/D exhibited the best optimal values but NSGA-II exhibited the best symmetry.

The EA's used herein were implemented in MATLAB and the system has the capability to work with a great number of transistors and it is possible to work with different technology sizes (0.5 μm, 0.35 μm and 0.18 μm) to explore the best design. Also it is possible to define the bounds of the search space to ensure that the optimal solutions are feasible.

Another issue, that we believe must be consider to improve the system, is to include the circuit process variability, because an "optimal" solution might be in a delicate point which does not support the natural variations of a fabrication process. It is impossible to avoid the process variations, but it is possible take them into account in order to know whether a solution can deal with them.

Acknowledgements. This work has been partially supported by CONACyT-MEXICO under the projects 48396-Y and 81604-R, and under the support for the sabbatical leave of the first author at University of California Riverside during 2009-2010.

References

1. Biolek, D., Senani, R., Biolkova, V., Kolka, Z.: Active elements for analog signal processing: Classification, review, and new proposals. Radioengineering 17(4), 15–32 (2008)
2. Tlelo-Cuautle, E., Duarte-Villaseñor, M.A., Guerra-Gómez, I.: Automatic synthesis of VFs and VMs by applying genetic algorithms. Circuits, Systems, Signal Processing 27(3), 391–403 (2008)
3. Tlelo-Cuautle, E., Duarte-Villaseñor, M.A.: Evolutionary electronics: automatic synthesis of analog circuits by GAs. In: Yang Ang, B.L.T., Yin, S. (eds.) Success in Evolutionary Computation. SCI, pp. 165–188. IGI Global (2008)
4. Tlelo-Cuautle, E., Duarte-Villaseñor, M.A., Reyes-García, C.A., Reyes-Salgado, G.: Automatic synthesis of electronic circuits using genetic algorithms. Computación y Sistemas 10(3), 217–229 (2007)
5. Tlelo-Cuautle, E., Moro-Frías, D., Sánchez-López, C., Duarte-Villaseñor, M.A.: Synthesis of CCII-s by superimposing VFs and CFs through genetic operations. IEICE Electron. Express 5(11), 411–417 (2008)
6. McConaghy, T., Gielen, G.G.E.: Template-Free Symbolic Performance Modeling of Analog Circuits via Canonical-Form Functions and Genetic Programming. IEEE Trans on Computer-Aided Design of integrated circuits 28(8), 1162–1175 (2009)
7. Sánchez-López, C., Tlelo-Cuautle, E.: Symbolic Behavioral Model Generation of Current-Mode Analog Circuits. In: Proc. IEEE ISCAS, pp. 2761–2764 (2009)
8. Martens, E., Gielen, G.: ANTIGONE: Top-down creation of analog-to-digital converter architectures. Integration-The VLSI Journal 42(1), 10–23 (2009)
9. Ramesh, C., Rusu, A., Ismail, M., Ismail, M., Skoglund, M.: System co-optimization in wireless receiver design with TrACS. Analog Integrated Circuits and Signal Processing 1-2, 117–127 (2008)

10. Tlelo-Cuautle, E., Guerra-Gómez, I., Reyes-García, C.A., Duarte-Villaseñor, M.A.: Synthesis of Analog Circuits by Genetic Algorithms and their Optimization by Particle Swarm Optimization. In: Chiong, R. (ed.) Intelligent Systems for Automated Learning and Adaptation: Emerging Trends and Applications, pp. 173–192. IGI Global (2010), doi:10.4018/978-1-60566-798-0.ch008
11. McConaghy, T., Palmers, P., Steyaert, M., Gielen, G.: Variation-Aware Structural Synthesis of Analog Circuits via Hierarchical Building Blocks and Structural Homotopy. IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems 28(9), 1281–1294 (2009)
12. Liu, B., Fernandez, F.V., Gielen, G., Castro-López, R., Roca, E.: A Memetic Approach to the Automatic Design of High-Performance Analog Integrated Circuits. ACM Trans on Design Automation of Electronic Systems 14(3), 1–24 (2009)
13. Gielen, G., Martens, E.: Classification of analog synthesis tools based on their architecture selection mechanisms. Integration, the VLSI Journal 41(2), 238–252 (2008)
14. Rutenbar, R.A., Gielen, G., Roychowdhury, J.: Hierarchical modeling, optimization, and synthesis for system-level analog and RF designs. Proc. of the IEEE 95(3), 640–669 (2007)
15. Fakhfakh, M., Masmoudi, S., Tlelo-Cuautle, E., Loulou, M.: Synthesis of switched current memory cells using the nullor approach and application to the design of high performance SI sigma delta modulators. WSEAS Trans. on Electronics, Special Issue: Modern circuit components for analogue signal processing and their applications 5(6), 265–273 (2008)
16. Gupta, S., Bhaskar, D., Senani, R.: New voltage controlled oscillators using CFOAs. AEU-Int. J. of Electronics and Communications 63(3), 209–217 (2009)
17. Alzaher, H.: CMOS digitally programmable quadrature oscillators. Int. Journal of Circuit Theory and Applications 36(8), 953–966 (2008)
18. Sánchez-López, C., Castro-Hernández, A., Perez-Trejo, A.: Experimental verification of the chua's circuit designed with UGC. IEICE Electron. Express 5(17), 657–661 (2008)
19. Sánchez-López, C., Trejo-Guerra, R., Muñoz-Pacheco, J.M., Tlelo-Cuautle, E.: N-scroll chaotic attractors from saturated functions employing CCII+s. Nonlinear Dynamics 61(1-2), 331–341 (2010), doi:10.1007/s11071-009-9652-3.
20. Trejo-Guerra, R., Tlelo-Cuautle, E., Cruz-Hernández, C., Sánchez-López, C.: Chaotic communication system using Chua's oscillators realized with CCII+s. Int. Journal of Bifurcations and Chaos 19(12), 4217–4226 (2009)
21. Castro-López, R., Roca, E., Fernández, F.V.: Multimode pareto fronts for design of reconfigurable analogue circuits. Electronics Letters 45(2), 95–96 (2009)
22. Liu, B., Wanga, Y., Yu, Z., Liu, L., Li, M., Wanga, Z., Lu, J., Fernández, F.V.: Analog circuit optimization system based on hybrid evolutionary algorithms. Integration, the VLSI Journal 42(2), 137–148 (2009)
23. Fakhfakh, M.: A novel alienor-based heuristic for the optimal design of analog circuits. Microelectronics Journal 40, 141–148 (2009)
24. Guerra-Gómez, I., Tlelo-Cuautle, E., Li, P., Gielen, G.: Simulation-based optimization of UGCs performances. In: IEEE Int. Caribbean Conference on Devices, Circuits and Systems (2008)
25. Eeckelaert, T., McConaghy, T., Gielen, G.: Efficient multiobjective synthesis of analog circuits using hierarchical Pareto-optimal performance hypersurfaces. In: Proc. Design Automation and Test in Europe, vol. 2(1), pp. 1070–1075 (2005)
26. Fakhfakh, M., Cooren, Y., Sallem, A., Loulou, M., Siarry, P.: Analog Circuit Design Optimization through the Particle Swarm Optimization Technique. Analog Integrated Circuits and Signal Processing 63(1), 71–82 (2010), doi:10.1007/s10470-009-9361-3.

27. Medeiro, F., Rodríguez-Macías, R., Fernández, F.V., Domínguez-Castro, R., Huertas, J.L., Rodríguez-Vázquez, A.: Global design of analog cells using statistical optimization techniques. Analog Integrated Circuits and Signal Processing 6, 179–195 (1994)
28. Barros, M., Guilherme, J., Horta, N.: Analog circuits optimization based on evolutionary computation techniques in Integration. The VLSI Journal 43(1), 136–155 (2010)
29. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evolutionary Computation 6(2), 182–197 (2002)
30. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. IEEE Trans. Evolutionary Computation 8(2), 173–195 (2000)
31. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Trans. Evolutionary Computation 13(2), 284–302 (2009)
32. Olensek, J., Burmen, A., Puhan, J., Tuma, T.: DESA: a new hybrid global optimization method and its application to analog integrated circuit sizing. Journal of Global Optimization 44(1), 53–77 (2008)
33. Guerra-Gómez, I., Tlelo-Cuautle, E., Reyes-García, C.A., Reyes-Salgado, G., de la Fraga, L.G.: Non-sorting genetic algorithm in the optimization of unity-gain cells. In: 6th International Conference on Electrical and Electronics Engineering, Toluca, Mexico, November 11-13, pp. 364–367. IEEE Press, Los Alamitos (2009)
34. Guerra-Gómez, I., Tlelo-Cuautle, E., McConaghy, T., Gielen, G.: Optimizing current conveyors by evolutionary algorithms including differential evolution. In: IEEE ICECS Special Session: Applications of Evolutionary Computation Techniques to Analog, Mixed-Signal and RF Circuit Design, Hammamet, Tunisia, December 13-16, pp. 259–262 (2009)
35. Roca, I., Fakhfakh, M., Castro-López, R., Fernández, F.V.: Applications of Evolutionary Computation Techniques to Analog, Mixed-Signal and RF Circuit Design An Overview. In: IEEE ICECS Special Session: Applications of Evolutionary Computation Techniques to Analog, Mixed-Signal and RF Circuit Design, Hammamet, Tunisia, December 13-16, pp. 251–254 (2009)
36. Guerra-Gómez, I., Tlelo-Cuautle, E., McConaghy, T., Gielen, G.: Decomposition-Based Multi-Objective Optimization of Second Generation Current Conveyors. In: IEEE MWSCAS, pp. 220–223 (2009)
37. Polanco-Martagón, S., Flores-Becerra, G., Tlelo-Cuautle, E.: Computing Optimum Sizes of a Voltage Follower using Fuzzy Sets. In: Tlelo-Cuautle, E., Polanco-Martagón, S. (eds.) IEEE MWSCAS, Applying Fuzzy Sets Intersection in the Sizing of Voltage Followers, LANMR, vol. 533, pp.209–216, 216–219 (2009), http://CEUR-WS.org/Vol-533/
38. Polanco-Martagón, S., Flores-Becerra, G., Tlelo-Cuautle, E.: Fuzzy Set Based Approach to Compute Optimum Sizes of Voltage Followers. In: IEEE ICECS, Hammamet, Tunisia, December 13-16, pp. 844–847 (2009)
39. Maricau, E., De Wit, P., Gielen, G.: An analytical model for hot carrier degradation in nanoscale CMOS suitable for the simulation of degradation in analog IC applications. Microelectronics Reliability 48(8-9), 1576–1580 (2008)
40. McConaghy, T., Gielen, G.G.E.: Globally Reliable Variation-Aware Sizing of Analog Integrated Circuits via Response Surfaces and Structural Homotopy. IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems 28(11), 1627–1640 (2009)
41. Torres-Muñoz, D., Tlelo-Cuautle, E.: Automatic biasing and sizing of CMOS analog integrated circuits. In: IEEE MWSCAS, pp. 915–918 (2005)
42. Storn, R., Price, K.: Minimizing the real functions of the ICEC'96 contest by Differential Evolution. In: IEEE International Conference on Evolutionary Computation, pp. 842–844 (1996)

43. Storn, R.: Diferential Evolution Research – Trends and Open Questions in Advances in Differential Evolution. Springer, Heidelberg (2008)
44. Zhang, J., Sanderson, A.C.: JADE: Self-Adaptive Differential Evolution with Fast and Reliable Convergence Performance. In: 2007 IEEE Congress on Evolutionary Computation, pp. 2251–2258 (2007)
45. Iorio, A.W., Li, X.: Solving rotated multi-objective optimization problems using differential evolution. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 861–872. Springer, Heidelberg (2004)
46. Zhang, Q., Hui, L.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Transactions on Evolutionary Computation 11(6), 712–731 (2007)

Part II

Algorithm Integration

Application of Estimation of Distribution Algorithms for Nuclear Fuel Management

Shan Jiang and Jonathan Carter

1 Introduction

In 2007, 30 countries were operating a total of 439 commercial nuclear power reactors which contributed about 16% of the world's total electrical power. With concerns about global warming it is likely that more reactors will be built in the near future.

In this paper we will describe, in general terms, the operation of a commercial nuclear reactor and how this leads to a difficult problem in combinatorial optimisation. We then describe the principles of the Estimation of Distribution Algorithm (EDA) and how we have adapted it to solve a combinatorial problem, it is demonstrated using the travelling salesman problem. Next we explain how the method was modified to solve the nuclear fuel management problem and how heuristic information was incorporated. Finally we examine the performance of the EDA on three test problems for the CONSORT reactor, a small research reactor at Imperial College in London, and show how this compares to the performance of a Genetic Algorithm (GA), which is regarded as the best current optimisation algorithm for this problem[24].

1.1 Understanding the Principles of a Reactor

All current power generation nuclear reactors use the same basic design. Figure 1 shows a schematic of the basic design. In the reactor vessel we find the nuclear core where nuclear fission is producing large amounts of heat. This heat is removed from

Shan Jiang
Imperial College, South Kensington, London, UK
e-mail: shan.jiang04@imperial.ac.uk

Jonathan Carter
Imperial College, South Kensington, London, UK
e-mail: J.N.Carter@imperial.ac.uk

the core by a fluid that flows through and around the fuel elements contained within the core. The most commonly used fluid is pressurised water, as shown here. Cool fluid is forced in at the base of the reactor by a pump, and hot fluid leaves the top of the reactor vessel. The hot fluid returns to the pump via a heat exchanger. In the heat exchanger water is boiled to form high pressure stream, which passes through a series of turbines which are used to turn electrical generators. After leaving the turbine the steam is cooled and condensed back to water by a secondary heat exchanger (not shown). This second heat exchanger is part of a system that releases waste heat energy to the environment. In many power stations the large cooling towers are the externally visible component of this.

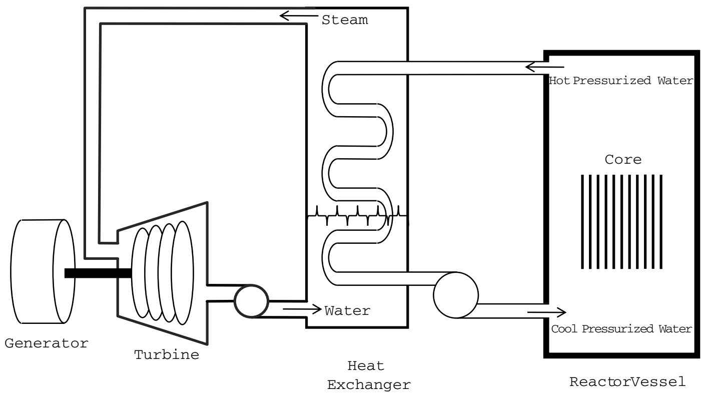


Fig. 1 A schematic of a typical pressurized water reactor.

In a nuclear reactor the fuel is contained in long tubes, known as a fuel assembly. Different assemblies can contain different amounts of nuclear fuel and how much energy can be release from a fuel assembly depends on where it is placed in the reactor. The optimal placement of the fuel assemblies within the reactor core is the problem that we attempt to solve.

Figure 2 is a schematic of the core of a typical pressurised water reactor, in this case the Russian VVER 440/230. There are a total of 349 fuel channels, of which 30 are occupied by safety assemblies, 7 by control assemblies and 312 by fuel assemblies. Each fuel assembly is one of three types depending on the level of enriched fuel that it contains. To ensure safety, there are 37 control rods, 30 of which are out of core during operation, and seven which are partially inserted and control the normal operation of the reactor. To shut the reactor down all 37 control/safety rods are fully inserted.[1]

We can observe that the core loading pattern has a high level of symmetry. This reflects the underlying symmetry that is incorporated into the design of the reactor

[1] Reactors are designed so that complete shutdown can be achieved without the need to fully insert all 37 control/safety rods.

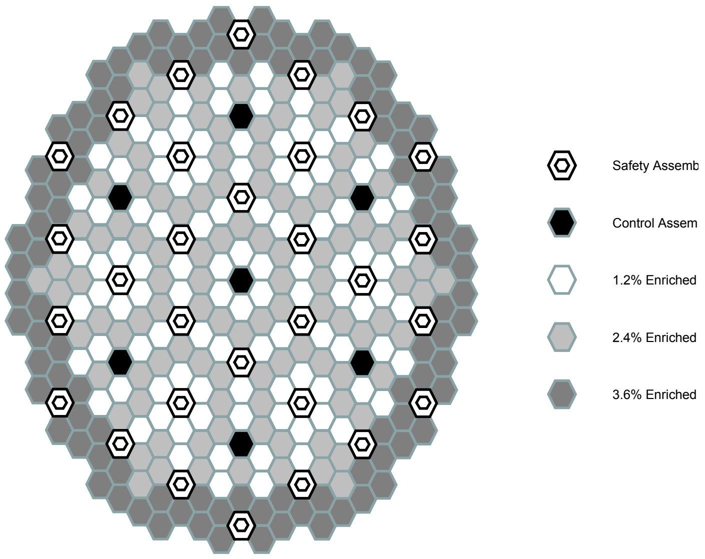


Fig. 2 A Core Map of the VVER 440/230, a typical Pressurized Water Reactor, showing a possible fuel loading pattern at the start of reactor life.

and its cooling/management systems. Typically reactors operate with either 4-fold or 6-fold symmetry, and the symmetry can be rotational and/or reflective. Figure 2 shows an initial fuel loading which is designed to ensure the safe start up of the reactor. The more interesting situation occurs periodically through the life of the reactor when used, burnt, fuel assemblies need to be removed from the reactor and replaced with fresh fuel. Depending on the reactor design, the time between refuelling can vary from a few weeks to many months. For example the Sizewell B pressurised water reactor in Britain, is designed to be shutdown and refuelled every 18 months. At this point a third of the fuel is removed and replaced with fresh fuel assemblies. However, we do not simply take one fuel assembly out and replace it with a new one, as we have the opportunity to shuffle the positions of all of the assemblies within the core. This is our optimisation problem: where exactly should each fuel assembly go?

1.2 The Optimisation Problem

Despite the complexity of nuclear reactors, the Loading Pattern (LP) optimisation problem can be expressed as a simple assignment problem: to which of the loadable positions should each of the available fuel assemblies be assigned, subject to a number of operational constraints, so as to maximise/minimise some objective function? For the VVER reactor there are 312 fuel assemblies, being a mixture of new and partially burnt, which have to be allocated to 312 fuel channels. If the reactor is managed on four fold symmetry, then the problem becomes one of optimally allocating 78 assemblies to 78 channels.

The objective of the shuffling of the fuel is two fold. Firstly we must ensure that the reactor is operating safely, and secondly we will be trying to optimise some measure of performance. In a commercial reactor this measure is normally the profit made from selling the electricity generated. In non-commercial reactors other criteria may be more important such as maximising the time between refuelling subject to achieving a minimum power output. In research reactors the objective is often to maximise the number of emitted neutrons which are used for scientific experiments.

In an optimisation process, the objective function(s) will be evaluated many times for different candidate solutions. Since the necessary physical experiments are not feasible, the only way to do it is through a computer simulation. Simulation software is able to provide accurate predictions of all of the key measurements of the reactor. However, even with the most up-to-date computing facilities, the reactor simulation is still very computationally expensive. This is a key driver for the continuing search for better optimisation algorithms. Furthermore, due to the complexity of the nuclear reaction and the reactor system, most objective functions are non-linear and have multiple local optima. This increases the difficulties of the optimisation process.

1.3 The Variables and the Search Space

As described above our problem is to allocate the available fuel assemblies to the available assembly channels. The size of the problem space, that is the number of different Loading Patterns (LPs), depends on the number of fuel assemblies available. In the VVER example there are 312 fuel assemblies to load into the core. If we ignore the fact that some of the new fuel assemblies are identical to each other, then the number of possible LPs is $312! \approx 2 \times 10^{644}$. If we assume that the reactor has four fold symmetry, that at a refuelling a third of the fuel is replaced and that all the new fuel assemblies are of the same type, then in each quarter of the reactor there are 52 used fuel assemblies and 26 identical new fuel assemblies the number of possible LPs for these 78 fuel assemblies is $\frac{78!}{26!} \approx 3 \times 10^{88}$. These numbers are too large to perform an exhaustive search. Because of the size of the search space and the complexity of the objective function, it is impractical to use manual methods to identify optimal LPs. Thus an optimisation algorithm is required to automate the search.

2 Description of a Basic Estimation of Distribution Algorithm

Estimation of Distribution Algorithms (EDAs) have evolved from Genetic Algorithms. They work by using a sequence of populations to estimate the probability that a particular solution is the one that you want. At the start of the optimisation process, when no information is available, one would typically have a uniform distribution. This means that every solution is considered to be equally likely to be the solution to the problem under investigation. The probability model is then refined as the parameter space is explored, using individual solutions, which are usually stochastically selected from the current probability model. In the limit of all possible solutions being evaluated, the probability model will collapse into a delta function.

All EDAs have the same iterative process of sampling from the current probability model, updating the probability model, and then repeating the iteration. Each variation on the basic principle uses a different probability model and a different way of updating that probability model. In this section, we describe the outline of a basic EDA and demonstrate its application through a simple example.

The main steps of an EDA are:

1. Initialise the probability model to a uniform distribution or pre-defined distribution;
2. Sample new solutions from the probability model and calculate their objective function values;
3. Select some individuals from the current population, based on their objective function values;
4. Revise the probability model using the information extracted from selected individuals;
5. If the stop criteria are not met, then go to step 2, otherwise end the search.

Key considerations are how to construct, sample and update the probability models.

To illustrate the application of an EDA, we will consider the Univariate Marginal Distribution Algorithm (UMDA, [14]), and use it to solve a onemax problem. The optimisation problem is:

$$\max f(X) = \sum_{i=1}^{3} x_i = x_1 + x_2 + x_3; \tag{1}$$

where the input variable X is:

$$X = [x_1, x_2, x_3], \qquad x_i \in \{0,1\}, i \in \{1,2,3\} \tag{2}$$

The steps in the UMDA are:

1. **Initialise a uniform distribution model** P. We use ${}^{t}P$ to represent the distribution model at iteration t, here $t = 0$. The data structure of the probability model is a real-valued vector P having the same dimension as X, in which each p_i represents the probability of x_i being 1.

$$^{0}P = [p_1, p_2, p_3] = [0.5, 0.5, 0.5] \tag{3}$$

2. **Sample M individuals from the model ^{0}P, and compute their objective function values**. In this example we will use $M = 6$. Initialise an empty individual ${}^{t}_{j}X$ (i.e. none of the ordinates are assigned a value), where t is the generation number and j is the individual ID in the current population. Here $t = 0$ and $j = 1$. For each p_i, generate a random number $r_i \in [0,1]$ from a uniform distribution:

$$\begin{aligned} r &= [\ r_1,\ r_2,\ r_3\ ] = [\ 0.4, 0.5, 0.1\ ] \\ {}^0P &= [\ p_1,\ p_2,\ p_3\ ] = [\ 0.5, 0.5, 0.5\ ] \\ {}^0_1X &= [\ x_1,\ x_2,\ x_3\ ] = [\ -,\ \ -,\ \ -\ ] \end{aligned} \tag{4}$$

Then compare each r_i and p_i, if $r_i < p_i$, then set $x_i = 1$, otherwise, $x_i = 0$. Since $0.4 < 0.5$, $0.5 = 0.5$ and $0.1 < 0.5$

$${}^0_1X = [1,0,1] \tag{5}$$

and the objective function is calculated:

$$f({}^0_1X) = 1+0+1 = 2 \tag{6}$$

Similarly, other individuals can be randomly generated and the current population becomes

$$\begin{aligned} {}^0_1X &= [1,0,1]\ f({}^0_1X) = 2 \\ {}^0_2X &= [1,1,1]\ f({}^0_2X) = 3 \\ {}^0_3X &= [1,0,0]\ f({}^0_3X) = 1 \\ {}^0_4X &= [0,1,0]\ f({}^0_4X) = 1 \\ {}^0_5X &= [1,0,1]\ f({}^0_5X) = 2 \\ {}^0_6X &= [0,0,1]\ f({}^0_6X) = 1 \end{aligned} \tag{7}$$

3. **Select the N best individuals.** In this example we will use $N = 3$. The selected IDs are $j \in \{1,2,5\}$

$$\begin{aligned} {}^0_1X &= [1,0,1] \\ {}^0_2X &= [1,1,1] \\ {}^0_5X &= [1,0,1] \end{aligned} \tag{8}$$

4. **Update the probability model by counting the frequency for each x_i in the selected individuals.** The updating method is described by

$${}^1P(x_i) = {}^1P(x_i = 1) = \frac{\sum_j x_{ij}}{N} \tag{9}$$

in which the updated ${}^1P$ will be the sampling model at generation/iteration 1, i is the ID number of the input variables, j is the ID of the selected individuals and N is the number of selected individuals. Hence

$$\begin{aligned} {}^1P(x_1 = 1) &= \tfrac{\sum_j x_{1j}}{N} = \tfrac{\sum_j x_{1j}}{3} = 1 \\ {}^1P(x_2 = 1) &= \tfrac{\sum_j x_{2j}}{N} = \tfrac{\sum_j x_{2j}}{3} = \tfrac{1}{3} \\ {}^1P(x_3 = 1) &= \tfrac{\sum_j x_{3j}}{N} = \tfrac{\sum_j x_{3j}}{3} = 1 \end{aligned} \tag{10}$$

The updated P is:

$${}^1P = [1, \frac{1}{3}, 1] \approx [1.0, 0.3, 1.0] \tag{11}$$

5. **Stop if maximum number of generations is reached, otherwise go back to 2 and sample new solutions**. The current best solution is

$$ {}^{0}_{2}X = [1,1,1] \; f({}^{0}_{2}X) = 3 \tag{12}$$

The UMDA is one of the simplest forms of EDA. However, its idea is the basis of many of the variants. It estimates the probability of the distribution of the promising solutions and maintains it with a separate data structure, in which the input variables are considered as independent of each other.

In this basic version, the distribution model is re-estimated in each generation. There are other variants of UMDAs using slightly different updating methods. For example, in the original UMDA paper, [14], an alternative implementation is introduced, which uses an incremental learning strategy to update the distribution model:

$$ {}^{t+1}P(x_i) = (1-\lambda)^{t}P(x_i) + \lambda \frac{\sum_{j=1}^{N} x_i}{N} \tag{13}$$

in which λ is a scalar, and N is the number of selected candidates.

3 Applying Estimation of Distribution Algorithms to Travelling Salesman Problems

The algorithm structure for EDAs is very similar to that of GAs. They both search by evolving a population of candidate solutions. The main difference is the way in which they reproduce and maintain the candidate pool. The key points for the application of EDAs to the Travelling Salesman Problem (TSP), and other problems, are the encoding method for representing the candidate solutions, the type of the probability model, and how to use it to generate new solutions. We discuss the encoding method used by EDAs for TSP first, followed by the use of the probability model and by a discussion of some performance-enhancement methods.

3.1 Encoding for the TSPs

The TSPs are a set of classical combinatorial optimisation problems that look for the shortest route by which a salesman can visit a set of cities. It can be described as ordering a set of cities in such a way as to minimise the length of the path along which they lie[2]. A TSP is easily abstracted as a permutation, which is one of the most natural ways of encoding TSPs, and has been used in the research of GAs and Simulated Annealing for TSP. There are other methods of encoding a TSP and they were mainly introduced in GA research. For example, the Ordinal

[2] There are several variants on the TSP, they can be cyclic or non-cyclic depending if you are required to finish where you started, and they can be symmetric or non-symmetric depending on whether the distance $A \rightarrow B$ is the same as the distance $B \rightarrow A$. This may mean that an encoding may contain some slight redundancy.

Representation [11], the Adjacency Representation [11], the Adjacency Listing Representation [20], the Position Listing Representation [16] and the Precedence Matrix Representation [8]. As pointed out by previous studies in GAs, the candidate solutions need to be encoded naturally into binary strings to produce good results [4]. In recent years, other types of data structure have been used in GAs. These representations are normally associated with specially-designed crossover and mutation operators.

It is generally believed that the GAs' performance on the TSP type of problems has not been particularly impressive [17]. This is mainly because of the way in which solutions are represented and reproduced in GAs (crossover and mutation) is not straightforward or 'natural'. The candidate solutions in the GAs contain the encoding of the problem explicitly, and the dynamics in the search process implicitly. Some hidden information, or the 'Schemata' in GA terminology, can easily be disrupted during the crossover and/or mutation process. In consequence the search will contain more random jumps and the search space may not be explored smoothly.

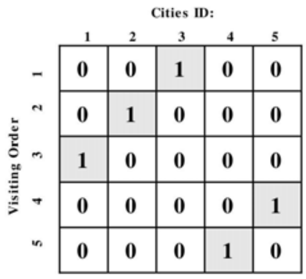
In the EDAs, we use a natural encoding of the solutions to preserve the generality of the algorithm. At the same time, a separate distribution probability model is used to keep the statistical information gathered during the search or any relevant information. New solutions are generated by sampling the model. This mechanism separates the encoding of the candidate solution and the dynamics of the optimisation process. It is clear that in such EDAs, the problem encoding is not associated with solution-reproducing operators. This enables users to use the most straightforward encoding method without looking for the associated crossover operator, or having to develop a new one. The EDAs' generality is also well preserved.

For the EDAs applied to TSPs, we use the permutation representation. A candidate solution for a TSP is represented in an integer vector that contains a permutation of integers from 1 to N, where N is the number of cities. A candidate tour of a five-city TSP is illustrated below:

$$\begin{aligned} \text{Cities}: &\ \{1,2,3,4,5\} \\ \text{CandidateTour}: &\ \{3,2,1,5,4\} \end{aligned} \tag{14}$$

where the candidate tour represents a possible shortest path, visiting city 3 first, then city 2, city 1, city 5, city 4 and back to city 3.

Alternatively, this permutation can also be transformed to a binary matrix form, as shown in figure 3. The rows stand for the visiting order and the columns are the city IDs. Entry $[m,n]$ set to '1' means the n_{th} city will be visited at the m_{th} stop of the tour. Since each city should be visited once and only once, each row and column has one and only one '1' bit. This binary matrix form of a permutation makes no difference in representing a tour. The reason we introduce the binary matrix is to make it easier to understand the relationship between the candidate tour and the probability model in subsequent sections. A candidate tour of the TSP is a combination of some entries of the whole matrix, as illustrated in figure 3. These entries can be regarded as the building block of the TSPs. The binary matrix contains all



Cities ID:

Visiting Order	1	2	3	4	5
1	0	0	1	0	0
2	0	1	0	0	0
3	1	0	0	0	0
4	0	0	0	0	1
5	0	0	0	1	0

Fig. 3 The binary matrix form of a Permutation Representation of a tour for a five-city TSP. Each row and each column only has one '1' bit.

the possible building blocks to form any candidate solution. The optimisation then consists of picking some of the matrix entries.

3.2 Probability Model for Generating Candidate Solutions

EDAs can be categorised into two sub-classes. One uses a relatively simple probability model that has no dependence between any input variable. The other sub-class uses a probability model that does have input variable dependencies. The former is obviously simpler while the latter is generally believed to have better modelling capacity, although, it can be very complicated and computationally expensive.

The probability model in EDAs is utilised to capture and preserve the information found during the search, as well as any relevant heuristic information. Given a complicated real-world application, the input variables - namely, the encoded candidate solutions - may or may not interact with each other. The dependent models are better at capturing these relationships, and ideally can improve the search. The dependency structure can be decided beforehand, using background knowledge, or achieved on the fly during the search.

Despite the theoretical advantage of probability models with variable dependency, simple EDAs with a non-dependent model are frequently used in real-world applications [22]. One reason for this is that the computational cost of the dependent models can be prohibitively high. Since one of the drivers for using an EDA is to find near-optimal solutions with reasonable time and effort [16], the application of the dependent models may not be cost-efficient.

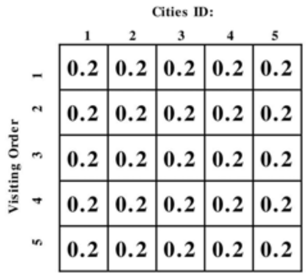
The other reason is that, when used in EDAs, the statistical data available for finding the dependencies is limited. The current candidate solutions are literally all the data that is available, which may be insufficient for extensive statistical analysis. The dependency models may fail to capture the variables' dependencies accurately, due to incomplete data. The performance of EDAs with non-dependent models can be improved by combining with local search or heuristic information. Promising results have been published, for example in [22], where an EDA with an independent probability model combined with derivative-free local search has been tested on a set of difficult continuous functions.

Given the above reasons, we will apply an EDA with an independent probability model to the TSPs. It is simple to understand and sufficiently well-analysed to be extended to the solution of very complicated real-world applications.

3.2.1 A Non-dependent Model for TSPs

The best-known EDAs with the non-dependent distribution model are the UMDA - Univariate Marginal Distribution Algorithm [14], the PBIL - Population Based Incremental Learning [3], and the cGA - Compact Genetic Algorithm [12].

For a five-city TSP, the data structure of the probability model used is illustrated in figure 4. A matrix entry (m,n) represents the probability of visiting city n at the m_{th} stop. Initially, the model can be a uniform distribution if no background information is given. Hence each matrix entry is set to 0.2 because there are five cities to be chosen randomly.



Cities ID:

Visiting Order	1	2	3	4	5
1	0.2	0.2	0.2	0.2	0.2
2	0.2	0.2	0.2	0.2	0.2
3	0.2	0.2	0.2	0.2	0.2
4	0.2	0.2	0.2	0.2	0.2
5	0.2	0.2	0.2	0.2	0.2

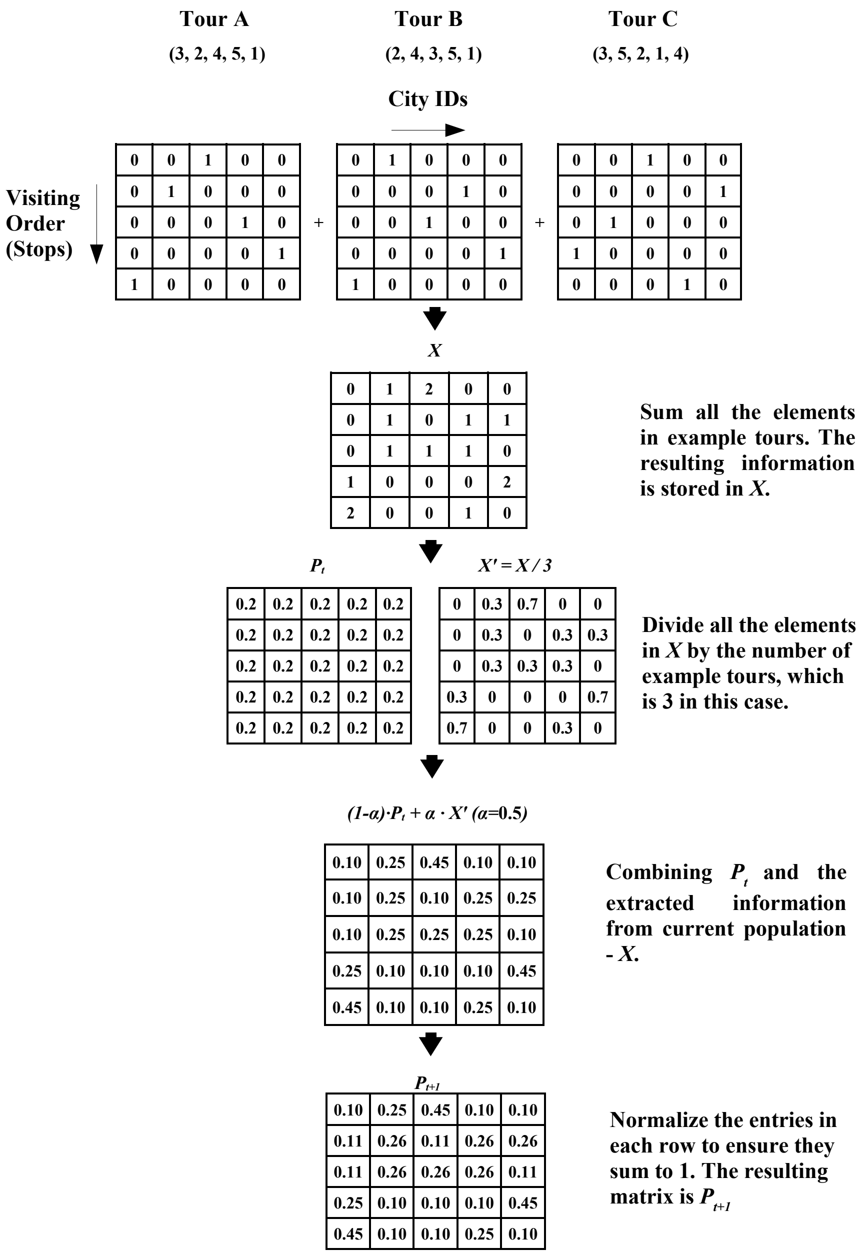
Fig. 4 The data structure of the initial non-dependent probability model P for a hypothetical five-city TSP. Each entry represents the initial probability of visiting a city (column) at a certain stop (row).

The probability model updating method we use is similar to that for PBIL. It can be described as:

$$P_{t+1} = (1-\alpha)P_t + \alpha X \tag{15}$$

where P is the probability model; t is the number of the iteration or generation; P^0 is a uniform distribution model; α is a scalar between $[0,1]$; and X is the frequency of occurrence of each city-stop arrangement, extracted from the selected candidate tours. The independent model implies that the arrangements of cities to stops are independent of each other. An example is illustrated in figure 5.

Having updated the probability model by estimating the distribution of the promising solutions, new candidate solutions will be generated by sampling the model. An example of generating new tours for TSPs is illustrated in figure 6.


Tour A
(3, 2, 4, 5, 1)
Tour B
(2, 4, 3, 5, 1)
Tour C
(3, 5, 2, 1, 4)
City IDs
Visiting Order (Stops)
+
+
X
Sum all the elements in example tours. The resulting information is stored in X.
P_t
$X' = X / 3$
Divide all the elements in X by the number of example tours, which is 3 in this case.
$(1-\alpha)\cdot P_t + \alpha \cdot X'$ (α=0.5)
Combining P_t and the extracted information from current population - X.
P_{t+1}
Normalize the entries in each row to ensure they sum to 1. The resulting matrix is P_{t+1}

Tour A

0	0	1	0	0
0	1	0	0	0
0	0	0	1	0
0	0	0	0	1
1	0	0	0	0

Tour B

0	1	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	0	0	1
1	0	0	0	0

Tour C

0	0	1	0	0
0	0	0	0	1
0	1	0	0	0
1	0	0	0	0
0	0	0	1	0

X

0	1	2	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	0	2
2	0	0	1	0

P_t

0.2	0.2	0.2	0.2	0.2
0.2	0.2	0.2	0.2	0.2
0.2	0.2	0.2	0.2	0.2
0.2	0.2	0.2	0.2	0.2
0.2	0.2	0.2	0.2	0.2

$X' = X / 3$

0	0.3	0.7	0	0
0	0.3	0	0.3	0.3
0	0.3	0.3	0.3	0
0.3	0	0	0	0.7
0.7	0	0	0.3	0

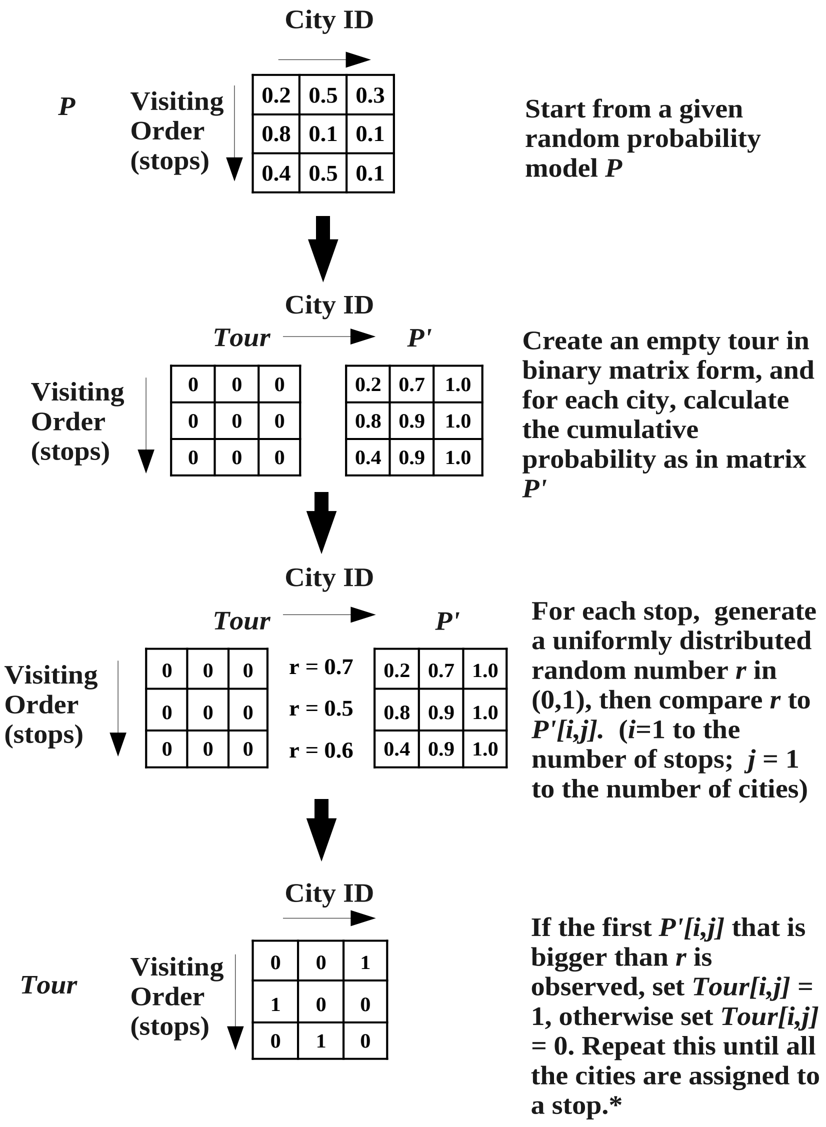
$(1-\alpha)\cdot P_t + \alpha \cdot X'$ (α=0.5)

0.10	0.25	0.45	0.10	0.10
0.10	0.25	0.10	0.25	0.25
0.10	0.25	0.25	0.25	0.10
0.25	0.10	0.10	0.10	0.45
0.45	0.10	0.10	0.25	0.10

P_{t+1}

0.10	0.25	0.45	0.10	0.10
0.11	0.26	0.11	0.26	0.26
0.11	0.26	0.26	0.26	0.11
0.25	0.10	0.10	0.10	0.45
0.45	0.10	0.10	0.25	0.10

Fig. 5 An example of updating a probability model in EDAs.



Start from a given random probability model *P*

Create an empty tour in binary matrix form, and for each city, calculate the cumulative probability as in matrix *P'*

For each stop, generate a uniformly distributed random number *r* in (0,1), then compare *r* to *P'[i,j]*. (*i*=1 to the number of stops; *j* = 1 to the number of cities)

If the first *P'[i,j]* that is bigger than *r* is observed, set *Tour[i,j]* = 1, otherwise set *Tour[i,j]* = 0. Repeat this until all the cities are assigned to a stop.*

*** In the final step, each city can only be assigned to a stop once and only once. So when a city is assigned, it should be excluded from the candidate list, to forbid it to be used for a later stop. The last available city in each row at each iteration should have its cumlative probability set to 1.0. The stops are processed from the top of the matrix to the bottom in this case.**

Fig. 6 An example of sampling new candidate tours from a non-dependent probability model in the EDAs.

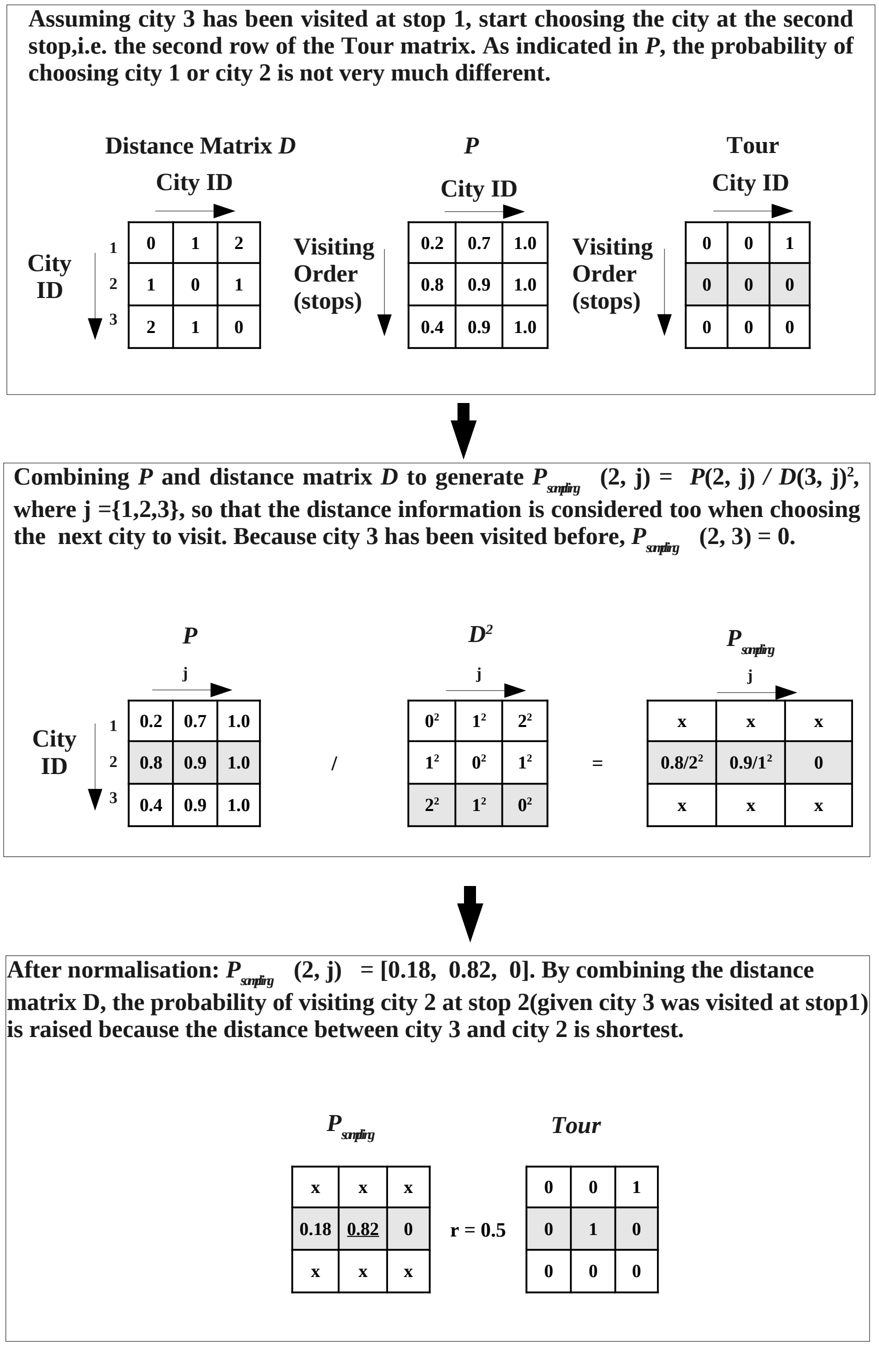

Assuming city 3 has been visited at stop 1, start choosing the city at the second stop,i.e. the second row of the Tour matrix. As indicated in P, the probability of choosing city 1 or city 2 is not very much different.
Distance Matrix D
City ID
City ID
1 2 3
0 1 2
1 0 1
2 1 0
P
City ID
Visiting Order (stops)
0.2 0.7 1.0
0.8 0.9 1.0
0.4 0.9 1.0
Tour
City ID
Visiting Order (stops)
0 0 1
0 0 0
0 0 0
Combining P and distance matrix D to generate P_sampling (2, j) = P(2, j) / D(3, j)², where j ={1,2,3}, so that the distance information is considered too when choosing the next city to visit. Because city 3 has been visited before, P_sampling (2, 3) = 0.
P
j
City ID
1 2 3
0.2 0.7 1.0
0.8 0.9 1.0
0.4 0.9 1.0
/
D²
j
0² 1² 2²
1² 0² 1²
2² 1² 0²
=
P_sampling
j
x x x
0.8/2² 0.9/1² 0
x x x
After normalisation: P_sampling (2, j) = [0.18, 0.82, 0]. By combining the distance matrix D, the probability of visiting city 2 at stop 2(given city 3 was visited at stop1) is raised because the distance between city 3 and city 2 is shortest.
P_sampling
x x x
0.18 0.82 0
x x x
r = 0.5
Tour
0 0 1
0 1 0
0 0 0

Fig. 7 An example of combining the heuristic information with the probability model in the EDAs for TSPs: generating a candidate tour with population-based learning and the distance matrix.

3.2.2 Combining with Heuristic Information

Using problem-dependent heuristic information can improve the performance of optimisation algorithms, especially the black-box type of methods. Examples include the use of a distance matrix in the Ant Colony Optimisation (ACO, [5]) and a multi-objective EDA with problem-dependent regularity information [23].

The distance between cities in the TSPs is useful and makes a direct contribution to the objective function - the tour length. We use this information in the proposed EDAs to improve its performance. The method is from the ACO [5]. Using the data structure of the probability model and the tour sampling operator in figures 6 and 5, it is straightforward to combine the distance information into the probability model in order to guide the tour regeneration. The probability model is perturbed in the sampling operator as follows:

$$P_{sampling}(i,j) = \frac{P(i,j)}{D(k,i)^{\beta}} \tag{16}$$

where $P_{sampling}$ is the probability used to sample new tours, P is the probability updated by the standard updating operator shown in figure 5, D is the distance matrix, $D(m,n)$ is the distance from city m to city n, i is the visiting order or the stops, j is the city IDs, k is the ID of the city visited at the last stop and β is a scalar.

The above method is incorporated in the sampling operator. The idea is that the choice of the next city to visit depends on how often it is used in known promising tours (see figure 5), as well as on the distance between the candidate cities and the last visited city. By using this sampling probability, the length of sub-paths is combined into the EDA.

Figure 7 illustrates how a new tour is generated for a 3-city TSP. The distance between city 1 and 2, and between 2 and 3, is one unit, and the distance between city 1 and city 3 is two units.

4 Adaptation of the Algorithm to the Nuclear Power Problem

In the remainder of this paper we will be describing how to apply the algorithm to a real reactor. The reactor is the research reactor at Imperial College (CONSORT). In this section we start by briefly describing the reactor, we then discuss the loading pattern representation and the heuristic information used. This section is then followed by three test cases.

4.1 The CONSORT Reactor

The CONSORT reactor is a small research reactor at Imperial College. It has just 24 fuel channels in the core for fuel assemblies, however there is no reactor symmetry. The simple shuffling problem size is approximately 6×10^{23}, smaller than for a commercial reactor, but still substantial. The objective function is also different from commercial reactors as this reactor is not used to generate power.

The CONSORT Reactor was first constructed in 1965, and was subsequently expanded in 1971. It has been in continuous safe operation since 1965 at the Imperial College Reactor Centre and is the only civil research reactor in the UK. It is designated as a low-power research reactor to provide neutrons for research and engineering applications. The reactor centre provides facilities for the university and other educational institutions to be used for teaching and research in many fields of nuclear science and technology, such as reactor physics, reactor engineering, neutron physics, solid-state physics, radiochemistry and activation analysis. The reactor centre also provides radioisotopes for use in other laboratories.

The CONSORT core consists of 24 fuel assemblies that contain highly enriched Uranium fuel plates in an Uranium/Aluminium alloy [9], [10]. There are four control rods, rods no. 1, 2, 3 and 4, in the core. A photograph of the CONSORT reactor is shown in figure 8 alongside a 2D core plan.

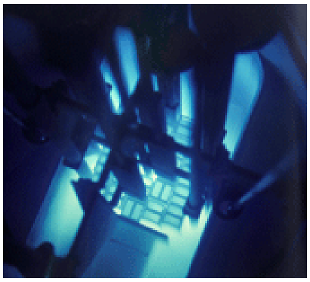
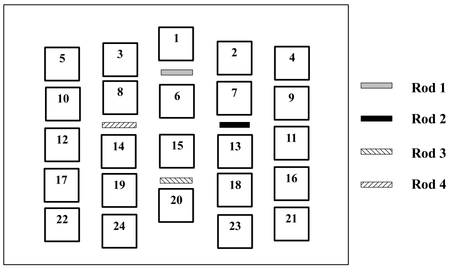



Fig. 8 A photograph of the CONSORT Reactor and a schematic of the core plan.

There are five types of fuel assemblies, referred to as MARK I_A, MARK I_B, MARK I_C, MARK II and MARK III. The MARK I_A fuel assemblies are the oldest and contain 12 fuel plates which are slightly curved and were manufactured in 1965. The Mark I_B fuel elements contain 6 fuel plates and the MARK I_C fuel elements contain 3 fuel plates. MARK II fuel elements also have 12 plates, with the fuel plates being thicker than in the MARK I; the plates are not curved and contain about 4g more Uranium metal, and are therefore more reactive than MARK I_A, MARK I_B and MARK I_C plates. MARK III elements have 16 plates that are thinner than the MARK I and MARK II plates. The reactivity (K_∞) of the five different types of fuel assemblies is summarised in table 1. The listed fuel inventory information is based on their initial designed status. Since the amount of uranium has decayed in the last four decades, we used a modified fuel inventory in our test cases by re-calculating their K_∞. There are only two MARK II fuel elements available and they must be inserted in channel numbers 6 and 15.

Table 1 K_∞ for the five fuel types for the Imperial College CONSORT Reactor

Fuel Name	MARK I_A	MARK I_B	MARK I_C	MARK II	MARK III
K_∞	1.66197	1.16471	0.79238	1.77473	1.78583

4.2 Loading Pattern Representation

An LP of a reactor is a set of assignments of N items to M positions, where N is the number of types of fuel assemblies and the M positions could include in-core fuel channels and out-of-core store locations.

In a binary matrix, with M rows and N columns, each row of the matrix represents an in-core fuel channel, while each column represents a fuel assembly ID or fuel assembly type ID. Entry $LP(i, j)$ is set to 1 if and only if fuel channel i is loaded with fuel ID j, otherwise $LP(i, j)$ is 0. Since one channel can only be occupied by one fuel element, there is only one 1 on each row.

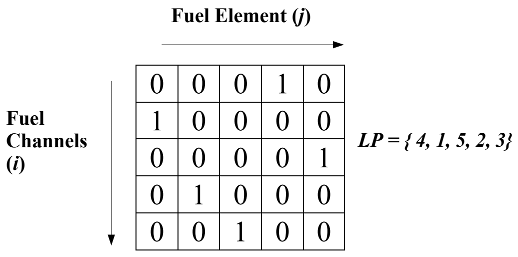


Fig. 9 A binary matrix representation of a reactor loading pattern.

4.3 Heuristic Information for the Fuel Management Problem

In real-world applications, using heuristic information in optimisation has been regarded as a successful enhancement method [21] and [22]. An example in reactor fuel-management optimisation can be found in [18]. It does not guarantee that the global optimum will be reached, but the use of heuristic information aids finding a near-optimal solution in reasonable time. The key questions are how to extract it from a given problem and how to use it in optimisation algorithms.

Let us assume that there is some information, H, for each building block indicating its contribution to an objective function. The greater the contribution made by a building block, the more likely it is that it will appear in the optimal solution. How we generate the H for reactor fuel-management optimisation will be described later.

The heuristic information H can be easily used to perturb the probability model P. The modified algorithm is similar to the one described earlier in a previous section. The only difference is that, when sampling new LPs, a perturbed P^h is used instead of the original P.

$$P^h = PH^{\beta} \tag{17}$$

where H contains heuristic information and β is an exponential scalar adjusting the weight between population-based learning and heuristic information. Each row of P^h must be normalised to obtain a valid pdf. An example of using H in EDAs is illustrated in figure 10. Using this idea, any useful heuristic information, expert knowledge or even random perturbations can be used to direct the optimisation to concentrate the search on some particular area without sacrificing too much exploration. There is no requirement that H is calculated using a single heuristic measure, as we do in the examples shown here. In principle the measures used could be different on every row of H, what is important is that H does capture useful information about the relative importance of a particular item appearing in a particular position. The range that H takes depends on the measures that are used to construct H, it should be noted that each row in P^h is normalised before a solution is constructed, which means the range of H is not critical, but that the relative magnitudes of components does matter.

5 Test Case 1: A Fresh Core

This case represents the situation at the start of the reactor's life. We set up a fresh core, where 'fresh' means we assume all the fuel assemblies are brand new. By doing so, there are only five different fuel types and the problem is therefore less complicated than a case when all the fuel assemblies have different reactivity. The objective of this test case is to find the optimal LP for the CONSORT reactor for the given fuel inventory, so as to maximise the effective multiplication factor - the K_{eff}. By doing this, the life of the reactor can be extended, allowing more experiments and research to be performed. Given the 24-channel CONSORT core and the fuel inventory shown in table 2, the search space (number of possible LPs) for this case is approximately 10^{13}.

5.0.1 LP Representation

A straight forward LP representation can be established using an integer vector. For example, an integer vector form of an LP is given below:

$$LP = [1,1,1,1,1,4,5,5,1,1,1,5,5,5,4,1,5,1,5,5,1,1,1,1] \tag{18}$$

Each entry in this vector indicates a fuel type, as shown in table 2. The index of this vector indicates the fuel channel index shown in figure 8. A loading pattern and its corresponding binary matrix which is used in our methodology is shown in figure 11.

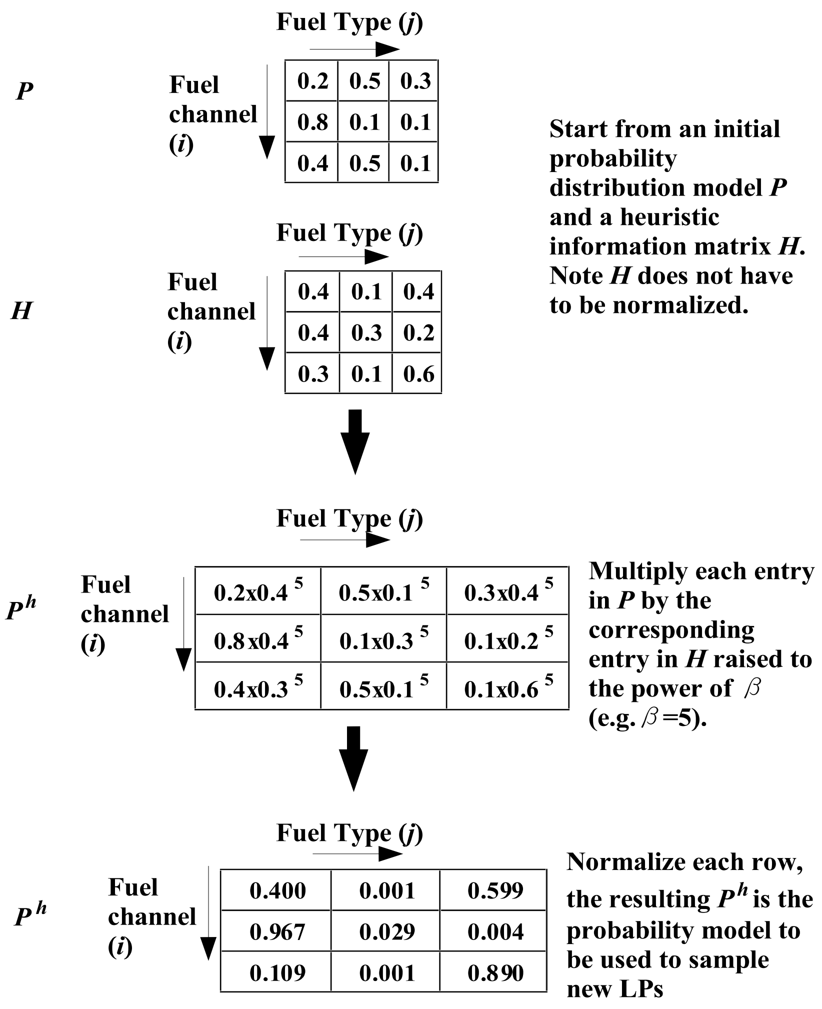
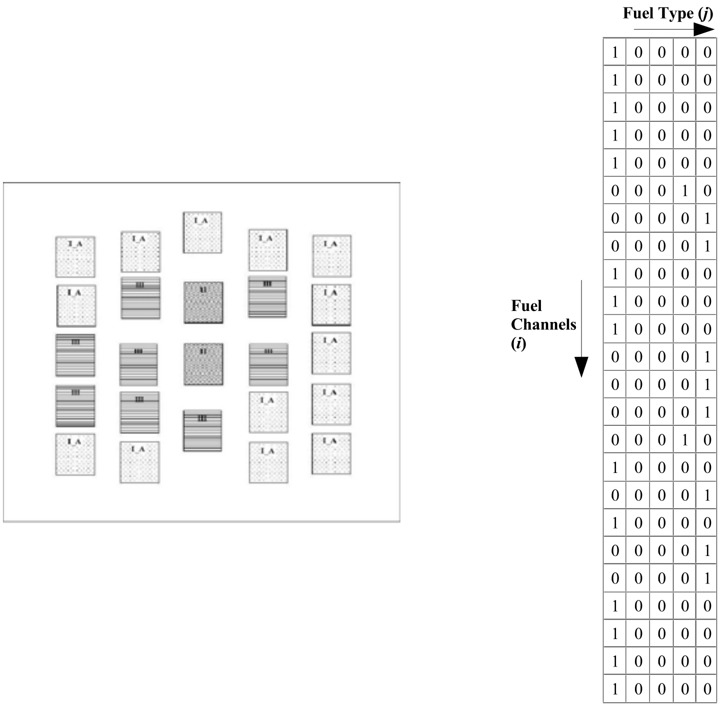

Fuel Type (j)
Fuel channel (i)
P
0.2 0.5 0.3
0.8 0.1 0.1
0.4 0.5 0.1
Start from an initial probability distribution model P and a heuristic information matrix H. Note H does not have to be normalized.
H
0.4 0.1 0.4
0.4 0.3 0.2
0.3 0.1 0.6
P^h
0.2x0.4^5 0.5x0.1^5 0.3x0.4^5
0.8x0.4^5 0.1x0.3^5 0.1x0.2^5
0.4x0.3^5 0.5x0.1^5 0.1x0.6^5
Multiply each entry in P by the corresponding entry in H raised to the power of β (e.g. β=5).
P^h
0.400 0.001 0.599
0.967 0.029 0.004
0.109 0.001 0.890
Normalize each row, the resulting P^h is the probability model to be used to sample new LPs

Fig. 10 Combining heuristic information with the probability model used in EDAs.

Table 2 The hypothetical fuel inventory of the CONSORT reactor for Test Case 1. Each fuel type is given an integer code.

Fuel Type	MARK I_A	MARK I_B	MARK I_C	MARK II	MARK III
Type Code	1	2	3	4	5
No. of Fuel	20	22	22	2	8



$$\begin{array}{|c|c|c|c|c|}
\hline
1 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 1 & 0 \\ \hline
0 & 0 & 0 & 0 & 1 \\ \hline
0 & 0 & 0 & 0 & 1 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 1 \\ \hline
0 & 0 & 0 & 0 & 1 \\ \hline
0 & 0 & 0 & 0 & 1 \\ \hline
0 & 0 & 0 & 1 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 1 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 1 \\ \hline
0 & 0 & 0 & 0 & 1 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & 0 \\ \hline
\end{array}$$

Fig. 11 An example initial LP for the CONSORT reactor and its corresponding binary matrix.

5.1 Heuristic Information

Useful heuristic data is the likelihood of a building-block being used in a promising candidate LP. In this example a building block is a particular fuel type being used in a particular fuel channel, ie an element in the binary matrix shown in figure 11. We propose to issue a 'weight' for every possible fuel-type-to-fuel-channel assignment, which will be used in our EDAs when searching for optimal LPs. This information is called the 'stand-alone K_{eff} with fuel coupling' (stand-alone K_{eff} for short). It is inspired by the technique used for ANN training in Erdogan and Geckinli's work [6], derived from the coupled reactor theory [1], and then applied in our approach for reactor loading pattern-optimisation problems.

For this particular test case, the stand-alone K_{eff} is organised in a 24×5 matrix, shown in table 3. Each entry of the matrix indicates the contribution of a particular fuel element-channel assignment to the objective function - the overall K_{eff}. Each entry (i, j) (i for channel and j for fuel type) is the calculated core K_{eff} when loading fuel j to channel i with all other channels filled with MARK I_A calculated with the reactor modelling code EVENT [7]. This is why MARK I_A has an identical value for different channels. MARK I_A is used to form a generic core state because it has the median reactivity.

Table 3 The calculated stand-alone K_{eff}s using EVENT [7] for the five different fuel types in the CONSORT reactor test case 1. The $S.D.$ is the fuel channel-wise standard deviation.

Channel No.	MARK I_A	MARK I_B	MARK I_C	MARK II	MARK III
1	1.20460	1.19972	1.19744	1.20690	1.20883
2	1.20460	1.19919	1.19654	1.20717	1.20930
3	1.20460	1.19919	1.19654	1.20717	1.20930
4	1.20460	1.20164	1.20016	1.20611	1.20735
5	1.20460	1.20164	1.20016	1.20611	1.20735
6	1.20460	1.18920	1.18184	1.21091	1.21600
7	1.20460	1.19186	1.18581	1.21004	1.21448
8	1.20460	1.19186	1.18581	1.21004	1.21448
9	1.20460	1.19833	1.19525	1.20754	1.20995
10	1.20460	1.19833	1.19525	1.20754	1.20995
11	1.20460	1.19652	1.19252	1.20831	1.21133
12	1.20460	1.19652	1.19252	1.20831	1.21133
13	1.20460	1.18736	1.17857	1.21153	1.21700
14	1.20460	1.18736	1.17857	1.21153	1.21700
15	1.20460	1.18168	1.16935	1.21312	1.21972
16	1.20460	1.19797	1.19469	1.20778	1.21038
17	1.20460	1.19797	1.19469	1.20778	1.21038
18	1.20460	1.19216	1.18589	1.20989	1.21412
19	1.20460	1.19216	1.18589	1.20989	1.21412
20	1.20460	1.19092	1.18435	1.21043	1.21515
21	1.20460	1.20156	1.20004	1.20615	1.20742
22	1.20460	1.20156	1.20004	1.20615	1.20742
23	1.20460	1.19907	1.19636	1.20728	1.20949
24	1.20460	1.19907	1.19636	1.20728	1.20949
$S.D.$	0.00000	0.00545	0.00827	0.00200	0.00356

5.2 The Algorithms

In this section we define three variants of the EDA algorithm, and a GA that is used as a benchmark.

5.2.1 EDA_S, a Simple EDA

The simple EDA is the algorithm described earlier and has the following steps,

1. Initialise a population of LPs randomly.
2. Select some good LPs according to their K_{eff} or other objective function(s).
3. Build or update the probability model using the following:

$$P_{(t+1)} = (1-\alpha)P_{(t)} + \alpha X \tag{19}$$

   where P is the probability model for core channels; t is the number of the iteration; P_0 is a uniform distribution model; α is a scalar between $[0,1]$; and X is the statistical information extracted from the selected LPs.
4. Sample new LPs from the updated probability model.
5. If stop criteria are not met, go back to (2), otherwise stop.

5.2.2 EDA_G, an EDA with an Elitism Strategy

A modified algorithm is developed by combining EDA_S with a generic elitism strategy, called EDA_G (G for generic). It is identical to EDA_S except that an elitism strategy is used to improve the exploitation of the algorithm, and therefore fast convergence and better solution quality are expected.

One of the well-known limitations of GAs is that they often fail to find a local optimum, even when the current best solutions are very close to it. This can be caused by inappropriate encoding, crossover method or random perturbations used in evolutionary operators (i.e. mutation).

An elitism strategy can be used as in [17] to improve the exploitation near the current best LP. Normally it is to ensure that the current best solution is kept at all times so that, hopefully, the crossover and mutation operators may generate more candidate solutions around the current best. The same method can be employed in EDAs but adapted to the EDA framework. The adapted elitism strategy is to perturb the probability model directly using the current best solution, as a result of which, the forthcoming search for promising LPs is explicitly biased in the direction indicated by the current best. A modified probability model updating method is

$$P_{(t+1)} = (1-\alpha)P_{(t)} + \alpha X + \eta X_b \tag{20}$$

In the additional term ηX_b, X_b is the best solution found so far and η is a scalar. This term guides the search towards the direction of the current best solution. It will keep the search close to the current location if the best is near the centroid of the population; if the best is far away from the centroid, it will accelerate the movement of the population centroid. Because of the introduction of the elitism term P needs to be normalised.

5.2.3 EDA_H, an EDA Combined with Heuristic Information

EDA_H is identical to EDA_G except that EDA_H samples a new population of LPs using the updated probability model P together with some heuristic information. The methods by which heuristic information is used in EDAs are described in an earlier section. The sampling probability is:

$$P' = PH^{\beta} \tag{21}$$

where H contains heuristic information - the stand-alone K_{eff} - and β is an exponential scalar adjusting the weight between population-based learning and heuristic information. H has an identical data structure to P. For this problem, we use the stand-alone K_{eff} with fuel coupling as H. The entry $H(i,j)$ is the calculated stand-alone K_{eff} of loading fuel j to channel i and other channels filled with MARK I_A. A larger $H(i,j)$ increases the probability of loading fuel j to channel i. An example is given in figure 10.

Similarly, in EDA_H, the probability of assigning a fuel type j to a fuel channel i depends on how often this assignment appeared in some known good LPs (positive) and its corresponding stand-alone K_{eff} with fuel coupling (positive), which can be considered a measurement of the 'contribution' that a certain fuel element makes to the objective function, the overall K_{eff}, when it is inserted into a specified channel in a generic reactor environment. The calculated stand-alone K_{eff} value for this test case is given in table 3.

To demonstrate how H affects the optimisation, let us assume that there are 10 candidate LPs, and five of them use a MARK III fuel element at channel 13. The other five LPs use MARK I_A at channel 13. After the application of heuristic information with $\beta = 4$, the probability of using a MARK III at channel 13 is higher than using a MARK I_A:

$$P^h(13, MARK\ III) = 0.5 \times 1.2170^4 \approx 1.09681 \tag{22}$$

$$P^h(13, MARK\ I\_A) = 0.5 \times 1.2046^4 \approx 1.05279 \tag{23}$$

where 1.2170 and 1.2046 are the respective stand-alone K_{eff}s. From this example we can see that in a case in which the population-based learning cannot give clear instructions, the heuristic information will lead the search. Note that P^h for each channel should be normalised after being multiplied by H.

Also, from the stand-alone K_{eff} with fuel coupling table, $H(13,5) = 1.217$ is larger than $H(22,5)$ which is 1.20742. This means that loading MARK III into channel 13 is potentially more effective than loading it into channel 22. Even though no channel dependence is considered in the EDAs presented in this work, the use of heuristic information in a 'weight-like' form PH^{β}, can compensate for this by applying the 'weight' containing the position-wise and adjacent fuel information. For channel 13, the vectors that give information about the relative probabilities for each of the five fuel types, j, are

$$P(13, j) = [0.2, 0.2, 0.2, 0.2, 0.2] \tag{24}$$

$$H^4(13,j) = [2.10557, 1.9876, 1.93057, 2.15445, 2.19362], \tag{25}$$

After normalisation, P^h and $P^h(22,j)$ are:

$$P^h(13,j) = [0.2030, 0.1906, 0.1816, 0.2077, 0.2115] \tag{26}$$
$$P^h(22,j) = [0.2004, 0.1984, 0.1974, 0.2015, 0.2023], \tag{27}$$

It can be seen that the assignment of MARK III to channel 13 is more likely than its assignment to channel 22.

5.2.4 Benchmark Genetic Algorithms

GAs are efficient and versatile algorithms for tackling complex, large-scale combinatorial optimisation problems. There are many examples of the application of GAs for reactor fuel management optimisation problems, eg Ziver[24]. For this reason, we are going to use GAs as a benchmark. Our GA-based algorithm is

1. Initialise the LP population randomly.
2. Select some LPs according to their K_{eff}.
3. Generate the new population of LPs by applying crossover and mutation to the selected individuals.
4. If stop condition is not met, go back to step 2, otherwise stop.

One of the key elements to any GAs success is its crossover operator. A study of GA crossover operators for ordering applications can be found in [17]. We have implemented the following crossover operators: 2-Point Crossover (2PX) with 1-D integer vector encoding, and the Heuristic Tie Breaking Crossover HTBX[16] with 2D permutation representation. We chose 2PX as it has been widely applied to various applications. The HTBX was specifically designed for reactor loading pattern optimisation and uses problem-dependent information. These GAs are referred to as GA_2PX, a GA with 2PX, and GA_HTBX, a GA with HTBX. Due to the difference between 2PX and HTBX, different encoding methods were used. For 2PX, the integer-vector form of LP representation is used, just as in the EDAs; for HTBX, a permutation representation is used. The fuel element ID is defined by reactivity ranking.

5.3 Results

The EDAs developed were tested extensively and compared against GAs that employed different crossover operators. A proportional selection method on ranked fitness [2] value rather than the raw value of K_{eff} is used for all the EDAs and GAs. It should be noted that this method is statistically equivalent to a two-person tournament selection scheme.

Because different encoding methods are used, the mutation operators in the EDAs and GAs are different too. In our EDAs, it is to randomly choose a fuel element and then assign it to a fuel channel that is also randomly chosen. For GAs, the mutation operator is the Swap mutation in [15], which is to randomly swap the fuel elements at two different channels.

An initial population consisting of fifty LPs is randomly generated and used as the same starting point for all the algorithms. All experiments are based on 30 independently created runs. Only one initial population has been generated and it was used for all the runs of all the algorithms.

The performance of GAs and other evolutionary algorithms can be very sensitive to the values of their control parameters. We tuned the parameters used in the GAs and EDAs carefully to ensure the validity of the comparison. The parameters are summarised in table 4.

Table 4 Parameter settings for the numerical experiments giving the population size, total number of generations, mutation and crossover rates.

Algorithms	Population Size	Max Generations	α	β	η	Mutation Rate	Crossover Rate
EDA_S	50	2000	0.001	N/A	N/A	0.05	N/A
EDA_G	50	2000	0.001	N/A	0.01	0.05	N/A
EDA_H	50	2000	0.001	30	0.01	0.05	N/A
GAs	50	2000	N/A	N/A	N/A	0.05	0.9

We recorded the best solution found in each generation, and plotted their average values from 30 independent runs. This is shown in figure 12. Experimental results after 100,000 LP evaluations are given in table 5. The maximum and minimum objective function values (error bounds), as predicted by the ANN used rather than using the EVENT software [7], found among 30 independent runs of EDA_H and GA_HTBX are recorded and plotted in figure 12. The results show that EDA_H,

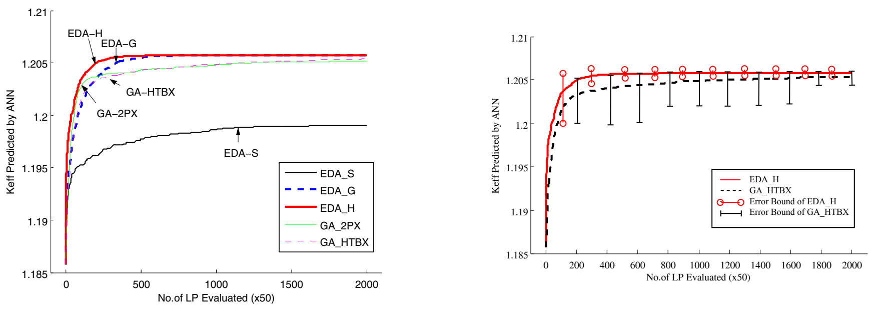


Fig. 12 Results showing the averaged optimisation process of EDAs and GAs on the CONSORT reactor case and the error bounds.

Table 5 The maximum K_{eff} found by EDAs and GAs from 30 independent runs and their corresponding average and standard deviation.

Algorithms	Best	Average	Standard Deviation
EDA_S	1.20173	1.19900	0.00108
EDA_G	1.20578	1.20572	0.00004
EDA_H	1.20578	1.20575	0.00005
GA_2PX	1.20573	1.20520	0.00041
GA_HTBX	1.20578	1.20537	0.00033

EDA_G and GA_HTBX find the same ‘optimal’ solution:

$$LP_{OPT} = 1,1,1,1,1,4,5,5,1,1,5,1,5,5,4,1,1,5,5,5,1,1,1,1 \tag{28}$$

$$K_{eff} = 1.20578 \tag{29}$$

Both EDA_G and EDA_H outperform the tested GAs in optimising the K_{eff}s. It was also found that the EDAs have smaller variations compared to GAs, which can be seen by the average, standard deviation and error-bound values from the 30 independent runs.

The better GA of the two GAs is the GA_HTBX, with a similar performance close to EDA_G. GA_HTBX and GA_2PX converge slightly faster than EDA_G initially but EDA_G outperforms them later. This is a well-known problem with GAs, they are efficient in finding a local neighbourhood containing a good solution, but they are poor local optimisers. It is not surprising that the GA_2PX performed very well and only slightly worse than HTBX at the end of the search. Considering the fuel inventory in this case, the integer vector form of LP representation in GA_2PX is a more natural and efficient encoding than the permutation form in GA_HTBX. There are in total 74 fuel elements in the fuel store, and only five different fuel types - which means that, in the permutation form of an LP, many of the entries are simply identical fuels that have been given a random ID. When a crossover or mutation operator is being executed, it could be swapping identical fuel elements. That is why GA_2PX with the integer vector LP encoding converges quickly at the beginning. This is another good example of a typical GA - good at locating a good area very quickly, but inefficient in converging to a local optimum.

EDA_G and EDA_H outperform the crossover operator HTBX in terms of convergence speed and stability. EDA_G uses neither physical information about fuel assemblies nor the reactor core’s 2D structure, but does use a ‘elitism-guided’ term to make full use of the best known LP so far. The effect of this is to improve the exploitation near the current best and therefore a better local convergence is achieved. EDA_H makes use of heuristic information, the stand-alone K_{eff} with fuel coupling, which includes the consideration of the channel position of a fuel element and the adjacent fuel elements, to improve the optimisation. It does not consider the core structure explicitly, but employs the stand-alone K_{eff} with fuel coupling to help the optimisation. So the stand-alone K_{eff} with fuel coupling is not only helpful in speeding up the convergence, but also enables the EDAs to use the reactor core channels' position and neighbourhood fuel dependence information.

In addition, the permutation representation used in GA_HTBX has to include the whole 'fuel store' including in-core and out-core fuel assemblies to ensure that the entire search space is explored, which greatly increases the problem size and computational time. The EDA approach does not need the out-of-core fuel store information, because the probabilities sampling operator allows it to search the entire search space.

6 Test Case 2: A Realistic Core Model

Test Case 2 is designed to be a more complicated case with a more realistic core state. The search space is larger and more irregular, due to more reactivity variance among fuel elements.

We demonstrate how the EDAs are adapted to this class of problems, and the results are compared to the well-established GA_HTBX algorithm. Based on the results from Test Case 1, only EDA_G, EDA_H and GA_HTBX are tested. GA_2PX is not used, as it will produce infeasible LPs in their permutation representation, as a result of which special treatment will be needed to repair them. As pointed out in Poon's work [16], GA_HTBX is one of the best candidates for this particular application, and has been regarded as the best of the available algorithms.

The optimisation task is to find the optimal loading pattern that maximises the overall K_{eff} for this modified CONSORT core, given that the fuel inventory consists of 35 fuel elements, as shown in table 6. Each fuel element is different and will have a unique ID. The core structure is the same as in Test Case 1.

The hard constraint remains the same, in that the two MARK II type fuel elements (fuel elements 10 and 11 in this case) must always be inserted into channels 6 and 15, due to safety and operational constraints. Given the core plan, the fuel inventory and the constraints, the whole search space is approximately 2×10^{29}.

Table 6 The CONSORT reactor fuel store information of the modified Test Case 2

Ranking	1	2	3	4	5	6	7
K_∞	1.68	1.68	1.68	1.68	1.67	1.67	1.67
Ranking	8	9	10	11	12	13	14
K_∞	1.67	1.67	1.61	1.60	1.54	1.54	1.53
Ranking	15	16	17	18	19	20	21
K_∞	1.53	1.53	1.53	1.53	1.53	1.52	1.52
Ranking	22	23	24	25	26	27	28
K_∞	1.52	1.52	1.52	1.52	1.52	1.52	1.52
Ranking	29	30	31	32	33	34	35
K_∞	1.52	1.52	1.52	1.52	1.51	1.13	0.77

6.1 LP Representation

Since each fuel element needs to be encoded explicitly, we use a permutation to represent the whole fuel store for this test case. The K_∞ ranking numbers are used as their unique IDs. An LP is then represented as an integer vector containing a full permutation of integers in $[1,35]$ (35 fuel elements). The first twenty-four integers indicate the in-core loading pattern, and the rest of them are not used in this loading pattern. An example is given below.

$$\begin{aligned} LP = {} & [1,2,3,4,5,10,6,7,8,9,12,13,14,15,11,16,17,18, \\ & 19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35] \end{aligned} \tag{30}$$

It should be noted that in this case, the order of out-of-core fuel elements is not relevant to the in-core LP. However, for the benchmark GA_HTBX to be able to explore the whole search space, the full permutation is required. This is because the crossover operator needs the LPs to contain all the information of the fuel inventory, so that it can regenerate valid new LPs.

In the EDAs, the LP representation is similar to the one used in Test Case 1 and is shown in figure 11. The corresponding matrix form is a 24×35 matrix. There is only one '1' in each row and each column, because each fuel element can only be used once in one fuel channel. The same 24×35 matrix structure is used for the probability model (but with real values). By using it, all the fuel elements will be considered when filling a fuel channel and no information will be lost.

6.2 Heuristic Information

To generate the stand-alone K_{eff}, we insert a fuel element j in channel i, filling all other channels with a 'generic' fuel m. An LP is then created and examined by the simulation software EVENT [7] to obtain its K_{eff}. This result is recorded as the stand-alone K_{eff} with fuel-coupling information for fuel j in channel i. This calculation is repeated for all the fuel elements and all channels, and the results can be presented in a 24×35 matrix, this is equivalent to that shown in figure 3. Each entry $[i, j]$ of this matrix represents the 'spatial contribution' of assigning fuel j to channel i in a more realistic context.

This set of data is too large to be shown in this paper, but it is available elsewhere[13]. The use of the stand-alone K_{eff} remains the same as in Test Case 1, as given in the following equation:

$$P' = PH^\beta \tag{31}$$

where P' is the probability model used to sample new candidate solutions, P is the probability model updated by the EDA algorithm, H is the stand-alone K_{eff} and β is an exponential scalar. The data structures of the probability model P and H are identical to each other as well as to the binary matrix form of an LP.

The stand-alone K_{eff} with fuel coupling can be used directly as H. If the variance of different assignments of fuel elements to channels is too small even when a large β has been used, our suggestion is to use the ranked stand-alone K_{eff} matrix instead of the original raw data. The total $24 \times 35 = 840$ entries in the original stand-alone K_{eff} matrix are ranked from 1 to 840, which represents the relative contribution of all the possible assignments of fuel elements to fuel channels. Unacceptable assignments (e.g. assigning fuel element 10 to any channel other than channel 5) should not be included in this ranking as it will disturb the effect of the stand-alone K_{eff}. The ranking stretches the differences between the fuel element-channel assignments so that a much smaller range of β can be used. In this case, $\beta = 2$ is sufficient.

6.3 Results

The parameters used are summarised in table 7. It should be noted that the β value in EDA_H is smaller in this case because we used the ranked stand-alone K_{eff} instead of the raw value. The reason for this is that the variance among fuel elements is much smaller compared to Test Case 1.

Table 7 The well-tuned parameters settings used in EDAs and GAs for Test Case 2

Algorithms	Population Size	Max Generations	α	β	η	Mutation Rate	Crossover Rate
EDA_G	50	2000	0.001	N/A	0.01	0.05	N/A
EDA_H	50	2000	0.001	2	0.01	0.05	N/A
GA_HTBX	50	2000	N/A	N/A	N/A	0.05	0.9

Numerical results after 100,000 LP evaluations are given in table 8. For both EDAs and the GA, the best solution found in each generation was recorded, and their average values from 30 independent runs are illustrated in figure 13. Figure 13 shows the maximum and minimum objective function values (error bounds) found among 30 independent runs, using the same initial starting population, of EDA_H and GA_HTBX. It is found that EDA_G and EDA_H algorithms both found better solutions than GA_HTBX, as well as better averaged best solutions over 30 independent runs. The standard deviations also suggest that both EDAs converge faster than GA_HTBX. The experimental results show that EDA_H and EDA_G outperformed the benchmark GA_HTBX in terms of solution quality, convergence speed and stability.

The fast convergence of EDA_G and EDA_H may cause a premature problem, as they can become stuck at some local optima. GA_HTBX, due to more randomised noise being used, suffers less from the local convergence problem. In order to resolve this for EDAs, parameter tuning is necessary.

Table 8 The maximum K_{eff} found by EDAs and GAs from 30 independent runs and their corresponding average and standard deviation

Algorithms	Best	Average	Standard Deviation
EDA_G	1.007480	1.007459	0.000039
EDA_H	1.007480	1.007477	0.000016
GA_HTBX	1.007400	1.007151	0.000221

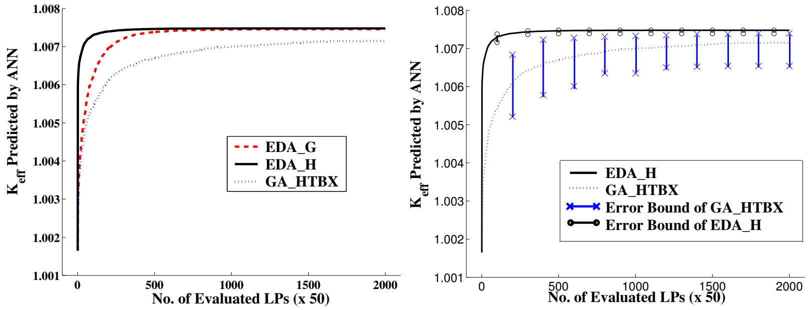


Fig. 13 The averaged performance of EDAs and GAs against number of LP evaluations of the modified CONSORT case and the error bounds.

7 Test Case 3: A Realistic Core with a Constraint

Test Case 3 is designed to be an even more complicated case based on the core model used in Test Case 2. A power peaking constraint was added to the previous objective function. The algorithms to be tested are EDA_G, EDA_H and GA_HTBX. The main purpose of this section is to demonstrate the application of EDAs to LP optimisation with a typical constraint, and to observe their performance.

Because the CONSORT reactor is a low-power research reactor, it does not generate excessive energy in the core. In reality, the Power Peaking Factor (PPF) is not a realistic safety concern.

We use the PPF as a penalty term in the objective function as a way of dealing with the constraints. The experiments are performed with a simple weighted sum of the K_{eff} and PPF. The optimisation task is then to maximise an objective function that consists of the core K_{eff} and the PPF constraint as a penalty term, which is:

$$F = w1K_{eff} + w2PPF \tag{32}$$

where $w1$ and $w2$ are scalars adjusting the weights between the objective term and the constraint term and PPF is the estimated power peaking factor given a loaded core. In this work, we used neutron flux density to replace the actual power in order to simplify the PPF calculation. It is calculated using the equation below:

$$PPF = \frac{\text{The Maximum Thermal Flux Density of all Fuel Channels}}{\text{Averaged Thermal Flux Density of all the Fuel Channels}} \tag{33}$$

In this test case, maximising F may not yield the optimal K_{eff} given the PPF constraint, because the two weights, $w1$ and $w2$, have a significant impact on the location of the optima. In this case, $w1$ will be set to 0.8, while $w2$ will be -0.2. In practice, these two parameters should be adjusted to suit real-world applications.

The core structure and the hard constraint are both the same, as in Test Case 2. The size of the search space remains the same, and is approximately 2×10^{29}. However, in order to distinguish different fuel elements better, the fuel inventory has been modified to increase the reactivity difference between different fuel elements, which is summarised in table 9. The problem caused by spreading the reactivity evenly is that the variation of the PPF will be relatively small.

Table 9 The modified CONSORT reactor fuel inventory for Test Case 3

Ranking	1	2	3	4	5	6	7
K_∞	1.80	1.79	1.78	1.77	1.76	1.75	1.74
Ranking	8	9	10	11	12	13	14
K_∞	1.73	1.72	1.71	1.70	1.69	1.68	1.67
Ranking	15	16	17	18	19	20	21
K_∞	1.66	1.65	1.64	1.63	1.62	1.61	1.60
Ranking	22	23	24	25	26	27	28
K_∞	1.59	1.58	1.57	1.56	1.55	1.54	1.53
Ranking	29	30	31	32	33	34	35
K_∞	1.52	1.51	1.50	1.49	1.48	1.47	1.46

7.1 LP Representation and Heuristic Information

In this test case an LP is represented in the same way as in Test Case 2.

As in the previous test cases, the heuristic information will be an estimate of the contribution of the building blocks (the assignments of fuel elements to fuel channels) to the objective function. Since there are two components in the objective function, the heuristic information for this test case will need to contain the information of both K_{eff} and PPF. We calculate one set of the stand-alone K_{eff}, which is a 24×35 matrix, as in the previous test cases. In addition, another matrix with an identical structure to the stand-alone K_{eff} is built, which can be called the stand-alone PPF.

To calculate the stand-alone PPF, an LP with a certain fuel element, j, inserted into one of the 24 fuel channels, i, and other fuel channels filled with a generic fuel elements, 'm', is set up and fed to an EVENT simulation [7]. The estimated stand-alone PPF of this LP can be calculated using equation 34, which is different to equation 33. This stand-alone PPF is intended to be a measure of the power generated by fuel element j in fuel channel i compared to the averaged power of all the fuel channels. Repeating this process for all the fuel elements and all fuel channels, a 24×35 matrix can be filled, just like the stand-alone K_{eff} is built. Both the stand-alone K_{eff} and PPF are available elsewhere[13].

$$PPF = \frac{\text{The Thermal Flux Density of the Chosen Fuel Channel } i}{\text{Averaged Thermal Flux Density of all the Fuel Channels}} \tag{34}$$

The stand-alone K_{eff} and PPF are combined in the same way as they are used in the objective function:

$$H = w1K_{eff} + w2PPF \tag{35}$$

where H is the heuristic information and $w1$ and $w2$ are weight scalars. In this case, they will be set to exactly the same values as in the objective functions, i.e. $w1 = 0.8$ and $w2 = -0.2$. This H matrix contains the 'spatial' contribution of each fuel element when it is assigned to a specific fuel channel.

7.2 Results

EDA_G, EDA_H and GA_HTBX were each tested 30 times. In each case the algorithms generate an initial population randomly at the beginning of the run. This ensures a better exploration for all the tested algorithms. We used the same parameter settings as in Test Case 2, summarised in table 7. The core model is the same and only the fuel inventory has changed.

The objective function changed due to the inclusion of the PPF term and the introduction of the two weights $w1$ and $w2$. To ensure a better understanding of this test case, we first tested the algorithms with only one of the two terms enabled.

The first scenario is to enable the K_{eff} term only, which is done by setting $w1$ to 1 and $w2$ to 0. This test case then turns into a K_{eff} maximisation problem identical to the previous test cases. The heuristic information is also modified to exclude the stand-alone PPF. The results are summarised in table 10. The results of maximising K_{eff} show a very similar pattern compared to the previous test cases. EDAs find better results than the tested GA. In addition, the standard deviations of 30 independent EDA runs are much smaller than for GA_HTBX. The averaged performances from 30 independent runs are plotted in figure 14.

The second scenario is to enable the PPF term only, which is done by setting $w1$ to 0 and $w2$ to -1. This test case then turns into a PPF minimisation problem. The heuristic information is also modified to exclude the stand-alone K_{eff}. The test results are summarised in table 10. The results of minimising PPF only for Test Case 3 show that both EDAs find smaller PPF values than the tested GA_HTBX. The standard deviation from 30 independent runs also suggests that the tested EDAs are more robust and less disruptive search methods than the tested GAs. The averaged performances from 30 independent runs on both scenarios are plotted in figure 14.

It can be seen that the heuristic information used in EDA_H, the stand-alone PPF, did not help EDA_H outperform EDA_G. The averaged best solution found

Table 10 Maximisation of K_{eff} and minimising PPF only in Test Case 3 by EDAs and GAs. Results are from 30 independent runs, showing the best, average and standard deviation. The corresponding PPF when the best K_{eff} is found (w1 = 1), and the K_{eff} when the best PPF is found (w2 = -1), are listed in the last column.

Maximising K_{eff} only in Test Case 3

Algorithms	Best	Average	Standard Deviation	PPF
EDA_G	1.047680	1.047672	0.000008	1.187150
EDA_H	1.047680	1.047674	0.000007	1.187150
GA_HTBX	1.047270	1.046737	0.000378	1.189550

Minimising PPF only in Test Case 3

Algorithms	Best	Average	Standard Deviation	K_{eff}
EDA_G	1.149250	1.149255	0.000007	1.017840
EDA_H	1.149250	1.149260	0.000006	1.017840
GA_HTBX	1.149660	1.150320	0.000551	1.025590

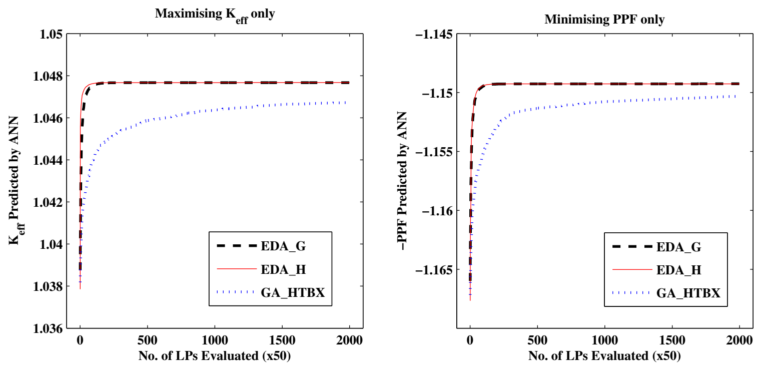


Fig. 14 The averaged performance comparison between EDA_G, EDA_H and EDA_HTBX on Test Case 3, with one objective at a time - Maximising K_{eff} (left) and minimising PPF (right).

in 30 independent runs with random initialisation by EDA_H is slightly worse than for EDA_G. It is understood from previous test cases that EDA_H concentrates on an area indicated by the heuristic information, while EDA_G performs a better exploration. This local convergence feature in EDA_H, however, may not be desirable in some cases, such as minimisation of PPF, because two radically different LPs could have very similar PPF values.

From the two preliminary tests, both EDAs perform well in maximising K_{eff}, and EDA_H shows less variation between different independent runs. On minimising PPF, both EDAs managed to find the best solution. The heuristic information used in EDA_H did not improve the averaged performance.

By combining K_{eff} maximisation and PPF minimisation together using weights $w1 = 0.8$ and $w2 = -0.2$, a more complicated test problem is created. The EDAs are doing well in maximising K_{eff}, but in minimising PPF, the concern is that EDA_H's strong local search feature may restrict its exploration, and the best solutions found may consequently lack diversity.

The experimental results summarised in table 11 show a similar pattern compared to the results from Test Case 2. Both EDAs outperformed GA_HTBX in 100,000 function evaluations in terms of solution quality and convergence speed. This comparison shows that the tested EDAs are as good as, if not better than, the tested GA_HTBX in global search, given a reasonably large number of function evaluations. Figure 15 shows the averaged performance of 30 independent runs of EDAs and GA_HTBX and figure 15 compares the error bounds between EDA_H and GA_HTBX. It is clear that EDA_H converged much faster than the tested GA, and found better solutions.

Table 11 The best objective function value found by EDAs and GAs from 30 independent runs and their corresponding average and standard deviation, the K_{eff} and PPF are also shown.

Algorithms	Best	Average	Standard Deviation	K_{eff}	PPF
EDA_G	0.605366	0.605291	0.000080	1.046260	1.158210
EDA_H	0.605366	0.605293	0.000079	1.046260	1.158210
GA_HTBX	0.605058	0.604047	0.000578	1.046460	1.160550

Table 12 The maximum and minimum K_{eff} and PPF values found by EDAs and GAs from 30 independent runs on Test Case 3 with the combined objective function F - $w1 = 0.8$ and $w2 = -0.2$.

The range of searched K_{eff}:

Algorithms	Maximum	Minimum
EDA_G	1.046600	1.007610
EDA_H	1.046650	1.007780
GA_HTBX	1.046460	1.005450

The range of searched PPF:

Algorithms	Maximum	Minimum
EDA_G	1.196800	1.152800
EDA_H	1.194560	1.152980
GA_HTBX	1.201160	1.152330

In this case, EDA_G and EDA_H produce very similar results. The heuristic information, H, utilised in EDA_H did not have a great impact on its performance, in terms of solution quality and convergence speed. There is a trade-off between the objective function and the constraint. A better local convergence, as in EDA_H, may not contribute significantly to exploring the trade-off.

From the K_{eff} and PPF values shown in table 12, it can be seen that GA_HTBX is exploring a considerably wider range of objective/constraint function space, while both EDAs focus on rather limited ranges, particularly EDA_H, which is restricted by the heuristic information. If this heuristic information does not improve the optimisation significantly, it might be switched off or possibly replaced by an alternative set of heuristic information.

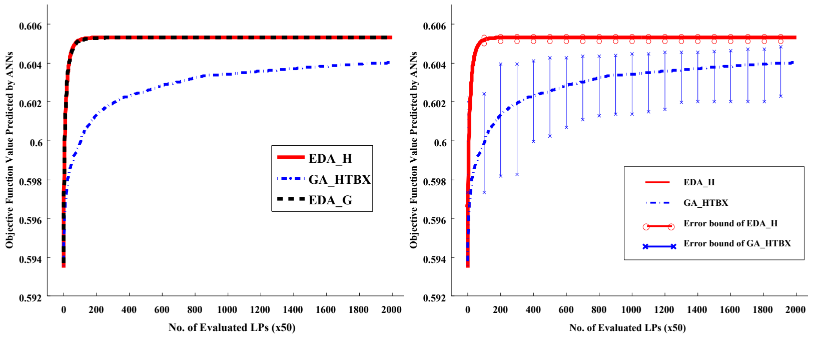


Fig. 15 The comparison between EDA_H and GA_HTBX on the modified CONSORT case on the modified CONSORT case with power peaking constraint and the error bounds.

8 Conclusions

We have illustrated how the Estimation of Distribution Algorithm can be applied to an allocation problem. The basic principles were demonstrated on the Travelling Salesman Problem, we also discussed how heuristic information can be incorporated into the algorithm. We then applied the algorithm to the nuclear fuel loading pattern problem. On three examples we compared the performance of two variants of the EDA and a GA. The GA has been regarded as the best algorithm available for the nuclear loading pattern problem. Both EDAs perform better than the GA. On the more realistic tests the EDA finds better solutions and finds them faster than the GA. There is little to choose between the EDAs in terms of the final value found. However the use of the heuristic information does give faster convergence. The one measure for which the GA out performs the EDAs is the range of objective function values tested. We interpret this as being a wider search in parameter space. In problems with a complex search space this may be an advantage.

References

1. Avery, R.: Theory of coupled reactors. In: 2nd UN Conference on Peaceful Uses of Atomic Energy, Geneva (1958)
2. Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Proceedings of the first International Conference on Genetic Algorithms and their Applications, pp. 101–111 (1985)
3. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. CMU-CS-94-163. Carnegie Mellon University, Ottawa (1994)
4. de Jong, K.A., Spears, W.M.: Using genetic algorithms to solve p-complete problems. In: Proc. of Third Int. Conf. Genetic Algorithms and their Applications, pp. 124–132. Morgan Kaufmann, San Francisco (1989)
5. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to traveling salesman problem. IEEE Trans on Evolutionary Computations 1(1), 53–66 (1997)
6. Erdogan, A., Geckinli, M.: A PWR reload optimisation code (XCORE) using artificial neural networks and genetic algorithms. Annals of Nuclear Energy 30, 35–53 (2003)
7. EVENT, http://amcg.ese.ic.ac.uk/index.php? title=EVENT Applied Modelling and Computation Group (AMCG), Imperial College
8. Fox, B.R., McMahon, M.B.: Genetic operators for sequencing problems. In: Foundations of Genetic Algorithms, pp. 284–300. Morgan Kaufmann, San Francisco (1990)
9. Franklin, S.J., Goddard, A.J.H., O'Connell, J.S.: Research reactor facilities and recent developments at Imperial College, London. In: Research Reactor Fuel Management 1998: European Nuclear Society (1998)
10. Franklin, S.J., Gardner, D., Mumford, J., Lea, R., Knight, J.: Business operations and decommissioning strategy for Imperial College London research reactor 'CONSORT' - a financial risk management approach. In: Research Reactor Fuel Management 2005, European Nuclear Society (2005)
11. Grefenstette, J., Gopal, R., Rosmaita, B., Van Gucht, D.: Genetic algorithms for the traveling salesman problem. In: Proceedings of the first International Conference on Genetic Algorithms and their Applications, p. 160 (1985)
12. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE Transactions on Evolutionary Computation 3, 287–297 (1999)
13. Jiang, S.: Nuclear Fuel Management Optimisation Using Estimation of Distribution Algorithms. PhD thesis, Department of Earth Science and Engineering, Imperial College London (2009)
14. Muhlenbein, H., Paab, G.: From recombination of genes to the estimation of distributions i. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
15. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A study of permutation crossover operators on the travelling salesman problem. In: Proceedings of the 2nd International Conference of Genetic Algorithms and Their Applications, pp. 224–230 (1987)
16. Poon, P.W.: The Genetic Algorithm Applied to PWR Reload Core Design. PhD thesis, Department of Engineering, University of Cambridge (1992)
17. Poon, P.W., Carter, J.N.: Genetic algorithm crossover operators for ordering applications. Computers and Operations Research 22(1), 135–147 (1995)
18. Poon, P.W., Parks, G.T.: Application of genetic algorithms to in-core nuclear fuel management optimization. In: Proc. Joint Int. Conf. Mathematical Methods and Supercomputing in Nuclear Applications, vol. 2, pp. 777–786 (1993)

19. van Geemert, R., Quist, A.J., Hoogenboom, J.E., Gibcus, H.P.M.: Research reactor in-core fuel management optimisation by application of multiple cyclic interchange algorithms. Nuclear Engineering and Design 186, 369–377 (1998)
20. Whitley, D., Starkweather, T., Fuquay, D.: Scheduling problems and the traveling salesman: the genetic edge recombination operator. In: Proceedings of the Third International Conference on Genetic Algorithms, Arlington, VA, pp. 116–121 (1989)
21. Zhang, Q.: CC483 Evolutionary Computation: Lecture notes, Department of Computer Science. University of Essex (2003)
22. Zhang, Q., Sun, J., Tsang, E., Ford, J.: Hybrid estimation of distribution algorithm for global optimisation. Engineering Computations 21(1), 91–107 (2004)
23. Zhang, Q., Zhou, A., Jin, Y.: RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm. IEEE Trans. on Evolutionary Computation 12(1), 41–63 (2008)
24. Ziver, A.K., Pain, C.C., Carter, J.N., et al.: Genetic algorithms and artificial neural networks for loading pattern optimisation of advanced gas-cooled reactors. Annals of Nuclear Energy 31, 431–457 (2004); ISSN: 0306-4549

Optimal Control Systems with Reduced Parametric Sensitivity Based on Particle Swarm Optimization and Simulated Annealing

Radu-Emil Precup*, Radu-Codruţ David, Stefan Preitl, Emil M. Petriu, and József K. Tar

Abstract. This chapter discusses theoretical and design aspects for optimal control systems with a reduced parametric sensitivity using Particle Swarm Optimization (PSO) and Simulated Annealing (SA) algorithms. Sensitivity models with respect to the parametric variations of the controlled process are derived and the optimal control problems are defined. The new objective functions in these optimization problems are integral quadratic performance indices that depend on the control error and squared output sensitivity functions. Different dynamic regimes are considered. Relatively simple PSO and SA optimization algorithms are developed for the minimization of the objective functions, which optimize the control system responses and reduce the sensitivity to parametric variations of the controlled process. Examples of optimization problems encountered in the design of optimal proportional-integral (PI) controllers for a class of second-order processes with integral component are used to validate the proposed methods.

Radu-Emil Precup · Radu-Codruţ David · Stefan Preitl
Department of Automation and Applied Informatics, "Politehnica" University of Timisoara, Bd. V. Parvan 2, RO-300223 Timisoara, Romania
e-mail: radu.precup@aut.upt.ro

Emil M. Petriu
School of Information Technology and Engineering, University of Ottawa, 800 King Edward, Ottawa, ON, K1N 6N5 Canada

József K. Tar
Institute of Intelligent Engineering Systems, Óbuda University, Bécsi út 96/B, H-1034 Budapest, Hungary

* Corresponding author.
Tel.: +40-2564032-26, -29, Fax: +40-2564032-14

1 Introduction

Many control systems are tuned based on idealized linear or linearized models of the controlled processes. However, industrial processes are subjected to parametric variations of the controlled processes, which result in models that are either nonlinear or only locally linearized around several nominal operating points or trajectories. Therefore, it is necessary to do a sensitivity analysis with respect to the parametric variations of the controlled process.

As shown in [1], the uncontrollable process parametric variations lead to undesirable behavior of the control systems. As reported in the literature, the parametric sensitivity of the control systems can be studied in the frequency domain [2–7] or in the time domain [1,8].

A variety of optimal control applications employing sensitivity models in the objective functions were reported in the literature [9–12]. Objective functions as extended quadratic performance indices were used in the design of Takagi-Sugeno proportional-integral-fuzzy controllers [8,13]. An application of Bellman-Zadeh's approach to decision making in fuzzy environments for multi-criteria optimization problems is presented in [14]. The elimination of the steady-state control error by an augmented state feedback tracking guaranteed cost control is reported in [15]. In a recent review paper [16] Campos and Calado present optimal control approaches to human arm movement control. A method to estimate the minimum variance bounds and the achievable variance bounds for the assessment of the Iterative Learning Control-based batch control systems is presented in [17].

The optimal control methods with reduced parametric sensitivity presented in this chapter use time domain sensitivity models in the objective functions. Building upon Precup's and Preitl's previous work on controller optimization criteria, [8], we propose new objective functions that employ the integrals of weighted squared output sensitivity functions added to the Integral of Squared Error (ISE), the Integral of Absolute Error (IAE), the Integral of Time multiplied by Squared Error (ITSE), and the Integral of Time multiplied by Absolute Error (ITAE) to reduce the effects of the parametric disturbances.

The dynamic regimes with regard to step-type modifications of the reference input and disturbance inputs are considered resulting in additional sensitivity models and objective functions. While the fundamental deviation of a control system relative to its nominal trajectory is described by the control error, the additional deviation can be described by the output sensitivity function in the sensitivity model.

Solving the optimization problems for the usually non-convex objective functions used in many control systems is not a trivial task as it can lead to several local minima. Different solutions such as derivative-free optimization algorithms [18–21], Particle Swarm Optimization (PSO) [22,23] and Simulated Annealing (SA) [24–27] were proposed in the literature to solve these problems.

PSO algorithms have a number of advantages, which make them attractive for the control systems design:

- compact implementation programs,
- computational efficiency,

- search algorithm using objective function values instead of the gradient information,
- they are not bound by conventional deterministic methods constraints such as the linearity, differentiability, convexity, separability or non-existence of constraints,
- very little, if any, solution dependence on the initial states of particles.

In our recent paper [28] we discussed two new PSO algorithms for the optimal design of proportional-integral (PI) controllers for a class of second-order processes with integral component and variable parameters. Other examples of PSO-based designs of robust, adaptive and predictive controllers are presented in [29–32]. Optimal fuzzy controllers and combinations of PSO and fuzzy controllers are presented in [33–35]. Other PSO applications to the design of optimal controllers for power systems, transportation systems, electrical drives and artificial intelligence are presented in [36–40].

SA algorithms have the distinct capability of finding the global minimum of certain objective functions under specific conditions. Several applications of SA for the adaptive and predictive optimal control are discussed in [41–43]. Intelligent SA-based control systems are presented in [44] and [45], and applications to the control of electrical drives and chemical processes are reported in [46–48].

This chapter will discuss new PSO and SA-based algorithms for the design of optimal control systems with reduced parametric sensitivity. The design of an optimal PI controller for a class of second-order processes with integral component [49] will then be presented as a representative case study.

The chapter is structured as follows. Section 2 provides a mathematical framework for the design of optimal control systems with reduced parametric sensitivity. Models of the controlled process and controller, sensitivity models and objective functions are among the most important topics that are discussed. Section 3 focuses on PSO and SA algorithms for the optimization of the controllers with objective functions that have a single variable, β. Useful recommendations are provided for practitioners on how to set the values of the parameters for PSO and SA algorithms. Section 4 is dedicated to the representative case study of optimal designs of PI controllers for a class of second-order processes with integral component. Conclusions and further discussions are presented in Section 5.

2 Framework for Optimal Control Systems Design

The controlled process is represented by the following Single Input-Single Output (SISO) state-space model

$$\begin{aligned}
\dot{\mathbf{x}}_P &= \mathbf{f}_P(\mathbf{x}_P, \boldsymbol{\alpha}, u, d), \\
y &= g_P(\mathbf{x}_P, \boldsymbol{\alpha}, d), \\
\mathbf{x}_P(t_0) &= \mathbf{x}_{P,0},
\end{aligned} \tag{1}$$

where: $t_0 \geq 0$ is the initial time moment, $\mathbf{x}_P = [x_{P,1} \quad x_{P,2} \quad \dots \quad x_{P,n}]^T \in R^n$ is the state vector of the controlled process, $\mathbf{x}_{P,0} \in R^n$ is the initial state vector of the controlled process, u is the control signal, d is the disturbance input, y is the controlled output, $\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_m]^T \in R^m$ is the parameter vector containing the parameters of the controlled process, $\alpha_j, j = \overline{1,m}$, $\mathbf{f}_P : R^{n \times m \times 1 \times 1} \to R^n$, $g_P : R^{n \times m \times 1} \to R$ are functions that are differentiable with respect to $\boldsymbol{\alpha}$ on R^m, T indicates the matrix transposition, and the real argument of the functions, t, $t \geq t_0$, is omitted to simplify the presentation.

The state-space model (1) includes the dynamics of the measuring element(s) as well as of the actuator(s).

Fig. 1 shows the structure of a conventional control system, where: C is the controller, P is the controlled process, r is the reference input, and e is the control error,

$$e = r - y. \tag{2}$$

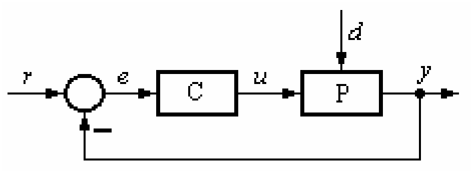


Fig. 1 Control system structure.

The controller is represented by the SISO state-space model

$$\begin{aligned} \dot{\mathbf{x}}_C &= \mathbf{f}_C(\mathbf{x}_C, \beta, e), \\ u &= g_C(\mathbf{x}_C, \beta, e), \\ \mathbf{x}_C(t_0) &= \mathbf{x}_{C,0}, \end{aligned} \tag{3}$$

where: $\mathbf{x}_C = [x_{C,1} \quad x_{C,2} \quad \dots \quad x_{C,p}]^T \in R^p$ is the state vector of the controller, $\mathbf{x}_{C,0} \in R^p$ is the initial state vector of the controller, β is the design parameter and $\mathbf{f}_C : R^{p \times q \times 1} \to R^n$, $g_C : R^{p \times q \times 1} \to R$ are continuous functions.

The majority of linear controllers [50] and many nonlinear controllers including the fuzzy controllers under certain conditions [51] can be expressed in terms of the model (3). However, the convergence of the integrals in the objective functions requires that all controllers should have an integral component in order to ensure the zero steady state of the control error for several disturbance inputs.

To express the state-space model of the control system, the elements of the two state vectors in the models (1) and (3) are grouped in the state vector $\mathbf{x}$ of the control system

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_P \\ \mathbf{x}_C \end{bmatrix} = [x_1 \quad x_2 \quad \dots \quad x_{n+p}]^T \in R^{n+p},$$
$$x_i = \begin{cases} x_{P,i} & \text{if } i = \overline{1,n} \\ x_{C,i-n} & \text{otherwise} \end{cases}, \ i = \overline{1, n+p}. \tag{4}$$

Next, the state-space models of the controlled process and controller in the models (1) and (3) are merged using equation (2) and the structure presented in Fig. 1. Therefore, the state-space model of the control system becomes

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} \mathbf{f}_P\{\mathbf{x}_P, \boldsymbol{\alpha}, g_C[\mathbf{x}_C, \beta, r - g_P(\mathbf{x}_P, \boldsymbol{\alpha}, d)], d\} \\ \mathbf{f}_C[\mathbf{x}_C, \beta, r - g_P(\mathbf{x}_P, \boldsymbol{\alpha}, d)] \end{bmatrix} \\ &= \mathbf{f}(\mathbf{x}, \boldsymbol{\alpha}, \beta, r, d) = [f_1 \quad f_2 \quad \dots \quad f_{n+p}]^T \in R^{n+p}, \\ y &= g_P(\mathbf{x}_P, \boldsymbol{\alpha}, d), \\ \mathbf{x}(t_0) &= \begin{bmatrix} \mathbf{x}_{P,0} \\ \mathbf{x}_{C,0} \end{bmatrix}. \end{aligned} \tag{5}$$

Considering the parameter of the controlled process $\alpha_j, j = \overline{1,m}$, the state sensitivity functions $\lambda_i^{\alpha_j}, i = \overline{1, n+p}$, and the output sensitivity function σ^{α_j} are defined according to

$$\lambda_i^{\alpha_j} = \left[\frac{\partial x_i}{\partial \alpha_j}\right]_{\alpha_j,0}, \ \sigma^{\alpha_j} = \left[\frac{\partial y}{\partial \alpha_j}\right]_{\alpha_j,0}, \ i = \overline{1, n+p}, \ j = \overline{1,m}, \tag{6}$$

where the subscript 0 indicates the nominal value of the appropriate parameter.

Using equations (6) to calculate the partial derivatives in the model (5) we obtain sensitivity models of the control system with respect to $\alpha_j, j = \overline{1,m}$, for the constant reference input r_0 and disturbance input d_0:

$$\begin{aligned} \dot{\lambda}_i^{\alpha_j} &= \sum_{k=1}^{n+p} \left[\frac{\partial f_i}{\partial x_k}\right]_{\alpha_j,0} \lambda_k^{\alpha_j} + \left[\frac{\partial f_i}{\partial \alpha_j}\right]_{\alpha_j,0}, \\ \sigma^{\alpha_j} &= \sum_{k=1}^{n} \left[\frac{\partial g_P}{\partial x_k}\right]_{\alpha_j,0} \lambda_k^{\alpha_j} + \left[\frac{\partial g_P}{\partial \alpha_j}\right]_{\alpha_j,0}, \\ \lambda_i^{\alpha_j}(t_0) &= 0, \ i = \overline{1, n+p}, \ j = \overline{1,m}. \end{aligned} \tag{7}$$

The initial state variables are important in the analysis of the sensitivity models (7).

In order to obtain good dynamics of the control systems and reduced sensitivity we define the following objective functions with the design parameter β as an independent variable:

- the extended ISE:

$$I_{ISE}^{\alpha_j}(\beta)=\int_0^{\infty}\{e^2(t)+(\gamma^{\alpha_j})^2[\sigma^{\alpha_j}(t)]^2\}dt,\ j=\overline{1,m}\,, \tag{8}$$

- the extended IAE:

$$I_{IAE}^{\alpha_j}(\beta)=\int_0^{\infty}\{|\,e(t)\,|+(\gamma^{\alpha_j})^2[\sigma^{\alpha_j}(t)]^2\}dt,\ j=\overline{1,m}\,, \tag{9}$$

- the extended ITSE:

$$I_{ITSE}^{\alpha_j}(\beta)=\int_0^{\infty}\{te^2(t)+(\gamma^{\alpha_j})^2[\sigma^{\alpha_j}(t)]^2\}dt,\ j=\overline{1,m}\,, \tag{10}$$

- the extended ITAE:

$$I_{ITAE}^{\alpha_j}(\beta)=\int_0^{\infty}\{t\,|\,e(t)\,|+(\gamma^{\alpha_j})^2[\sigma^{\alpha_j}(t)]^2\}dt,\ j=\overline{1,m}\,, \tag{11}$$

where γ^{α_j}, $j=\overline{1,m}$, are the weighting parameters. The dynamic regimes with regard to step-type modifications of the reference input and disturbance inputs are considered; therefore, the number of objective functions in (8)–(11) is doubled.

Making use of the objective functions (8)–(11) the optimization problems which ensure the optimal design of the controllers are defined as follows:

$$\begin{aligned}&\beta^* =\arg\min_{\beta\in Do} I_{ISE}^{\alpha_j}(\beta),\ \beta^* =\arg\min_{\beta\in Do} I_{IAE}^{\alpha_j}(\beta),\\&\beta^* =\arg\min_{\beta\in Do} I_{ITSE}^{\alpha_j}(\beta),\ \beta^* =\arg\min_{\beta\in Do} I_{ITAE}^{\alpha_j}(\beta),\ j=\overline{1,m},\end{aligned} \tag{12}$$

where β^* is the optimal value of the variable β, and *Do* is the feasible domain of the variable β.

When setting the domain *Do*, we should first take in consideration the stability of the control system. Other inequality-type constraints can be imposed for the optimization problems defined in equation (12). For example, they can concern the actuator saturation [52], the robust stability of the control system or the controller robustness [53].

3 Particle Swarm Optimization and Simulated Annealing Algorithms

Particle Swarm Optimization

Particle Swarm Optimization (PSO) was originally designed and introduced by Eberhart and Kennedy [22,23]. PSO is a search algorithm inspired by the social behavior of birds, bees or schools of fishes. PSO uses the swarm intelligence concept, modeled by particles (agents) with specific positions and velocities, which interact locally with their environment to create coherent global functional patterns.

Social concepts like evolution, comparison and imitations of other individuals are typically associated with intelligent agents that interact in order to adapt to the environment and develop optimal patterns of behavior. Mutual learning allows individuals to behave in a similar way and to acquire adaptive patterns of behavior. The swarm intelligence is based on the following principles [54]:

- *proximity*, i.e. the population should be able to carry out simple time and space calculations,
- *quality*, i.e. the population should be able to respond to quality factors in the environment,
- *diverse response*, i.e. the population should not commit its activity to excessively long narrow channels,
- *stability*, i.e. the population should not change its behavior every time the environment changes,
- *adaptability*, i.e. the population should be able to change its behavior when it is worth the computational price.

PSO algorithm is an evolutionary algorithm, which similarly with the genetic algorithms starts with a random generation of candidate solutions and then searches for the optimal solution. In the case of PSO algorithm, the individual particles are updated in parallel, a new solutions depends on the previous one and on the solutions corresponding to its neighboring particles. The same rules apply to all updates. The particles are moving in a D-dimensional search space search space $\Re^D$ with randomly chosen velocities and positions, knowing their best values so far and the positions in the search space $\Re^D$. The position and velocity of each particle in the search space are updated at each step of the iteration. The velocity of each particle is adjusted according to its own previous moving history as well as to that of the other particles [28].

A swarm particle can be represented by two D-dimensional vectors, $X_i = [x_{i1} \quad x_{i2} \quad \dots \quad x_{iD}]^T \in \Re^D$ standing for the particle position and the particle velocity $V_i = [v_{i1} \quad v_{i2} \quad \dots \quad v_{iD}]^T$. Let $P_{i,Best} = [p_{i1} \quad p_{i2} \quad \dots \quad p_{iD}]^T$ be the best position of a specific particle and $P_{g,Best} = [p_{g1} \quad p_{g2} \quad \dots \quad p_{gD}]^T$ be the best position of the swarm.

The particle velocity and position updating rules can be expressed in terms of the state-space form [55] as follows:

$$V_i^{k+1} = wV_i^k + c_1 r_1 (P_{g,Best} - X_i^k) + c_2 r_2 (P_{i,Best} - X_i^k), \tag{13}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1}, \tag{14}$$

where: r_1, r_2 are random variables with a uniform distribution between 0 and 1, $i,\ i = \overline{1,n}$, is the index of the current particle in the swarm, n is the number of particles in the swarm, $k,\ k = \overline{1, j_{\max}}$, is the index of the current iteration, $j_{\max}$ is the maximum number of iterations. The parameter w in equation (13) stands for the inertia weight, which shows the effect of the previous velocity vector on the new one. $V_{\max}$ is the upper limit placed on the velocity in all the dimensions preventing the particle from moving too rapidly in the search space. The constants c_1 and c_2 represent the weighting factors of the stochastic accelerations pulling the particles towards their final positions. Adopting too low values of these weights will allow particles to roam far from the target regions before being tugged back. On the other hand, too high values will result in abrupt movements towards, or overshooting, the target regions.

The individual particles within the swarm communicate and learn from each other, and based on this they move to improve their previous position relative to their neighbors. Different neighborhood topologies can emerge on the basis of the communication strategy of the particles within the swarm. A star-type topology is created in the majority of cases. In that topology, each particle can communicate with every other individual forming a fully connected social network, so that each particle could access the overall best position.

PSO algorithm consists of the following steps [22,23,54,55]:

1. Initialize the swarm placing particles at random positions inside the d-dimensional search space,
2. Evaluate the fitness of each particle using its current position,
3. Compare the performance of each individual to its best performance so far,
4. Compare the performance of each particle to the best global performance,
5. Change the velocity of each particle according to equation (13),
6. Move each particle to its new position according to equation (14),
7. Go to step 2, until the maximum number of iterations is reached.

The flowchart of PSO algorithm is presented in Fig. 2 (a).

Simulated Annealing

Simulated Annealing (SA) is a random-search technique based on an analogy with the well-known annealing process used in metallurgy, consisting in a heat treatment that alters the microstructure of metal causing changes in properties such as strength and hardness and ductility. The final properties of the metal are very much dependent on the heating and cooling process. If the temperature cools too quickly, the final product will be brittle. If the temperature cools down slowly, the resulting product will have the right hardness and ductility.

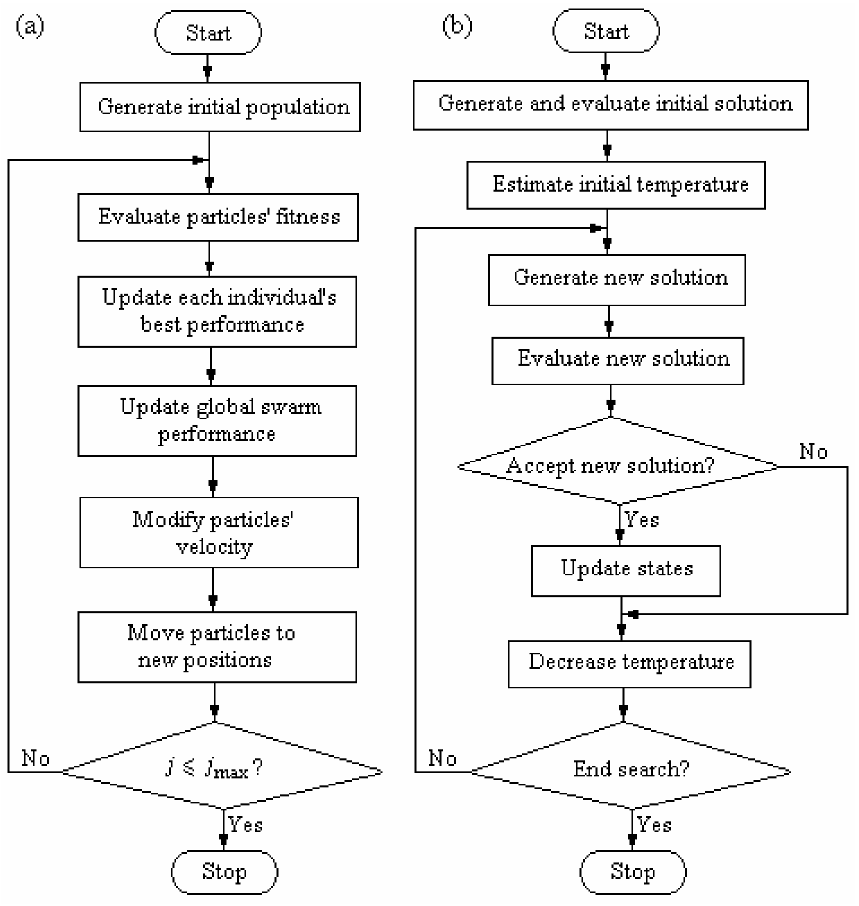

(a)
Start
Generate initial population
Evaluate particles' fitness
Update each individual's best performance
Update global swarm performance
Modify particles' velocity
Move particles to new positions
$j \leqslant j_{\max}$?
No
Yes
Stop
(b)
Start
Generate and evaluate initial solution
Estimate initial temperature
Generate new solution
Evaluate new solution
Accept new solution?
No
Yes
Update states
Decrease temperature
End search?
No
Yes
Stop

Fig. 2 Flowchart of PSO algorithm (a) and of SA algorithm (b).

SA algorithms were originally developed [24] to deal with highly nonlinear problems. It approaches the minimization problem similarly to the way a ball is rolling from valley to valley down a hill slope until it finally reaches the lowest possible (minimum altitude) location. If the ball does not have enough energy, it cannot roll high enough and it becomes trapped in a valley somewhere higher on the hill slope, above the lowest possible position [27]. The decision making on a particle staying in, or rolling off, a valley is based on a probabilistic energy framework.

SA algorithms start with a high temperature and an initial solution. Considering the initial vector solution φ with the corresponding fitness value $C(\varphi)$ of the fitness function C, the next probable vector solution ψ is chosen from the vicinity of φ, and it will have the fitness value $C(\psi)$.

SA algorithms contain a probabilistic-based framework for solution acceptance. Defining

$$\Delta C_{\varphi\psi} = C(\varphi) - C(\psi), \tag{15}$$

the probability of φ being the next solution, referred to as p_{ψ}, is

$$p_{\psi} = \begin{cases} 1 & \text{if } \Delta C_{\varphi\psi} \leq 0, \\ \exp(-\Delta C_{\varphi\psi} / \theta) & \text{otherwise.} \end{cases} \tag{16}$$

where θ is the current temperature value of the algorithm.

If $p_{\psi} > r_n$, where r_n is a randomly selected number, $0 \leq r_n \leq 1$, then ψ will be the new solution. Otherwise a new solution must be generated. As it can be observed in this framework, there is a valid probability of replacing the current solution with a higher cost solution.

The above process repeats for a predetermined number of steps, and the temperature is next reduced. The algorithms end when the temperature value is so low that it does not allow any modification of the fitness function, and the last value is the solution.

SA algorithms implemented here to solve the optimization problems defined in equation (12) can be formulated in terms of the following steps:

- *Step 1*. Set $m = 0$ and the minimum temperature $\theta_{\min}$. Choose the initial temperature θ_m.
- *Step 2*. Generate the initial solution φ and calculate its corresponding fitness value $C(\varphi)$.
- *Step 3*. Generate a probable solution ψ by perturbing φ and evaluate the fitness value $C(\psi)$.
- *Step 4*. Calculate $\Delta C_{\varphi\psi}$ making use of equation (15). If $\Delta C_{\varphi\psi} \leq 0$ then ψ is the new solution. Else select randomly r_n, $0 \leq r_n \leq 1$, and calculate p_{ψ} by means of the function defined in equation (16). If $p_{\psi} > r_n$ then ψ is the new solution.
- *Step 5*. Reduce the temperature according to the temperature decrement rule

$$\theta_{m+1} = f_{cs}(\theta_m), \tag{17}$$

where f_{cs} is the cooling schedule.

- *Step 6*. If $\theta_m > \theta_{\min}$ then go to step 3, else stop.

The subscript *m* in SA algorithms stands for the iteration index.

The flowchart of an SA algorithm is illustrated in Fig. 2 (b).

The most common temperature decrement rule is the well-accepted exponential cooling schedule

$$\theta_{m+1} = \alpha_{cs}\theta_m, \tag{18}$$

where $\alpha_{cs} = \text{const}$, $\alpha_{cs} < 1$, $\alpha_{cs} \approx 1$.

The fitness functions C implemented in our SA algorithms to solve the optimization problems defined in equation (12) are the objective functions defined in equations (8)–(11). With this regard the vector arguments $\boldsymbol{\varphi}$ or $\boldsymbol{\psi}$ of these fitness functions are replaced by the scalar parameter β.

SA algorithms are very versatile as they do not depend on any restrictive properties of a model [24–27]. In addition they can be used in combination with other gradient-based algorithms due to their flexibility and ability to approach the global optimum.

4 Case Study

As shown in [50] the PI controllers can be tuned by the Extended Symmetrical Optimum (ESO) method to guarantee a good compromise to the desired / imposed control performance indices making use of a single design parameter referred to as β. This design parameter ensures the generalization of Kessler's Symmetrical Optimum (SO) method [56,57] to obtain performance enhancements.

This case study will show how PSO and SA algorithms combined with the ESO method can be used for an efficient tuning of a PI controller for a class of second-order processes with integral component [49]. The ESO method allows simplifying the implementation of PSO and SA algorithms.

The class of second-order processes with integral component considered here as case study is a particular controlled process (P) in the framework of the control system structure presented in Fig. 1. The accepted class of controlled processes can be modeled by the state-space model as that presented in equation (1) or by the transfer function $P(s)$

$$P(s) = Y(s)/U(s) = k_P/[s(1+sT_\Sigma)], \tag{19}$$

where $Y(s)$ is the Laplace transform of y, $U(s)$ is the Laplace transform of u, zero initial conditions are assumed, s is the complex argument specific to the Laplace transform, k_P is the controlled process gain, and T_Σ is the small time constant or the sum of small time constants of the process. The sum of small time constants is used when the transfer function defined in equation (19) is a simplified model of higher order processes.

This process is used as a representative benchmark for many servo systems applications where equation (19) can be viewed as a convenient simplified linearized mathematical model. Thus the two parameters are variable and the sensitivity analysis and design of controllers with reduced sensitivity is of interest, and the parameter vector of the controlled process is ($m = 2$)

$$\boldsymbol{\alpha} = [\alpha_1 = k_P \quad \alpha_2 = T_\Sigma]^T \in R^2 . \tag{20}$$

The controlled process consists of the two blocks with the transfer functions presented in Fig. 3. The integral component of the process, with the transfer function $1/s$, is illustrated in equation (19) and in Fig. 3 because of the integration property of the Laplace transform.

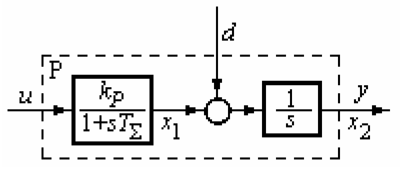


Fig. 3 Process structure.

The state-space model of this controlled process is the following particular expression of the state-space model (1):

$$\begin{aligned} &\dot{x}_1(t) = x_2(t) + d(t), \\ &\dot{x}_2(t) = -(1/T_\Sigma)x_2(t) + (k_P/T_\Sigma)u(t), \\ &y(t) = x_1(t), \end{aligned} \tag{21}$$

where x_1 and x_2 are the state variables shown in Fig. 3.

The performance specifications imposed to the control systems concern first the reference input *r* tracking and the regulation in the presence of disturbance inputs (*d*). These performance specifications are expressed in terms of maximum values of the control system performance indices; the control systems design should ensure the fulfillment of the constraints resulted from these maximum values, i.e. to ensure smaller performance indices with respect to the maximum imposed ones. It is difficult to meet generally all performance specifications.

Fig. 4 exemplifies the definitions of three performance indices with respect to the step modification of *r*: σ_1 – the overshoot, t_r – the 10 % to 90 % rise time, and t_s – the 2 % settling time. The following notations are used in Fig. 4: y_0 – the initial value of *y*, y_f – the final value of *y*, $y_{10} = 0.1|y_f - y_0|$, $y_{90} = 0.9|y_f - y_0|$, $y_{98} = 0.98|y_f - y_0|$, and $y_{102} = 1.02|y_f - y_0|$.

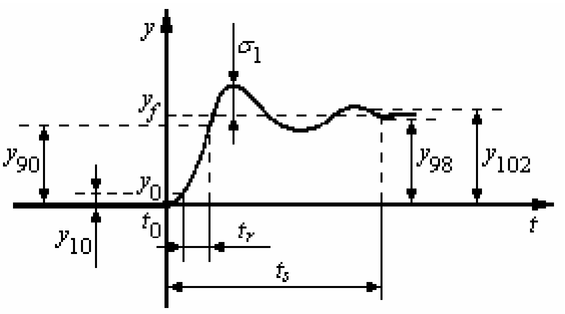


Fig. 4 Control system performance indices defined with respect to the step modification of *r*.

Good control system performance indices can be obtained if the process is controlled by means of the control system structure presented in Fig. 1, where C is a PI controller. The transfer function of the PI controller is *C*(s)

$$C(s) = U(s)/E(s) = k_c(1+sT_i)/s = k_C[1+1/(sT_i)],\ k_C = T_i k_c, \tag{22}$$

where $U(s)$ is the Laplace transform of *u*, $E(s)$ is the Laplace transform of *e*, zero initial conditions are assumed, and the tuning parameters of the PI controller are k_C (k_c) – the gain, and T_i – the integral time constant.

The choice of the single design parameter β specific to the ESO method within the domain

$$Do = \{\beta \mid 1 < \beta < 20\}, \tag{23}$$

guarantees a compromise between the control system performance indices as shown in Fig. 5.

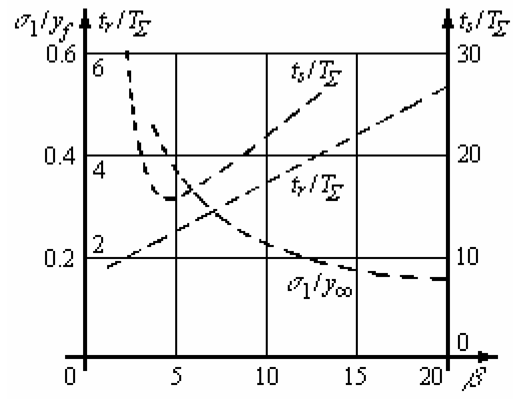


Fig. 5 Control system performance indices with respect to *r* versus β.

The ESO-based PI tuning conditions give the tuning parameters of the PI controller:

$$k_c = 1/(\beta\sqrt{\beta}\, k_P T_\Sigma^2),\ T_i = \beta T_\Sigma,\ k_C = 1/(\sqrt{\beta}\, k_P T_\Sigma), \tag{24}$$

where the parameter β is chosen such that so set the performance indices (Fig. 5) in order to fulfill the performance specifications, and $\beta = 4$ corresponds to Kessler's SO method.

The control system performance indices can be improved viz. alleviated by filtering the reference input in terms of introducing the reference filter F in the control system structure. This results in the two-degree-of-freedom control system structure presented in Fig. 6, where $\tilde{r}$ is the filtered reference input. A simple ESO-based reference filter is characterized by the transfer function $F(s)$

$$F(s) = \tilde{R}(s)/R(s) = 1/(1 + s\,T_i), \tag{25}$$

where $\tilde{R}(s)$ is the Laplace transform of $\tilde{r}$, $R(s)$ is the Laplace transform of r, and zero initial conditions are assumed.

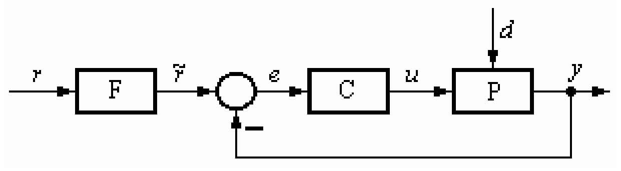


Fig. 6 Two-degree-of-freedom control system structure with reference filter.

The values of the design parameter β will be obtained as follows as solutions to the optimization problems (12).1

Accepting that x_3 is the output of the integral component in the parallel structure of the PI controller (Fig. 7), the state-space model of this controller is

$$\begin{aligned} \dot{x}_3(t) &= (1/T_i)e(t), \\ u(t) &= k_C(x_3(t) + e(t)). \end{aligned} \tag{26}$$

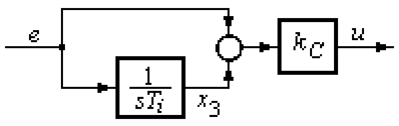


Fig. 7 Parallel structure of the PI controller.

In order to derive the sensitivity models with respect to the parametric variations of the controlled process, we have to tune the linear PI controller in terms of equations (24) for the nominal values of the process parameters k_{P0} and $T_{\Sigma 0}$. The state-space model of the PI controller becomes then

$$\begin{aligned} \dot{x}_3(t) &= [1/(\beta T_{\Sigma 0})]e(t), \\ u(t) &= [1/(\sqrt{\beta}\, k_{P0} T_{\Sigma 0})](x_3(t) + e(t)). \end{aligned} \tag{27}$$

Merging the models (25) and (27), the state-space model of the control system is

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) + d(t), \\ \dot{x}_2(t) &= -[k_P /(\sqrt{\beta}\, k_{P0} T_{\Sigma 0} T_\Sigma)]x_1(t) - (1/T_\Sigma)x_2(t) \\ &\quad + [k_P /(\sqrt{\beta}\, k_{P0} T_{\Sigma 0} T_\Sigma)]x_3(t) + [k_P /(\sqrt{\beta}\, k_{P0} T_{\Sigma 0} T_\Sigma)]r(t), \\ \dot{x}_3(t) &= -[1/(\beta T_{\Sigma 0})]x_1(t) + [1/(\beta T_{\Sigma 0})]r(t), \\ y(t) &= x_1(t). \end{aligned} \tag{28}$$

Using formulas (7) in equation (28) we obtain the sensitivity model with respect to k_P

$$\begin{aligned} \dot{\lambda}_1^{k_P}(t) &= \lambda_2^{k_P}(t), \\ \dot{\lambda}_2^{k_P}(t) &= -[1/(\sqrt{\beta}\, T_{\Sigma 0}^2)]\lambda_1^{k_P}(t) - (1/T_{\Sigma 0})\lambda_2^{k_P}(t) \\ &\quad + [1/(\sqrt{\beta}\, T_{\Sigma 0}^2)]\lambda_3^{k_P}(t) - [1/(\sqrt{\beta}\, k_{P0} T_{\Sigma 0}^2)]x_{10}(t) \\ &\quad + [1/(\sqrt{\beta}\, k_{P0} T_{\Sigma 0}^2)]x_{30}(t) + [1/(\sqrt{\beta}\, k_{P0} T_{\Sigma 0}^2)]r_0(t), \\ \dot{\lambda}_3^{k_P}(t) &= -[1/(\beta T_{\Sigma 0})]\lambda_1^{k_P}(t), \\ \sigma^{k_P}(t) &= \lambda_1^{k_P}(t), \end{aligned} \tag{29}$$

and the sensitivity model with respect to T_Σ

$$\begin{aligned} \dot{\lambda}_1^{T_\Sigma}(t) &= \lambda_2^{T_\Sigma}(t), \\ \dot{\lambda}_2^{T_\Sigma}(t) &= -[1/(\sqrt{\beta}\, T_{\Sigma 0}^2)]\lambda_1^{T_\Sigma}(t) - (1/T_{\Sigma 0})\lambda_2^{T_\Sigma}(t) + [1/(\sqrt{\beta}\, T_{\Sigma 0}^2)]\lambda_3^{T_\Sigma}(t) \\ &\quad + [1/(\sqrt{\beta}\, T_{\Sigma 0}^3)]x_{10}(t) + (1/T_{\Sigma 0}^2)x_{20}(t) \\ &\quad - [1/(\sqrt{\beta}\, T_{\Sigma 0}^3)]x_{30}(t) - [1/(\sqrt{\beta}\, T_{\Sigma 0}^3)]r_0(t), \\ \dot{\lambda}_3^{T_\Sigma}(t) &= -[1/(\beta T_{\Sigma 0})]\lambda_1^{T_\Sigma}(t), \\ \sigma^{T_\Sigma}(t) &= \lambda_1^{T_\Sigma}(t). \end{aligned} \tag{30}$$

Eight optimization problems (12) for the given application, corresponding to the dynamic regimes characterized by the unit step modification of the reference input, are solved as follows by means of PSO and SA algorithms. The other eight-optimization problems out of the 16 possible optimization problems, corresponding to the dynamic regimes characterized by modifications of the disturbance input, are not analyzed here because of the following reasons:

- the controller designs are generally done with respect to the reference input,
- the sensitivity models (29) and (30) do not depend on the disturbance input and the disturbance input affects only the nominal behavior of the control system,
- the controllers have integral character that ensure the disturbance rejection.

However, the behavior of the optimized control systems with respect to unit step modifications of the disturbance input is analyzed in order to outline the disturbance rejection. The effects of the weighting parameters on the solutions are analyzed in all optimization problems.

The analysis of PSO algorithm involves the effects of the number of particles in the swarm, the weightings of the stochastic acceleration terms that pull each particle towards their end positions and the maximum number of iterations of the algorithm. Besides, the effects of linear cooling schedule, number of steps for each perturbation of the solution and the acceptance / rejection rates are analyzed for SA algorithm.

The PSO algorithm described in the previous section was implemented in Matlab in order to validate the proposed PI controller design method for the control of the process with the transfer function (19) and the parameters $k_P = 1$ and $T_\Sigma = 1\,\text{s}$.

For the sake of simplicity the fitness functions corresponding to each objective function (12) are represented by the generic names I^{k_P} and I^{T_Σ}. They were used to evaluate the population and calculate the local best $P_{i,Best}$ and global best $P_{g,Best}$. The fitness functions were calculated by repeated simulations of the control systems' behavior with respect to the unit step modification of the reference input accepting the presence of the reference filter described by formula (25).

The optimization problems derived from equations (12) are reduced to finding the optimal value β^* of the design parameter β, thus reducing the solution search space to $D = 1$. The values of the weighting parameters used in the minimization of I^{k_P} were chosen to belong to the set $(\gamma^{k_P})^2 \in \{0, 0.1, 1, 10\}$. The values of the parameters in the corresponding PSO algorithm were set to $n = 10, c_1 = c_2 = 1.2$. The analysis of the effect of the maximum number of iterations on the optimal values of the controller tuning parameters and minimum values of the objective functions is illustrated in Tables 1 to 4.

Table 1 Results of the analysis of the effects of the maximum number of iterations on the optimal controller parameters and minimum value of objective function in case of $I_{ISE}^{k_P}$

$(\gamma^{k_P})^2$	$j_{\max}$	β^*	k_C^*	T_i^*	$I_{ISE}^{k_P}$
0	30	31.7219	0.1776	31.7219	0.2466
0	50	48.7477	0.1432	48.7477	0.2438
0	100	53.0471	0.1373	53.0471	0.2432
0	200	106.7760	0.0968	106.7760	0.2241
0	500	92.8131	0.1038	92.8131	0.2303
0.1	30	5.1401	0.4411	5.1401	0.5507
0.1	50	5.1388	0.4411	5.1388	0.5507
0.1	100	5.1388	0.4411	5.1388	0.5507
1	30	2.6509	0.6142	2.6509	2.0253
1	50	2.6511	0.6142	2.6511	2.0253
1	100	2.6512	0.6142	2.6512	2.0253
10	30	2.1286	0.6854	2.1286	15.0580
10	50	2.1322	0.6848	2.1322	15.0579
10	100	2.1322	0.6848	2.1322	15.0579

Table 2 Results of the analysis of the effects of the maximum number of iterations on the optimal controller parameters and minimum value of objective function in case of $I_{IAE}^{k_P}$

$(\gamma^{k_P})^2$	$j_{\max}$	β^*	k_C^*	T_i^*	$I_{IAE}^{k_P}$
0	30	3.8982	0.5065	3.8982	2.1032
0	50	3.8981	0.5065	3.8981	2.1032
0	100	3.8981	0.5065	3.8981	2.1032
0	200	3.9027	0.5062	3.9027	2.1031
0	500	3.9027	0.5062	3.9027	2.1031
0.1	30	3.8231	0.5114	3.8231	2.2828
0.1	50	3.8252	0.5113	3.8252	2.2828
0.1	100	3.8252	0.5113	3.8252	2.2828
1	30	3.2637	0.5535	3.2637	3.8144
1	50	3.2309	0.5563	3.2309	3.8139
1	100	3.2272	0.5567	3.2272	3.8138
10	30	2.2880	0.6611	2.2880	17.2018
10	50	2.2861	0.6614	2.2861	17.2018
10	100	2.2861	0.6614	2.2861	17.2018

Table 3 Results of the analysis of the effects of the maximum number of iterations on the optimal controller parameters and minimum value of objective function in case of $I_{ITSE}^{k_P}$

$(\gamma^{k_P})^2$	$j_{\max}$	β^*	k_C^*	T_i^*	$I_{ITSE}^{k_P}$
0	30	4.5597	0.4683	4.5597	1.8650
0	50	4.5596	0.4683	4.5596	1.8650
0	100	4.5595	0.4683	4.5595	1.8650
0	200	4.5595	0.4683	4.5595	1.8650
0.1	30	4.3409	0.4800	4.3409	2.0606
0.1	50	4.3409	0.4800	4.3409	2.0606
1	30	3.4282	0.5401	3.4282	3.6606
1	50	3.4281	0.5401	3.4281	3.6606
1	100	3.4281	0.5401	3.4281	3.6606
10	30	2.4135	0.6437	2.4135	17.3087
10	50	2.4135	0.6437	2.4135	17.3087

Table 4 Results of the analysis of the effects of the maximum number of iterations on the optimal controller parameters and minimum value of objective function in case of $I_{ITAE}^{k_P}$

$(\gamma^{k_P})^2$	$j_{\max}$	β^*	k_C^*	T_i^*	$I_{ITAE}^{k_P}$
0	30	3.5658	0.5296	3.5658	13.0250
0	50	3.5658	0.5296	3.5658	13.0250
0.1	30	3.5630	0.5298	3.5630	13.1970
0.1	50	3.5658	0.5296	3.5658	13.1968
0.1	100	3.5658	0.5296	3.5658	13.1968
1	30	3.5732	0.5290	3.5732	14.7451
1	50	3.5158	0.5333	3.5158	14.7382
1	100	3.5158	0.5333	3.5158	14.7382
10	30	3.1316	0.5651	3.1316	29.6065
10	50	3.3519	0.5462	3.3519	29.7806
10	100	3.1148	0.5666	3.1148	29.6057

A similar analysis was done for the family objective functions I^{T_Σ} accepting the weighting parameter $(\gamma^{T_\Sigma})^2 \in \{0,0.1,1,10\}$ and the same parameters in PSO algorithm. The results are illustrated in Tables 5 to 8.

Table 5 Results of the analysis of the effects of the maximum number of iterations on the optimal controller parameters and minimum value of objective function in case of $I_{ISE}^{T_\Sigma}$

$(\gamma^{T_\Sigma})^2$	$j_{\max}$	β^*	k_C^*	T_i^*	$I_{ISE}^{T_\Sigma}$
0	30	32.0494	0.1766	32.0494	0.2465
0	50	57.1700	0.1323	57.1700	0.2426
0	100	45.2117	0.1487	45.2117	0.2443
0	200	72.4913	0.1175	72.4913	0.2386
0	500	144.4070	0.0832	144.4070	0.2118
0.1	30	3.6023	0.5269	3.6023	0.6885
0.1	50	3.6036	0.5268	3.6036	0.6885
0.1	100	3.6036	0.5268	3.6036	0.6885
1	30	2.9529	0.5819	2.9529	2.9854
1	50	2.9531	0.5819	2.9531	2.9854
1	100	2.9530	0.5819	2.9530	2.9854
10	30	2.8696	0.5903	2.8696	25.5593
10	50	2.8689	0.5904	2.8689	25.5593
10	100	2.8689	0.5904	2.8689	25.5593

Table 6 Results of the analysis of the effects of the maximum number of iterations on the optimal controller parameters and minimum value of objective function in case of $I_{IAE}^{T_\Sigma}$

$(\gamma^{T_\Sigma})^2$	$j_{\max}$	β^*	k_C^*	T_i^*	$I_{IAE}^{T_\Sigma}$
0	30	3.9031	0.5062	3.9031	2.1031
0	50	3.9027	0.5062	3.9027	2.1031
0	100	3.9027	0.5062	3.9027	2.1031
0.1	30	3.6591	0.5228	3.6591	2.3995
0.1	50	3.6607	0.5227	3.6607	2.3995
0.1	100	3.6607	0.5227	3.6607	2.3995
1	30	3.0690	0.5708	3.0690	4.7623
1	50	3.0887	0.5690	3.0887	4.7619
1	100	3.0887	0.5690	3.0887	4.7619
10	30	2.8860	0.5886	2.8860	27.3707
10	50	2.8860	0.5886	2.8860	27.3707

Table 7 Results of the analysis of the effects of the maximum number of iterations on the optimal controller parameters and minimum value of objective function in case of $I_{ITSE}^{T_\Sigma}$

$(\gamma^{T_\Sigma})^2$	$j_{\max}$	β^*	k_C^*	T_i^*	$I_{ITSE}^{T_\Sigma}$
0	30	4.5602	0.4683	4.5602	1.8650
0	50	4.5595	0.4683	4.5595	1.8650
0.1	30	4.0134	0.4992	4.0134	2.2035
0.1	50	3.9491	0.5032	3.9491	2.2029
0.1	100	3.9480	0.5033	3.9480	2.2029
1	30	3.1716	0.5615	3.1716	4.6304
1	50	3.1703	0.5616	3.1703	4.6304
1	100	3.1704	0.5616	3.1704	4.6304
10	30	2.8994	0.5873	2.8994	27.2770
10	50	2.8994	0.5873	2.8994	27.2770

Table 8 Results of the analysis of the effects of the maximum number of iterations on the optimal controller parameters and minimum value of objective function in case of $I_{ITAE}^{T_\Sigma}$

$(\gamma^{T_\Sigma})^2$	$j_{\max}$	β^*	k_C^*	T_i^*	$I_{ITAE}^{T_\Sigma}$
0	30	3.6162	0.5259	3.6162	13.0291
0	50	3.5658	0.5296	3.5658	13.0250
0	100	3.5658	0.5296	3.5658	13.0250
0.1	30	3.5631	0.5298	3.5631	13.3043
0.1	50	3.5641	0.5297	3.5641	13.3043
0.1	100	3.5641	0.5297	3.5641	13.3043
1	30	3.4401	0.5392	3.4401	15.7773
1	50	3.4450	0.5388	3.4450	15.7773
1	100	3.4389	0.5392	3.4389	15.7772
10	30	3.0613	0.5715	3.0613	39.0402
10	50	3.0620	0.5715	3.0620	39.0401
10	100	3.0620	0.5715	3.0620	39.0401

Similar results were obtained for different combinations of the parameters regarding PSO algorithm for example $n \in \{10, 11, ..., 30\}$, $0.3 \le c_1 < 1.2$, and $0.3 \le c_2 < 1.2$. Therefore, the analysis of the effects of the parameters n, c_1, c_2 proves that starting with a small value of $j_{\max}$ the same value of the optimal solution is obtained no matter the values of n, c_1, c_2 taken into consideration.

Numerous experiments done here with the implemented PSO algorithms prove that their sensitivity with respect to the initial conditions associated to equations (13) and (14) is relatively small. The considered case study assisted by PSO algorithms shows that the results exhibit reduced sensitivity with respect to the initial conditions of the sensitivity models.

The same generic families of fitness functions I^{k_P} and I^{T_Σ} highlighting the objective functions (12) were used for SA algorithm implemented in Matlab as well.

In order to get a higher rate of convergence we used the following linear cooling schedule

$$T^{K+1} = 0.9T^K , \tag{31}$$

where T is the temperature and K is the current iteration index.

A more abrupt cooling schedule could generate similar results, but at the expense of a higher probability of being trapped into a local minimum. A maximum number of 300 iterations at each temperature level, with a success rate of 50 and a rejection rate of 1000, allowed obtaining a maximum convergence rate and a low probability to be trapped in a local minimum.

An identical set of weighting parameters, $(\gamma^{k_P})^2 \in \{0,0.1,1,10\}$ and $(\gamma^{T_\Sigma})^2 \in \{0,0.1,1,10\}$, was used for both families of objective functions I^{k_P} and I^{T_Σ}.

The optimal controller parameters and minimum objective functions for the family of objective functions I^{k_P} are presented in Tables 9 to 12.

Table 9 Optimal controller parameters that minimize $I_{ISE}^{k_P}$

$(\gamma^{k_P})^2$	β^*	k_C^*	T_i^*	$I_{ISE}^{k_P}$
0	36.8668	0.1647	36.8668	0.2455
0.1	5.1357	0.4413	5.1357	0.5507
1	2.6519	0.6141	2.6519	2.0253
10	2.1319	0.6849	2.1319	15.0579

Table 10 Optimal controller parameters that minimize $I_{IAE}^{k_P}$

$(\gamma^{k_P})^2$	β^*	k_C^*	T_i^*	$I_{IAE}^{k_P}$
0	3.9030	0.5062	3.9030	2.1031
0.1	3.8254	0.5113	3.8254	2.2828
1	3.2271	0.5567	3.2271	3.8138
10	2.2862	0.6614	2.2862	17.2018

Table 11 Optimal controller parameters that minimize $I_{ITSE}^{k_P}$

$(\gamma^{k_P})^2$	β^*	k_C^*	T_i^*	$I_{ITSE}^{k_P}$
0	4.5606	0.4683	4.5606	1.8650
0.1	4.3406	0.4800	4.3406	2.0606
1	3.4273	0.5402	3.4273	3.6606
10	2.4134	0.6437	2.4134	17.3087

Table 12 Optimal controller parameters that minimize $I_{ITAE}^{k_P}$

$(\gamma^{k_P})^2$	β^*	k_C^*	T_i^*	$I_{ITAE}^{k_P}$
0	3.5727	0.5291	3.5727	13.0251
0.1	3.5658	0.5296	3.5658	13.1968
1	3.5024	0.5343	3.5024	14.7393
10	3.1148	0.5666	3.1148	29.6057

A similar analysis was performed for the family of objective functions I^{T_Σ} leading to the results shown in Tables 13 to 16.

Table 13 Optimal controller parameters that minimize $I_{ISE}^{T_\Sigma}$

$(\gamma^{T_\Sigma})^2$	β^*	k_C^*	T_i^*	$I_{ISE}^{T_\Sigma}$
0	37.0825	0.1642	37.0825	0.2454
0.1	3.6026	0.5269	3.6026	0.6885
1	2.9536	0.5819	2.9536	2.9854
10	2.8687	0.5904	2.8687	25.5593

Table 14 Optimal controller parameters that minimize $I_{IAE}^{T_\Sigma}$

$(\gamma^{T_\Sigma})^2$	β^*	k_C^*	T_i^*	$I_{IAE}^{T_\Sigma}$
0	3.9027	0.5062	3.9027	2.1031
0.1	3.6620	0.5226	3.6620	2.3995
1	3.0887	0.5690	3.0887	4.7619
10	2.8864	0.5886	2.8864	27.3707

Table 15 Optimal controller parameters that minimize $I_{ITSE}^{T_\Sigma}$

$(\gamma^{T_\Sigma})^2$	β^*	k_C^*	T_i^*	$I_{ITSE}^{T_\Sigma}$
0	4.5607	0.4683	4.5607	1.8650
0.1	3.9478	0.5033	3.9478	2.2029
1	3.1709	0.5616	3.1709	4.6304
10	2.8995	0.5873	2.8995	27.2770

Table 16 Optimal controller parameters that minimize $I_{ITAE}^{T_\Sigma}$

$(\gamma^{T_\Sigma})^2$	β^*	k_C^*	T_i^*	$I_{ITAE}^{T_\Sigma}$
0	3.5531	0.5305	3.5531	13.0262
0.1	3.5440	0.5312	3.5440	13.3051
1	3.4389	0.5392	3.4389	15.7772
10	3.0619	0.5715	3.0619	39.0401

This case study shows the reduced effects of the initial conditions associated to the sensitivity models (29) and (30) of SA algorithm on the optimal values of the controller tuning parameters.

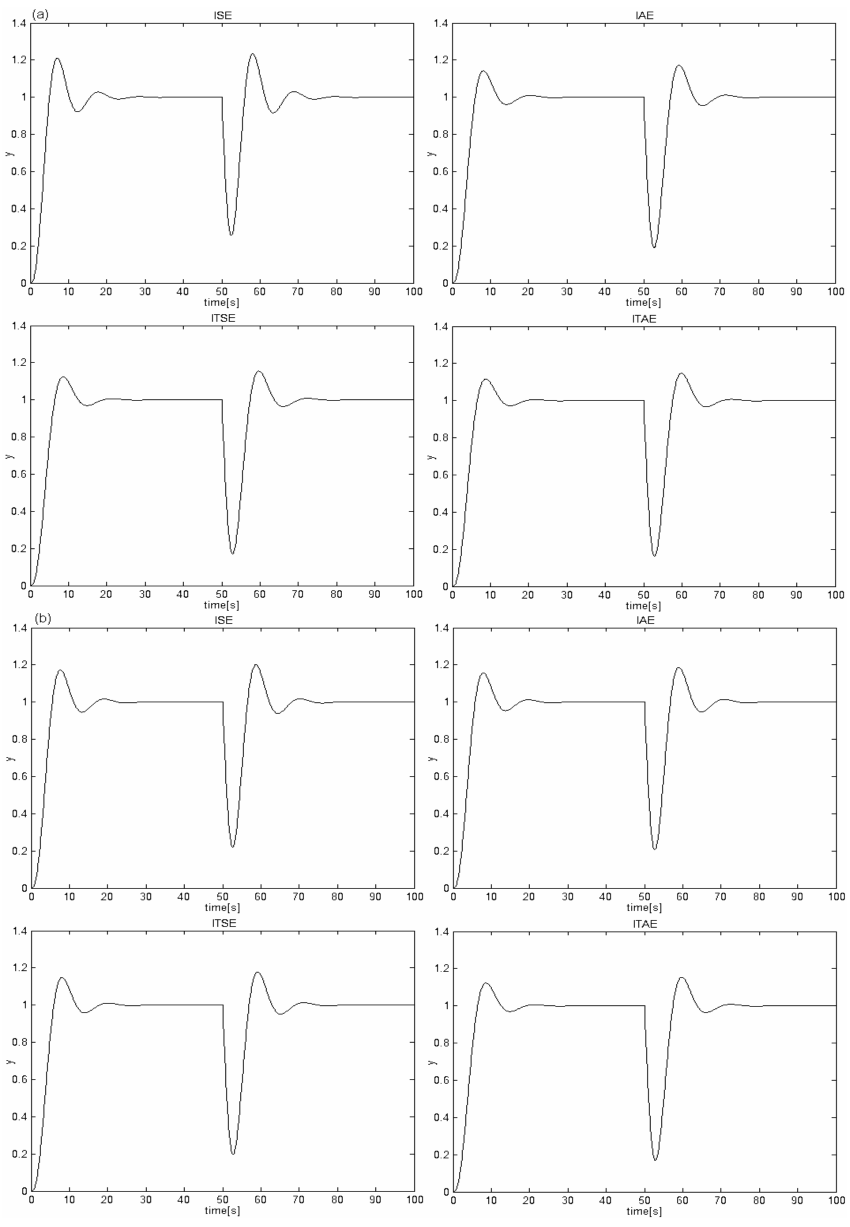


Fig. 8 Simulation results of optimal control systems obtained by PSO algorithm for the sensitivity model (29) (a) and sensitivity model (30) (b).

The controlled output (y) versus time diagrams given in Fig. 8 (a) and Fig. 8 (b) illustrate the optimization of the control system behavior by the minimization of the families of objective functions I^{k_P} and I^{T_Σ}, for $(\gamma^{k_P})^2 = 1$ and $(\gamma^{T_\Sigma})^2 = 1$, respectively, for the parameters of the optimal PI controller obtained by PSO algorithm with $j_{\max} = 100$. These simulations employed a unit step reference input, followed by a –0.5 step disturbance input applied at 50 s.

Considering the same simulation scenario, Fig. 9 (a) and Fig. 9 (b) illustrate the behaviors of the control systems optimized by the minimization of the objective functions I^{k_P} and I^{T_Σ}, for $(\gamma^{k_P})^2 = 1$ and $(\gamma^{T_\Sigma})^2 = 1$, respectively, and the parameters of the optimal PI controller obtained by SA algorithm.

The simulation results presented in Figs. 8 and 9 convincingly show that the controller exhibits good setting time and overshoot performance indices. The rapid decay of the system responses in Figs. 8 and 9 is explained by the application of the –0.5 step disturbance input (*d* in Fig. 1 and in Fig. 3) at 50 s.

Tables 1 to 16 show close values of the optimal parameters of the controllers for all the eight optimization problems solved. Furthermore, PSO and SA algorithms applied in solving the same optimization problems lead to very close solutions to those problems.

Figs. 10 and 11 illustrate the evolution of the design parameter β and the corresponding values of the objective function $I_{ISE}^{k_P}(\beta)$ versus the iterations of PSO and SA algorithm, respectively. The weighting parameter values used in these cases were $(\gamma^{k_P})^2 = 1$ and $(\gamma^{T_\Sigma})^2 = 1$ for both objective functions, and $j_{\max} = 100$ was considered in PSO algorithm.

Using the same weighting parameters and maximum number of iterations in PSO algorithm, Figs. 12 and 13 illustrate the evolution of the design parameter β and the corresponding values of another objective function, $I_{ISE}^{T_\Sigma}(\beta)$, versus the iterations of PSO and SA algorithm, respectively.

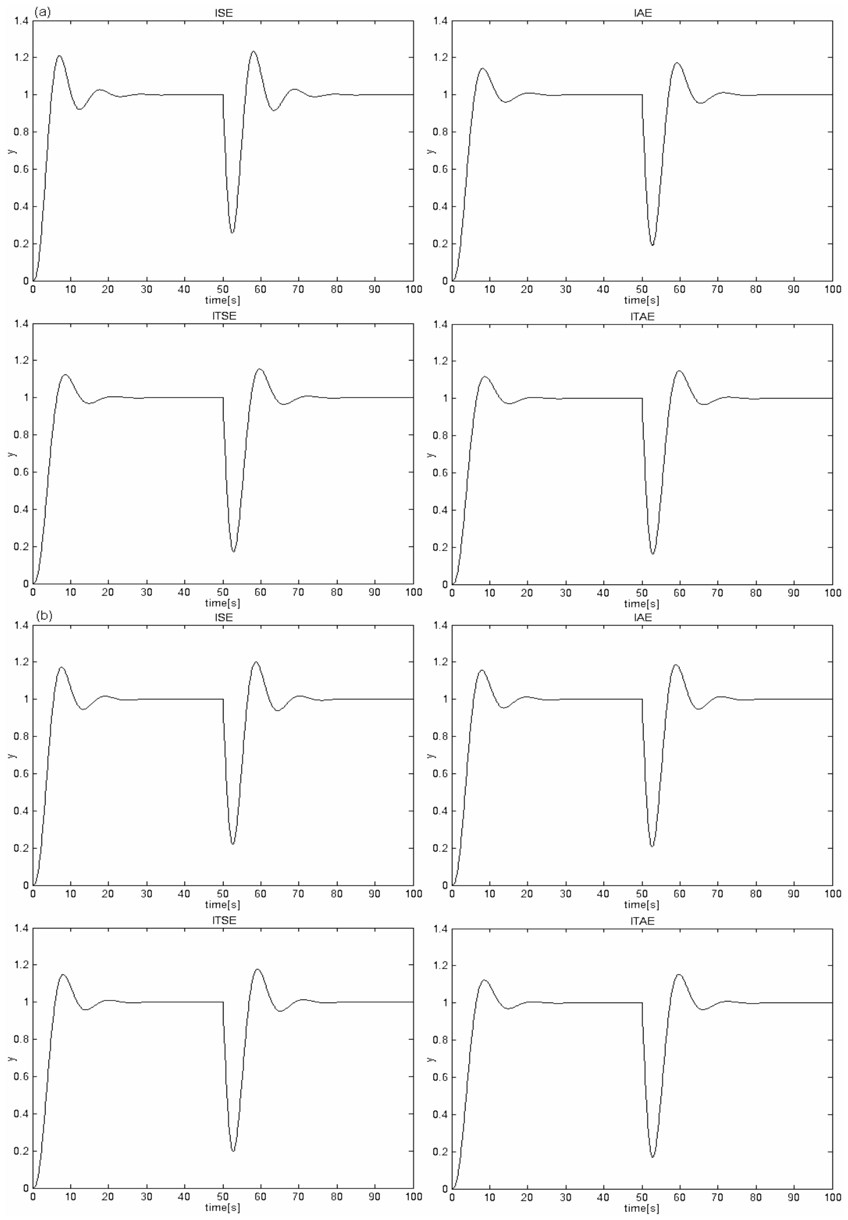
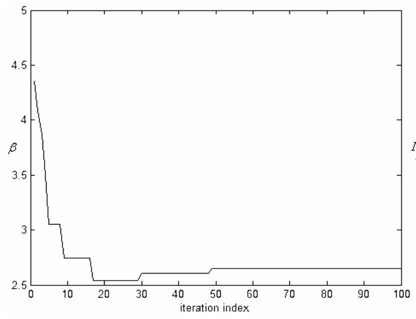


Fig. 9 Simulation results of optimal control systems obtained by SA algorithm for the sensitivity model (29) (a) and sensitivity model (30) (b).



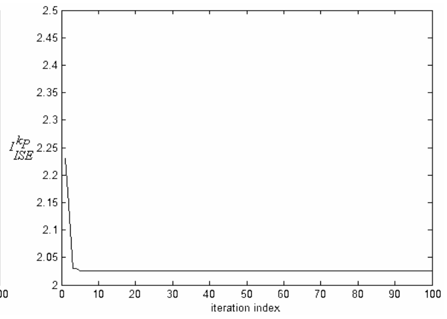
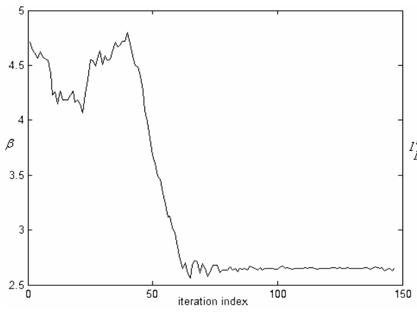


Fig. 10 Evolution of design parameter β and objective function $I_{ISE}^{k_P}$ versus the iteration index in PSO algorithm.



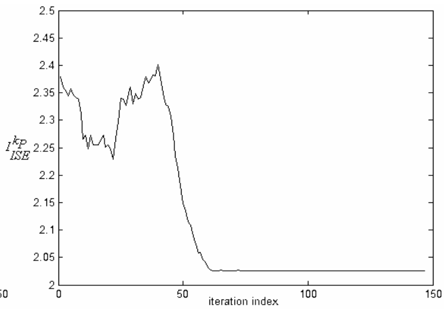
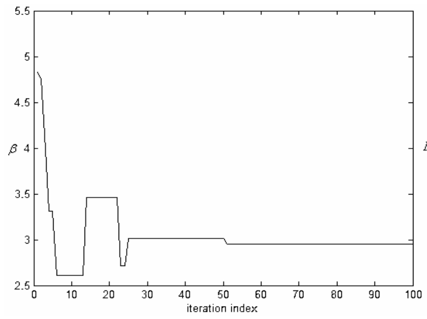


Fig. 11 Evolution of design parameter β and objective function $I_{ISE}^{k_P}$ versus the iteration index in SA algorithm.



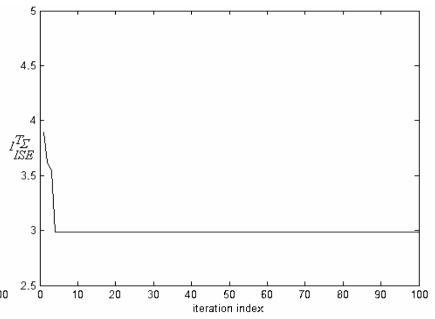


Fig. 12 Evolution of design parameter β and objective function $I_{ISE}^{T_\Sigma}$ versus the iteration index in PSO algorithm.

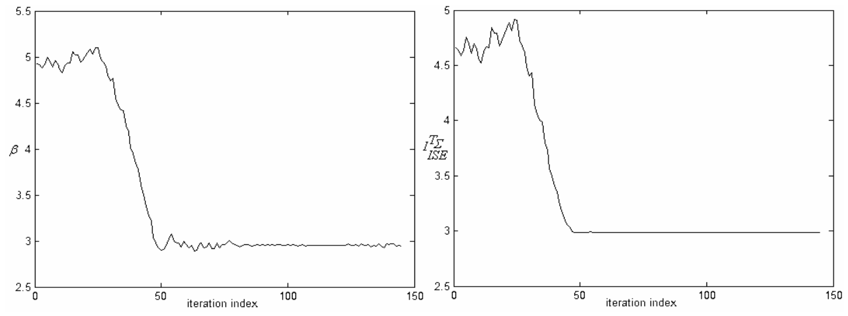


Fig. 13 Evolution of design parameter β and objective function $I_{ISE}^{T_\Sigma}$ versus the iteration index in SA algorithm.

The close values produced by the two PSO and SA optimization algorithms in similar situations demonstrate that the global minimum has been reached and thus the correct solutions to the optimization problems defined in equations (12) were obtained. Both algorithms have also avoided being trapped in local minima.

5 Conclusions

In this chapter, an application of PSO and SA optimization algorithms to the optimal design of PI controllers with a reduced sensitivity with respect to the parametric variations of the controlled process was presented. The proposed optimization problems ensure the weighted minimization of several integrals of the control error eventually weighted by time, and output sensitivity functions.

The advantages of using these algorithms include an expedient implementation, good computational efficiency and convergence.

Simulations were conducted to illustrate the good behavior of the optimal control systems in the dynamic regimes characterized by the step modifications of reference and disturbance inputs. Both optimization algorithms have a reduced sensitivity to the initial conditions of the sensitivity models.

The main limitation for these algorithms regards the degrees of freedom represented by some of their parameters. SA algorithm also presents a more computational-intensive disadvantage compared to PSO algorithm.

Future research will study the applicability of these optimization algorithms to other sensitivity-based optimization problems in the frequency domain, the extension of their area of application to other processes and controller structures [58–63]. All applications should be accompanied by the systematic analysis and guarantee of the convergence of the algorithms such that to enable the implementation of low cost and hybrid algorithms.

Acknowledgments. This work was supported by the CNCSIS and CNMP of Romania. The support from the cooperation between the "Politehnica" University of Timisoara, Romania, the Óbuda University, Budapest, Hungary, and the University of Ljubljana, Slovenia, in the framework of the Hungarian-Romanian and Slovenian-Romanian Intergovernmental Science & Technology Cooperation Programs is dully acknowledged.

References

[1] Rosenwasser, E., Yusupov, R.: Sensitivity of automatic control systems. CRC Press, Boca Raton (2000)

[2] Yaniv, O.: Design of low-order controllers satisfying sensitivity constraints for unstructured uncertain plants. Int. J. Robust Nonlinear Control 14, 1359–1370 (2004)

[3] Oded, T., Arkady, L.: Design of low order controllers satisfying sensitivity constraints for unstructured uncertain plants. In: Proceedings of 23rd IEEE Convention of Electrical and Electronics Engineers in Israel, Herzlia, Israel, pp. 33–36 (2004)

[4] Shirao, J., Imai, J., Konishi, M.: Structure design with sensitivity control performance limitation for electromechanical systems. In: Proceedings of SICE-ICASE International Joint Conference SICE-ICCAS 2006, Busan, Korea, pp. 2362–2367 (2006)

[5] Precup, R.E., Preitl, S., Korondi, P.: Fuzzy controllers with maximum sensitivity for servosystems. IEEE Trans. Ind. Electron 54, 1298–1310 (2007)

[6] Marchetti, G., Barolo, M., Jovanovic, L., Zisser, H., Seborg, D.E.: An improved PID switching control strategy for type 1 diabetes. IEEE Trans. Biomed. Eng. 55, 857–865 (2008)

[7] Wang, Y.G., Xu, X.M.: PID tuning for unstable processes with sensitivity specification. In: Proceedings of Chinese Control and Decision Conference CCDC 2009, Guilin, China, pp. 3460–3464 (2009)

[8] Precup, R.E., Preitl, S.: Optimisation criteria in development of fuzzy controllers with dynamics. Eng. Appl. Artif. Intell. 17, 661–674 (2004)

[9] Köppen, M.: Light-weight evolutionary computation for complex image-processing applications. In: Proceedings of 6th International Conference on Hybrid Intelligent Systems HIS 2006, Auckland, New Zealand, pp. 3–3 (2006)

[10] Zhou, H., Schaefer, G., Shi, C.: A mean shift based fuzzy c-means algorithm for image segmentation. In: Proceedings of 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society EMBC 2008, Vancouver, BC, Canada, pp. 3091–3094 (2008)

[11] Schaefer, G., Nakashima, T., Zavisek, M.: Analysis of breast thermograms based on statistical image features and hybrid fuzzy classification. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., Rhyne, T.-M., Monroe, L. (eds.) ISVC 2008, Part I. LNCS, vol. 5358, pp. 753–762. Springer, Heidelberg (2008)

[12] Abraham, A., Corchado, E., Corchado, J.M.: Hybrid learning machines. Neurocomputing 72, 2729–2730 (2009)

[13] Precup, R.E., Preitl, S.: On the stability and sensitivity analysis of fuzzy control systems for servo-systems. In: Nedjah, N., de Macedo Mourelle, L. (eds.) Fuzzy Systems Engineering, Theory and Practice, pp. 131–161. Springer, Heidelberg (2005)

[14] Ekel, P.Y., Menezes, M., Schuffner Neto, F.H.: Decision making in a fuzzy environment and its application to multicriteria power engineering problems. Nonlinear Analysis: Hybrid Syst. 1, 527–536 (2007)

[15] Xu, J., Wu, H., Wang, Y.: Unpower aerocraft augmented state feedback tracking guaranteed cost control. J. Syst. Eng. Electron 19, 125–130 (2008)

[16] Fmmo, C., Calado, JMF.: Approaches to human arm movement control - A review. Annu Rev Control 33, 69–77 (2009)

[17] Chen, J., Kong, C.K.: Performance assessment for iterative learning control of batch units. J. Process Control 19, 1043–1053 (2009)

[18] Kim, D.H., Abraham, A., Cho, J.H.: A hybrid genetic algorithm and bacterial foraging approach for global optimization. Inf. Sci. 177, 3918–3937 (2007)

[19] Nolle, L.: On a novel ACO-estimator and its application to the target motion analysis problem. Knowl-Based Syst. 21, 225–231 (2008)

[20] Köppen, M., Kinoshita, Y., Yoshida, K.: Auxiliary objectives for the evolutionary multiobjective principal color extraction from logo images. In: Proceedings of IEEE Congress on Evolutionary Computation CEC 2008, Hong Kong, China, pp. 3537–3544 (2008)

[21] Plant, W.R., Schaefer, G., Nakashima, T.: An overview of genetic algorithms in simulation soccer. In: Proceedings of IEEE Congress on Evolutionary Computation CEC 2008, Hong Kong, China, pp. 3897–3904 (2008)

[22] Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks ICNN 1995, Perth, Australia, pp. 1942–1948 (1995)

[23] Kennedy, J., Eberhart, R.C.: A new optimizer using particle swarm theory. In: Proceedings of 6th International Symposium on Micro Machine and Human Science MHS 1995, Nagoya, Japan, pp. 39–43 (1995)

[24] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 20, 671–680 (1983)

[25] Geman, S., Geman, D.: Stochastic relaxation, Gibbs distribution and the Bayesian restoration in images. IEEE Trans. Pattern Anal. Mach. Intell. 6, 721–741 (1984)

[26] Kalai, A.T., Vempala, S.: Simulated annealing for convex optimization. Math. Oper. Res. 31, 253–266 (2006)

[27] Ledesma, S., Torres, M., Hernández, D., Aviña, G., García, G.: Temperature cycling on simulated annealing for neural network learning. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 161–171. Springer, Heidelberg (2007)

[28] David, R.C., Rădac, M.B., Preitl, S., Tar, J.K.: Particle swarm optimization-based design of control systems with reduced sensitivity. In: Proceedings of 5th International Symposium on Applied Computational Intelligence and Informatics SACI 2009, Timisoara, Romania, pp. 491–496 (2009)

[29] Kim, T.H., Maruta, I., Sugie, T.: Robust PID controller tuning based on the constrained particle swarm optimization. Automatica 44, 1104–1110 (2008)

[30] Lin, C.M., Lin, M.H., Chen, C.H., Yeung, D.S.: Robust PID control system design for chaotic systems using particle swarm optimization algorithm. In: Proceedings of 2009 International Conference on Machine Learning and Cybernetics ICMLC, Baoding, China, vol. 6, pp. 3294–3299 (2009)

[31] Fang, H., Chen, L.: Application of an enhanced PSO algorithm to optimal tuning of PID gains. In: Proceedings of Chinese Control and Decision Conference CCDC 2009, Guilin, China, pp. 35–39 (2009)

[32] Su, C., Wu, Y.: Adaptive neural network predictive control based on PSO algorithm. In: Proceedings of Chinese Control and Decision Conference CCDC 2009, Guilin, China, pp. 5829–5833 (2009)

[33] Mukherjee, V., Ghoshal, S.P.: Intelligent particle swarm optimized fuzzy PID controller for AVR system. Electr. Power Syst. Res. 77, 1689–1698 (2007)

[34] Mohan Rao, A.R., Sivasubramanian, K.: Multi-objective optimal design of fuzzy logic controller using a self configurable swarm intelligence algorithm. Comput. Struct. 86, 2141–2154 (2008)

[35] Tang, H.Y., Ding, B., Qi, W.G.: Research on traffic mode of elevator applied fuzzy C-mean clustering algorithm based on PSO. In: Proceedings of International Conference on Measuring Technology and Mechatronics Automation ICMTMA 2009, Zhangjiajie, China, vol. 2, pp. 582–585 (2009)

[36] Veenhuis, C., Köppen, M., Vicente-Garcia, R.: Evolutionary multi-objective optimization of particle swarm optimizers. In: Proceedings of IEEE Congress on Evolutionary Computation CEC 2007, Singapore, pp. 2273–2280 (2007)

[37] Das, S., Abraham, A., Konar, A.: Automatic kernel clustering with a multi-elitist particle swarm optimization algorithm. Pattern Recognit. Lett. 29, 688–699 (2008)

[38] Shayeghi, H., Shayanfar, H.A., Jalili, A.: Load frequency control strategies: a state-of-the-art survey for the researcher. Energy Convers Manag. 50, 344–353 (2009)

[39] Sabat, S.L., dos Santos Coelho, L., Abraham, A.: MESFET DC model parameter extraction using quantum particle swarm optimization. Microelectron Reliab. 49, 660–666 (2009)

[40] Liang, X.R., Fan, Y.K., Jiang, T.: Application of PSO algorithm to coordinated ramp control. In: Proceedings of 2009 International Conference on Machine Learning and Cybernetics ICMLC, Baoding, China, vol. 3, pp. 1712–1716 (2009)

[41] Zhang, Y., Hu, Y.: On PID controllers based on simulated annealing algorithm. In: Proceedings of 27th Chinese Control Conference CCC 2008, Kunming, China, pp. 225–228 (2008)

[42] Cao, X.R.: Stochastic learning and optimization - A sensitivity-based approach. Annu. Rev. Control 33, 11–24 (2009)

[43] Qiu, X.Z., Xu, Z.G., Zhang, L.M., Zhou, J.X., Si, F.Q.: Nonlinear predictive control on the load system of a thermal power unit based on AOSVR and SAPSO. In: Proceedings of Asia-Pacific Power and Energy Engineering Conference APPEEC 2009, Wuhan, China, p. 4 (2009)

[44] Wu, M., Xu, C.H., She, J.H., Yokoyama, R.: Intelligent integrated optimization and control system for lead-zinc sintering process. Control Eng. Pract. 17, 280–290 (2009)

[45] Haber, R.E., Haber-Haber, R., Jiménez, A., Galán, R.: An optimal fuzzy control system in a network environment based on simulated annealing. An application to a drilling process, Appl. Soft Comput. 9, 889–895 (2009)

[46] Buyamin, S., Finch, J.W.: Comparative study on optimising the EKF for speed estimation of an induction motor using simulated annealing and genetic algorithm. In: Proceedings of IEEE International Electric Machines & Drives Conference IEMDC 2007, Antalya, Turkey, vol. 2, pp. 1689–1694 (2007)

[47] Sayol, J., Nolle, L., Schaefer, G., Nakashima, T.: Comparison of simulated annealing and SASS for parameter estimation of biochemical networks. In: Proceedings of IEEE Congress on Evolutionary Computation CEC 2008, Hong Kong, China, pp. 3568–3571 (2008)

[48] Qin, X.S., Huang, G.H., He, L.: Simulation and optimization technologies for petroleum waste management and remediation process control. J. Environ. Manag. 90, 54–76 (2009)

[49] Åström, K.J., Hägglund, T.: PID controllers theory: design and tuning. Instrument Society of America, Research Triangle Park, NC (1995)

[50] Preitl, S., Precup, R.E.: An extension of tuning relations after symmetrical optimum method for PI and PID controllers. Automatica 35, 1731–1736 (1999)

[51] Abonyi, J.: Fuzzy model identification for control. Birkhäuser, Boston (2003)

[52] Itagaki, N., Nishimura, H., Takagi, K.: Two-degree-of-freedom control system design in consideration of actuator saturation. IEEE/ASME Trans. Mechatronics 13, 470–475 (2008)

[53] Precup, R.E., Preitl, S.: PI and PID controllers tuning for integral-type servo systems to ensure robust stability and controller robustness. Electrical Eng (Archiv für Elektrotechnik) 88, 149–156 (2006)

[54] del Valle, Y., Venayagamoorthy, G.K., Mohagheghi, S., Hernandez, J.C., Harley, R.G.: Particle swarm optimization: Basic concepts, variants and applications in power systems. IEEE Trans. Evol. Comput. 12, 171–195 (2008)

[55] Khanesar, M.A., Tavakoli, H., Teshnehlab, M., Shoorehdeli, M.A.: A novel binary particle swarm optimization. In: Proceedings of Mediterranean Conference on Control & Automation MED 2007, Athens, Greece, p. 6 (2007)

[56] Kessler, C.: Das symmetrische Optimum. Teil I. Regelungstechnik 6, 395–400 (1955)

[57] Kessler, C.: Das symmetrische Optimum. Teil II. Regelungstechnik 6, 432–436 (1955)

[58] Horváth, L., Rudas, I.J.: Modeling and problem solving methods for engineers. Academic Press, Elsevier, Burlington, MA (2004)

[59] Johanyák, Z.C., Kovács, S.: Sparse fuzzy system generation by rule base extension. In: Proceedings of 11th International Conference on Intelligent Engineering Systems INES 2007, Budapest, Hungary, pp. 99–104 (2007)

[60] Vaščák, J.: Fuzzy cognitive maps in path planning. Acta Tech. Jaurinensis Ser. Intell. Comput. 1, 467–479 (2008)

[61] Klančar, G., Matko, D., Blažič, S.: Wheeled mobile robots control in a linear platoon. J. Intell. Robotic Syst. 54, 709–731 (2009)

[62] Chiou, J.S., Liu, M.T.: Numerical simulation for fuzzy-PID controllers and helping EP reproduction with PSO hybrid algorithm. Simul Modell Pract. Theory 17, 1555–1565 (2009)

[63] Liu, K., Tan, Y., He, X.: An adaptive staged PSO based on particles' search capabilities. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010. LNCS, vol. 6145, pp. 52–59. Springer, Heidelberg (2010)

Comparison of Evolutionary and Swarm Intelligence Methods for History Matching and Uncertainty Quantification in Petroleum Reservoir Models

Yasin Hajizadeh, Vasily Demyanov, Linah Mohamed, and Mike Christie

Abstract. Petroleum reservoir models are vital tools to help engineers in making field development decisions. Uncertainty of reservoir models in predicting future performance of a field needs to be quantified for risk management practices. Rigorous optimisation and uncertainty quantification of the reservoir simulation models are the two important steps in any reservoir engineering study. These steps facilitate decision making and have a direct impact on technical and financial performance of oil and gas companies.

Optimisation of reservoir models to match past petroleum production data – history matching – entails tuning the simulation model parameters to reproduce dynamic data profiles observed at the production wells. History matching is an inverse problem with non-unique solution; thus, different combinations of reservoir model parameters can provide a good match to the data. Multiple history matched reservoir models are used to quantify uncertainty of future hydrocarbon production from a field.

Recently application of evolutionary and swarm intelligence algorithms to history matching problems has become very popular. Stochastic sampling algorithms are used to explore the model parameter space and to find good fitting models. Exploration/exploitation of the search space is essential to obtain a diverse set of history matched reservoir models. Diverse solutions from different regions of the search space represent different possible realizations of the reservoir model and are essential for realistic uncertainty quantification of reservoir performance in the future.

This chapter compares the application of four recent stochastic optimisation methods: Ant Colony Optimisation, Differential Evolution, Particle Swarm Optimisation and the Neighbourhood Algorithm for the problem of history matching. The algorithms are integrated within a Bayesian framework to quantify uncertainty of the predictions.

Yasin Hajizadeh · Vasily Demyanov · Linah Mohamed · Mike Christie
Institute of Petroleum Engineering, Heriot Watt University,
EH14 4AS, Edinburgh, United Kingdom
e-mail: Yasin.Hajizadeh@pet.hw.ac.uk

Two petroleum reservoir examples illustrate different aspects of the comparative study. The Teal South case study is a real reservoir with a simple structure and a single producing well. History matching of this model is a low dimensional problem with eight parameters. The second case study – PUNQ-S3 reservoir – is a synthetic benchmark problem in petroleum industry. The PUNQ-S3 model has a more complex geological structure than Teal South model, which entails solving a high dimensional optimisation problem. This model is fitted to multivariate production data coming from multiple wells.

Keywords: Inverse problem, evolutionary algorithms, Ant Colony Optimisation, Differential Evolution, Particle Swarm Optimisation, Neighbourhood Algorithm, history matching, Bayesian uncertainty quantification, petroleum, reservoir modelling.

1 Introduction

In the multibillion dollar oil and gas industry, making correct investment decisions depend on the ability to predict accurately the future performance of petroleum reservoirs. Reservoir simulation models are routinely used in petroleum industry to support field development decision making. Reservoir models include different information about geological, petrophysical and fluid properties of the reservoir. These models are used to estimate future hydrocarbon production from the field or to plan additional recovery operations in reservoir engineering studies.

There are two important steps in any petroleum reservoir forecasting study. First, computer simulation models need to be calibrated based on the past production observations of the reservoir. This step, called history matching (HM), usually takes a lot of manpower and computer resources. History matching entails tuning the properties of the reservoir model in such a way that computer simulations reproduce the observed production rate or pressure measurements available from the wells. History matching being an inverse problem has no unique solution. Thus, several combinations of the reservoir model parameter values, e.g. rock porosity and permeability, may provide a good match to the data. Non-uniqueness of history matching solutions describes uncertainty of reservoir production modelling. The second step in reservoir studies is uncertainty quantification of the predictions made by the computer models. Multiple history matched models of the reservoir are used to quantify uncertainty in predicting the future behaviour of the reservoir. Each history matched model, which reflects well the available production observations, may feature different geological and petrophysical properties. Due to a wide range of uncertainty about the sub-surface system, there would exist models with different properties that may provide equally good history match. Such diverse models are likely to show different production behaviour in the future due to different reservoir parameters. Diverse history matched models are important for realistic assessment of prediction uncertainty.

Real reservoir engineering studies are restricted by the number of simulations that can be performed in a limited time. Running a single simulation model may take hours or even days in a real life problem. This limitation motivates application of effective adaptive optimisation algorithms in a search for possible combinations of reservoir model properties which provide a good history match. Therefore, powerful and robust optimisation methods are needed to navigate the model parameter space and identify good fitting solutions. Such automation of the history matching process is seen as a major improvement for subsurface engineering teams in history matching studies, which are often done manually in contemporary industry practice.

Use of gradient optimisation methods for history matching started in late 1960's [1,2]. Later, when stochastic methods entered reservoir engineering arena, many works showed that simple optimisation methods are not good tools for solving complex history matching problems [3]. It was in early 90's that the trend in history matching was geared towards generating multiple history matched models [4]. Fig. 1 shows an example of misfit function for a history matching problem. In this example we will need to identify not only the global minimum, but also multiple local minima in the search space. It has been shown that a single best history matched model is not necessary a good predictor for future performance of the reservoir [5]. On the other hand it is difficult and inefficient to obtain multiple history matched models using conventional Monte Carlo approaches because these methods are not intelligent enough to maximise the number of multiple good-fitting models in a limited number of simulations.

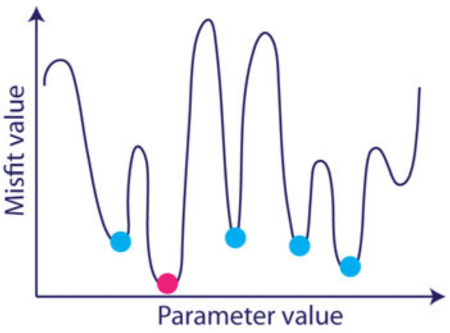


Fig. 1 Global minimum (pink) and multiple local minima (blue) in history matching problems.

Any optimisation method used in history matching must be fast in navigating high dimensional search spaces and efficient in finding multiple good models with a limited number of simulations. The importance of the optimisation algorithm choice and tuning becomes essential when we try to estimate tens or even hundreds of parameters in the presence of multiple local minima. Stochastic population-based algorithms are often seen as good candidates to solve history matching problems because they adaptively search for multiple good models and are less likely to get trapped in local minima. Examples of stochastic optimisation methods previously used for history matching include Genetic Algorithms (GA)

[6,7], Neighbourhood Algorithm (NA) [8], Particle Swarm Optimisation (PSO) [9,10,11], Evolutionary Strategies (ES) [12], Ant Colony Optimisation (ACO) [13] and Differential Evolution (DE) [14].

In this work we have compared the performance of four recent evolutionary and swarm intelligence methods to tackle history matching and uncertainty quantification problems. We compare their ability to find multiple history matched models which can realistically quantify uncertainty of the model forecast. The comparison is made using both synthetic and real reservoir studies, which are well known benchmarks in petroleum industry. The algorithms are integrated into a Bayesian framework for uncertainty quantification of reservoir predictions.

2 Uncertainty Quantification within Bayesian Framework

Uncertainty is an inherent feature of our understanding of reality. We can distinguish between uncertainty in mathematical models, which reflect our knowledge about nature; numerical errors, related to the accuracy of the computed solution due to discretization; and data uncertainty, which correspond to the quality and relevance of the available observations, etc.

Here we suggest a general framework for uncertainty quantification of natural systems predictions (see Fig. 2). Our understanding of the modelled natural system is based on data. By data we mean prior knowledge, which is used to build mathematical model relationships, and observations, which reflect the true but actually unknown behaviour of the system subject to measurements uncertainty.

Our prior beliefs set a range of model definitions described by parameters or scenarios. From these beliefs we parameterise the reservoir description and set prior probabilities for these parameters. Thus, a mathematical/statistical model of a petroleum reservoir describes the distribution of the porous medium properties, which can be defined by geological body types, spatial correlation range, etc. For example, in a fluvial reservoir laid down by an ancient river system we can distinguish sinuosity, width, depth, and other parameters of the meandering channels. Our prior knowledge about the parameters is based on the studies of equivalent outcrops and contemporary river systems. Setting our prior beliefs wide enough within physically realistic ranges provides a way of finding most likely and probable model solution. Evolutionary algorithms, in fact, act as the sampling method in our search space to find reservoir models that can reproduce the historical observations from the field. Multiple models obtained using these algorithms are sampled from our prior beliefs described by the probability distribution.

The model likelihood is evaluated by comparing the simulated model solution (reservoir simulation) with the available observations (i.e. history matching). By minimising the objective function (maximising the likelihood) through sampling a number of possible reservoir descriptions from the prior, we update our beliefs about a given set of models [15]. The likelihood $p(m|O)$ describes the probability of the model given the data, i.e. the measure of to what degree the observed and modelled data differ. Hence, it is directly related to the minimised objective function via the likelihood model. For instance, it is common to assume the log of

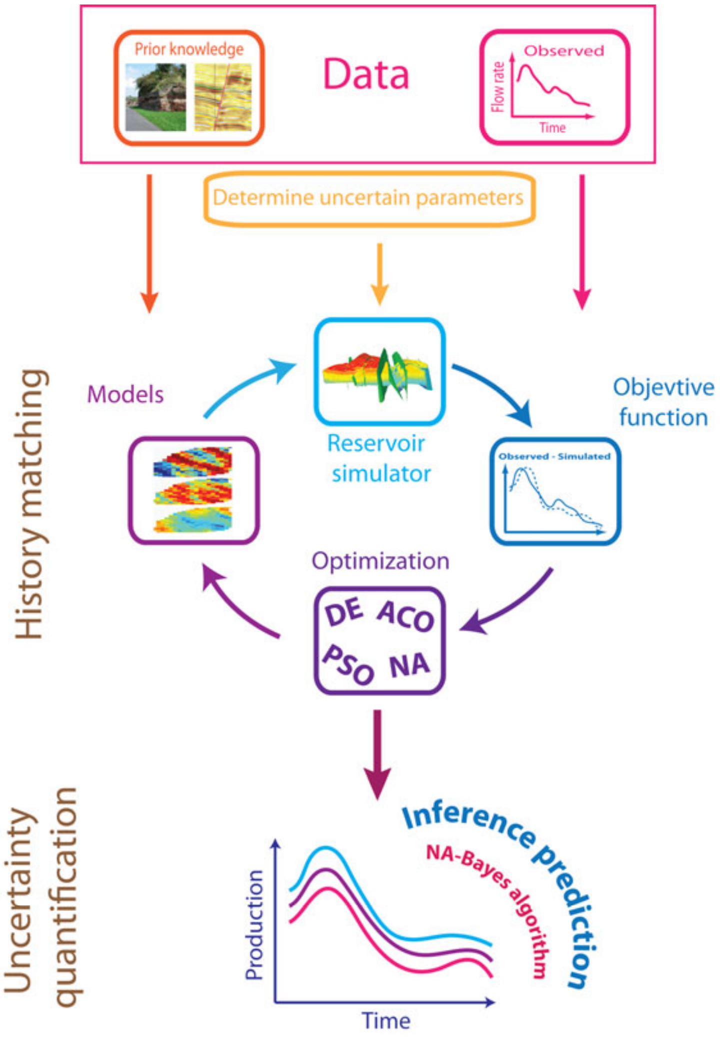


Fig. 2 History matching and uncertainty quantification framework.

the likelihood to equal the negative objective function. The later characterises how well the simulations fit the observed data. The goodness of fit is, often, evaluated by the square difference between observations and simulations normalised by double squared errors (inverse covariance matrix). This definition of the objective functions together with it's relation to the negative log of the likelihood assumes Gaussian statistics of errors. However, the choice of the objective function may vary depending on the optimisation task; some examples will be shown in the case studies section.

A Bayesian framework is a statistically consistent way relating our beliefs about the geological description to the available observations. Bayes theorem provides a formal way to update our beliefs about probabilities when we are provided with information:

$$p(m \mid O) = \frac{p(O \mid m)p(m)}{\int_M p(O \mid m)p(m)dm} \tag{1}$$

Bayesian theorem relates posterior probability *p(m|O)*) with prior probability, *p(m)* with the likelihood *p(O|m)*. Bayesian *inference* provides a way to evaluate the posterior probability *p(m|O)* of the multiple models generated using evolutionary optimisation. Multiple good fitting models generated by evolutionary algorithms models are highly likely (large *p(O|m)*) but are not equally probable. Their posterior probability *(p(m|O))* is computed numerically by Markov Chain Monte Carlo (MCMC) integration. An ensemble of models can be used to quantify the uncertainty of predictions (see Fig. 2).

3 Evolutionary Optimisation

Adaptive stochastic algorithms have been used for decades for solving optimisation and inverse problems in many engineering and scientific fields. Many of them, such as Genetic Algorithms (GA) and Simulated Annealing (SA) were inspired by the rules of nature. Evolutionary and swarm algorithms are more recent developments based on the idea of using a system of simple agents to describe complex phenomena with non-parametric dynamics. This is one of their key differences from the earlier algorithms, which were based on fairly complex components (chromosome in GA or a system state in SA). Recently proposed algorithms are more flexible and have more advanced adaptive capabilities provided by interaction of simple agents. The agents' dynamics is the workhorse of evolutionary and swarm intelligence algorithms. Similarities and differences between the algorithms can be observed by comparing the way how the agents are moving between the states of the system. It is essential to maintain a good balance between exploration and exploitation while searching for optimal solutions. Finding multiple local minima are essential for realistic quantification of prediction uncertainty, which is often not the case in many traditional optimisation problems. Therefore, this chapter aims to understand the search capabilities of Ant Colony Optimisation, Differential Evolution, Particle Swarm Optimisation and the Neighbourhood Algorithm using comparative study.

3.1 Ant Colony Optimisation

Ant Colony Optimisation (ACO) metaheuristic was proposed in 1992 by Dorigo [16]. It was inspired by studying the behaviour of real ants searching for their food. Real ants mark the path they follow with a chemical odorous substance called pheromone. The amount of pheromone laid by ants is a function of path length and food quality. Pheromone serves as a guide for other ants to find their way to good quality foods.

ACO was first designed for discrete problems and later was adapted to handle continuous optimisation problems. One of the most successful attempts to extend original ACO concept to handle continuous variables was by Socha and Dorigo [17]. The heart of ACO_R is a solution archive with *k* models which keeps the track of solutions (Fig. 3). The solutions in this archive are ranked based on their quality, which means models with lower misfit values get a higher rank and will appear at the top of the list. There are *k* rows and *n* columns in the archive, where *k* is the number of models that are kept in the archive and *n* is the number of dimensions of the problem or parameters in history matching. Each single row in this archive consists of a vector of reservoir model parameters (***s***) and corresponding objective function *f*(*s*) value that we obtain after running the flow simulation. The i^{th} unknown parameter value of the l^{th} model is denoted by s^i_l. The misfit values show how well the proposed model can fit historical data. The last column in this archive includes the weights (ω_l) of the solutions. These weights are a function of solution quality (Eq.2) and will be used to probabilistically build new solutions.

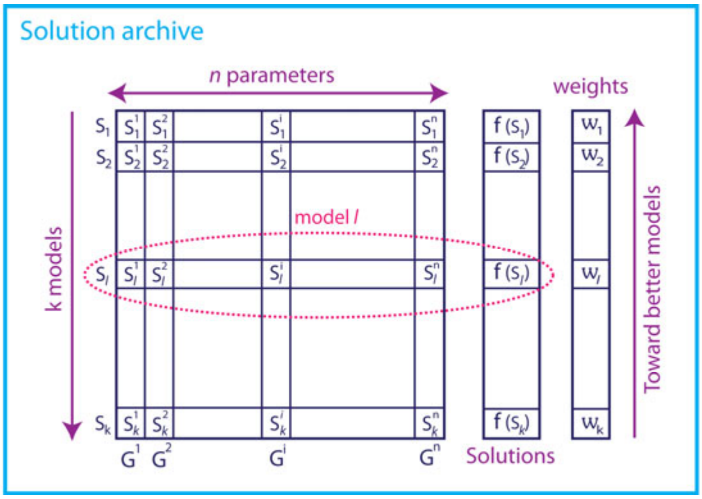


Fig. 3 Solution archive in ACO_R and its components (adopted from [17]).

The archive is initially filled with random solutions and the misfits (*f*(*s*)) of models are evaluated. If the number of ants evaluated at each iteration of the ACO_R algorithm is *m*, then at each iteration, *m* new solutions are added to the population and from the archive which now contains *k*+*m* models, the *m* worst solutions are removed to keep the archive size fixed. This action simulates the pheromone update part in discrete ACO. The remaining models in the archive are sorted according to their misfit score. The aim is to bias the search process towards the good regions of the search space with low misfit models by probabilistically constructing new solutions. Next, we compute the weights for each model in the archive. The weight of each member in the archive is computed based on the following equation and will be used to probabilistically select the members of archive:

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \tag{2}$$

where k is the size of the archive and q is a parameter of the algorithm that controls the balance between exploration/exploitation in ACO$_R$. Small values of q give preference to the best-ranked solutions, while for larger q values the probability of selecting models in the archive becomes more uniform.

In ACO$_R$ the selection probability is not a direct function of the fitness, but is computed indirectly from the fitness considering the effect of parameter q. In order to generate new solutions, one of the solutions in the archive is chosen according to the following probability:

$$p_l = \frac{\omega_l}{\sum_{r=1}^{k} \omega_r} \tag{3}$$

To construct a single model, at each step *(i=1, 2 ... n)* we sample a value for each unknown parameter in the problem. The mixture of Gaussian kernels form the probability density function (PDF) which will be sampled to obtain new members of the archive. For example in Fig. 4, we have 4 models in the archive to compute this mixture. The components of the solutions are used to dynamically generate probability density functions and modify their shape.

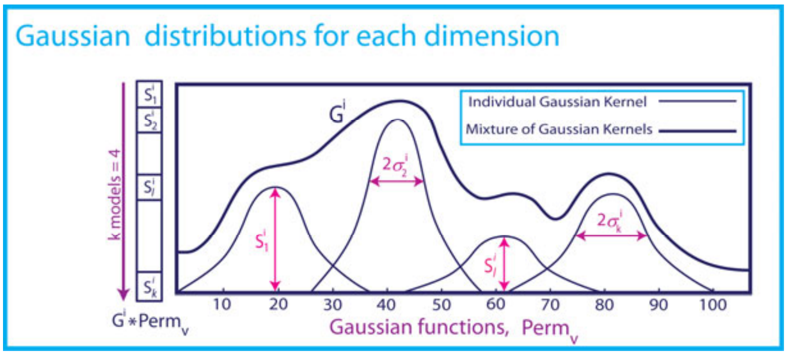

Gaussian distributions for each dimension
Individual Gaussian Kernel
Mixture of Gaussian Kernels
G^i
$2\sigma_2^i$
$2\sigma_k^i$
S_1^i
S_l^i
S_2^i
S_k^i
k models = 4
10 20 30 40 50 60 70 80 90 100
$G^i * Perm_v$
Gaussian functions, $Perm_v$

Fig. 4 Gaussian distributions in ACO$_R$ which are sampled to obtain new models.

For each dimension of the problem, there are k individual Gaussian functions. Each Gaussian distribution is characterised by its mean and standard deviation values. Mean value of each distribution is equal to the corresponding value in the solution archive. Considering a single column in the archive, the first number in this column will be considered as the mean value for the first individual Gaussian function and so on. For the standard deviations, we compute the average distance between the selected member (s_l) to the other members of that column in the solution archive. This computed value is multiplied by ξ which is called pheromone evaporation rate. The higher ξ value, worst solutions are forgotten faster. With lower ξ values, the search is less concentrated on the previously visited areas of the search space; hence the convergence speed of the algorithm will be higher. For obtaining new solutions, ants sample the mixture of Gaussian functions. The mixture of Gaussian kernels (Fig. 4, bold line) is the weighted sum of the individual Gaussian kernels and is computed with the following equation:

$$G^i(s) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(s-\mu_l^i)^2}{2\sigma_l^{i2}}} \tag{4}$$

where μ_l^i and σ_l^i are the mean and standard deviations of the Gaussian distributions. In the next step the Gaussian kernel is sampled to obtain a new model and this process is continued for all dimensions of the problem by each of the m ants, and at each iteration of the algorithm. The new members replace the models with least performance and this process continues until the search terminates with a stopping condition.

Socha and Dorigo showed that ACO$_R$ obtains best results among different implementations of Ant Colony Optimisation algorithm for handling continuous problems [17]. ACO$_R$ has also been used in other engineering fields and was proven to be a very efficient optimisation algorithm [18,19].

3.2 Differential Evolution

Differential Evolution (DE) is a powerful global optimisation algorithm and was introduced by Rainer Storn and Kenneth Price in 1995 [20]. DE grew out of Price's attempts to solve the Chebychev Polynomial fitting problem that has been posed to him by Storn. DE is a parallel population-based search algorithm which uses N_p D-dimensional parameter vectors as the population in each generation. Then it tries to evolve this population by simple arithmetic operations on these vectors to form new solutions to the problem. Fig. 5 illustrates the concept of building a difference vector in differential evolution. In this figure, DE is applied to find the minimum of a unimodal function.

In the first step the population is initialized with 6 individuals randomly scattered in the search space. Each individual member in the population is a vector of real numbers. All vectors are uniquely indexed from 1 to N_p for bookkeeping. In the second step two vectors are randomly selected among the current population and in the next step which is shows in part 3 of Fig. 5, the difference vector

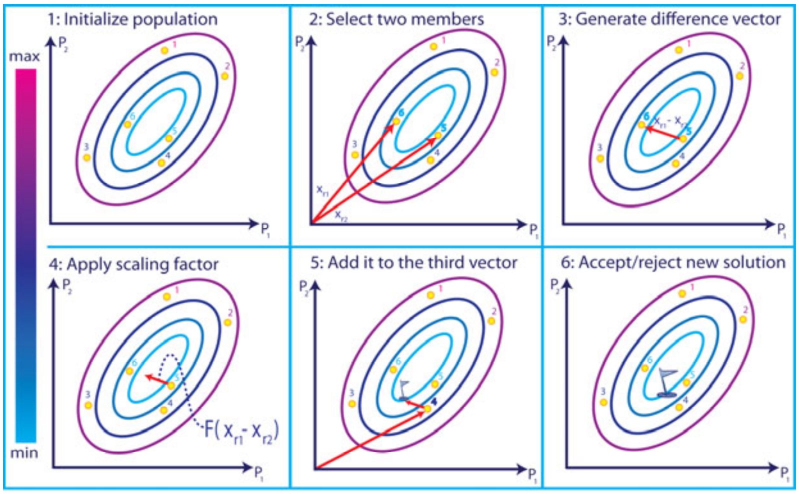


Fig. 5 Illustration of Differential Evolution algorithm.

between two selected members is computed. In step 4, this vector is multiplied by a number called scaling factor (*F*). Depending on the selection of the scaling factor value, the difference vector may become larger or smaller than its original size. In step 5, we select another member (individual 4) in the population and then add the scaled difference vector which is obtained in step 4, to this new selected member. This forms the trial vector and can be written as:

$$\upsilon_{G+1} = r_{1,G} + F(r_{2,G} - r_{3,G}) \tag{5}$$

where r_1 is the base vector and r_2 and r_3 are two other vectors chosen from the population. Scaling factor (*F*) is a real constant parameter. To increase the population diversity, crossover operation is performed after mutation step in differential evolution. In this step the parent (donor) vector is mixed with the mutated vector to produce the trial vector. There are two types of crossover schemes for differential evolution – binomial and exponential. In this work, we use exponential crossover scheme. Finally we come to the selection step which is indicated in part 6 of Fig. 5. In DE each trial vector competes against the population vector of same index. Because this new trial vector is the first new member generated, it is going to compete with the member indicated by index of 1. For the function shown in Fig. 5, new trial vector will be selected to proceed to the next generation because it has a lower objective function comparing to the individual number 1. The above procedure is repeated for each individual in N_p population to form the next generation of solutions. To summarize the concept of DE, we have the following pseudo code:

```
1. Initialize population with $N_p$ individuals.
2. Obtain objective function values for each member.
```

3. Select vectors.
4. Mutation (introduce perturbation).
5. Crossover (increase population diversity).
6. Selection (if new vector is better than parent).
7. Repeat steps 3-6 until stopping criteria is met (maximum number of iterations).

Various types of DE are different in the way they perform mutation and crossover steps. In the literature usually different variants of DE are presented in the form of DE/*x*/*y*/*z*, where *x* is the vector that will be mutated, *y* specifies number of difference vectors used and *z* is crossover scheme. In this work, we have used two strategies for building the trial vector (*random* and *best*). Recalling equation 5, in *DE/Rand* strategy, r_1, r_2 and r_3 vectors are all chosen randomly. For *DE/Best* scheme, the vector with lowest objective function value will be selected as the base vector (r_1). Since we use one difference vector and an exponential crossover scheme in our work, full name of these strategies can be written as *DE/Rand/1/exp* and *DE/Best/1/exp*.

3.3 Particle Swarm Optimisation

The Particle Swarm Optimisation (PSO), originally introduced by Kennedy and Eberhart in 1995 [21], has proven to be a powerful contender to other population-based evolutionary algorithms for global optimisation problems. It is a stochastic optimisation technique inspired by social behaviour of bird flocking or fish schooling. PSO has been successfully applied in a variety of fields including petroleum engineering (e.g., [9,10]).

The PSO algorithm starts with a random initialization of a swarm of particles in the search space. Each particle is considered as a candidate solution to a problem in *D*–dimensional space, with particle *i* representing x_i. Each particle maintains a memory of its previous best position, $pbest_i$, and a velocity along each dimension, represented as v_i. The *pbest* vector of the particle with the best fitness in the neighbourhood is designated as *gbest*. The importance of these two positions, *gbest* and the $pbest_i$, is weighted at each iteration by two factors c_1 and c_2 known as the cognitive and social scaling factor parameters [22]. These two elements are among the main governing parameters of swarm behaviour and algorithm efficiency, and have been the subject of many studies [23,24,25]. The update equation of the personal best position $pbest_i$ is presented in Eq. (6), assuming a minimisation problem where *f* denotes the objective function that is being minimised and *k is* the iteration (generation) number.

$$pbest_i^{k+1} = \begin{cases} pbest_i^k & \text{if } f(x_i^{k+1}) \geq f(pbest_i^k) \\ x_i^{k+1} & \text{if } f(x_i^{k+1}) < f(pbest_i^k) \end{cases} \tag{6}$$

In the PSO algorithm, at each iteration, particle *i*'s velocity (v_i) and position (x_i) are updated using Eqs. (7) and (8),

$$v_i^{k+1} = wv_i^k + c_1\, rand_1 \times (\, pbest_i^k - x_i^k\,) + c_2\, rand_2 \times (\, gbest^k - x_i^k\,) \quad (7)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (8)$$

where c_1 and c_2 are non-negative constant real parameters, $rand_1$, and $rand_2$ are two random vectors with each component corresponding to a uniform random number between 0 and 1. Inertial weight (w) influences the convergence of the algorithm. Usually, the inertial weight is chosen to decrease linearly from 0.8 to 0.4 facilitating exploration at early stages of the optimisation process while focusing on exploitation of promising regions at later stages. We can summarize the PSO workflow in the following steps:

1. Initialize the swarm by assigning a random position in the search space to each particle with sensible random velocity.
2. Evaluate the fitness function for each particle.
3. For each particle, update the position and value of *pbest* – the best solution the particle has seen. If current fitness value of one particle is better than its *pbest* value, then its *pbest* value and the corresponding position are replaced by the current fitness value and position, respectively.
4. Update the global best fitness value and the corresponding best position.
5. Update the velocities and positions of all the particles using Equations (7) and (8).
6. Repeat steps 2–5 until a stopping criterion is met.

There exist several variants of PSO for improving the speed of convergence of the algorithm. They vary in the neighbourhood topology, velocity update rules, handling particle behaviour at the boundary, inertial weight choices (w) and others [10]. In our application we used the absorbing boundary strategy for handling the particles outside allowable search space. The off-limit particle is relocated at the boundary of the search space in that dimension, and the velocity component in that dimension is zeroed. Other boundary strategies can be found in [10]. The maximum velocity allowed in one dimension is a fraction of the scaled parameter range of the corresponding dimension and is set to 0.4 in this work.

3.4 Neighbourhood Algorithm

The Neighbourhood Algorithm (NA) is a recent stochastic optimisation method proposed by Sambridge [26]. It was originally developed for solving inverse problems in seismology and soon was applied in other fields including petroleum engineering [8,27].

The Neighbourhood algorithm uses the Voronoi cells concept to find good regions of the search space. The optimisation process in neighbourhood algorithm for generating multiple good fitting models is working in the following way:

1. At beginning of the sampling procedure, like almost other stochastic methods, the algorithm is initialized by randomly generating n_{si} models in the search space and the objective function value is evaluated by running reservoir simulation models and comparing the simulated results with observed data.
2. In each iteration the algorithm generates n_s models and calculates the objective function value for each of these members.
3. This is followed by ranking all models according to their misfit value.
4. From this ranking, the best n_r models with lowest misfit scores are chosen and then new n_s models are generated by uniform random walk within Voronoi cells of these best n_r cells. It is important to note that misfit value is assumed to be constant within each Voronoi cell.
5. This procedure is repeated until a predetermined stopping criteria (maximum number of iterations) is met.

One of the main advantages of the NA is that only two parameters control the behaviour of the algorithm, n_s and n_r. Sambridge states that n_s/n_r ratio rather than the individual values of these two tuning parameters, control the amount of exploration and exploitation for NA [26]. For example n_s/n_r=1 results in maximum exploration of the search space and with n_s/n_r<1, the algorithm will be more exploitative.

4 Case Studies

We consider two reservoir models in this chapter. The Teal South case study is a real reservoir with a simple structure and one well. This model is used to understand the behaviour of the algorithms in a low dimensional history matching problem. The second case – PUNQ-S3 reservoir – is a synthetic problem (based on the data from a real reservoir) which is widely used to benchmark the optimisation algorithms for history matching. The PUNQ-S3 model has a more complex geological structure than Teal South and history matching is based on multivariate data and multiple wells.

4.1 Teal South Reservoir

The Teal South reservoir model is used to test different optimisation algorithms for a low dimensional history matching problem. This reservoir (Fig. 6) is located in Eugene Island in the Gulf of Mexico. Teal South has a single production well that penetrates the 4500 ft sand. The reservoir simulation model is set up on an 11×11×5 corner point grid. The production history of reservoir consisting production rates of oil, gas and water for 1247 days is shown in Fig. 7. There are five geological layers in the model with uniform properties. The unknown parameters in history matching are horizontal permeability multipliers for each of these five layers (P1-P5), a single value for vertical to horizontal permeability ratio (P6), rock compressibility (P7) and aquifer strength (P8). History matching is only done on field oil production rate, which is included in a univariate objective function (Eq. 9). Parameterisation for the Teal South model and their prior range are shown in Table 1.

Table 1 Parameterisation and prior ranges for Teal South model

Parameters	Units	Prior Range
k_h (for each layer)	mD	$10^{\{1, 3\}}$
k_v/k_h	-	$10^{\{-4, -1\}}$
Rock compressibility	psi^{-1}	5×10^{-6} - 1×10^{-4}
Aquifer strength	MMSTB	$10^{\{7, 9\}}$

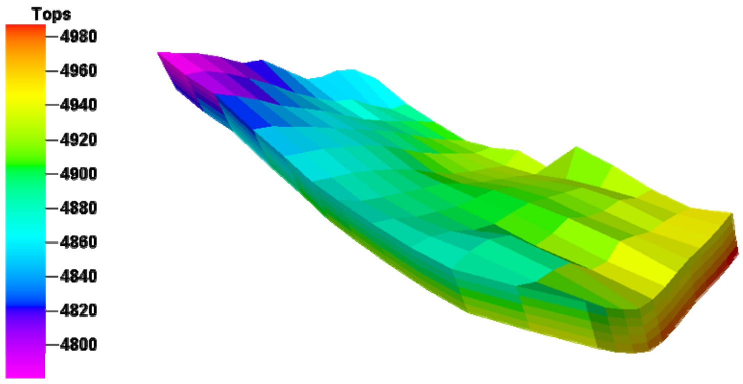


Fig. 6 Teal south reservoir.

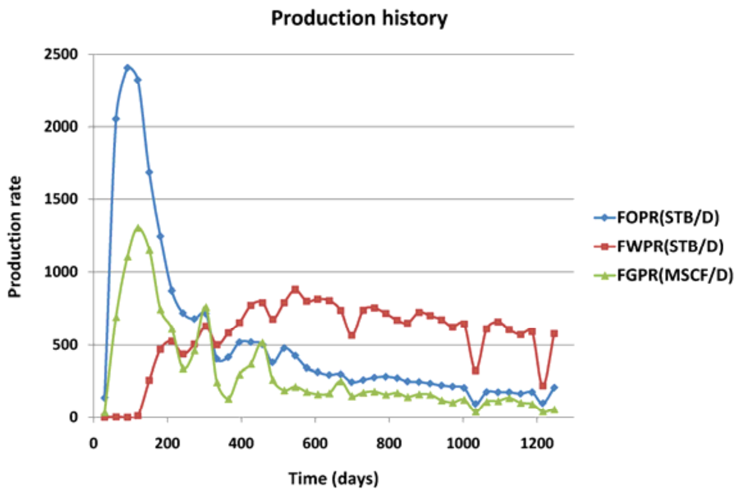


Fig. 7 Production history for Teal South.

We use different optimisation techniques to estimate unknown properties reported in Table 1. After estimating these properties, a flow simulation is performed for the resulted reservoir model using ECLIPSE simulator. The goal is to minimise the difference between output of simulation and the field observations using following least square norm objective function:

$$M = \sum_{n=1}^{N} \frac{(q_{obs}(t_i) - q_{sim}(t_i))^2}{2\sigma^2} \tag{9}$$

where N is the number of observations, q_{obs} is the observed flow rate in production well, q_{sim} is the flow rate obtained from reservoir simulator and σ^2 is the variance of the observed data.

4.2 PUNQ-S3 Model

The PUNQ-S3 reservoir is a synthetic model with 5 layers [28]. The top depth of PUNQ-S3 reservoir is 2430 *m*. It has a dip angle of about 1.5 degree and is bounded by a fault to the east and south and a relatively strong aquifer on the north and west provides a pressure support. There is also a small gas cap in the PUNQ-S3 reservoir model in layer 1. Six production wells are marked with black dots as can be seen in Fig. 8. The PUNQ-S3 model contains 19×28×5 grid blocks, with about two third of the grid blocks (1761) active. The grid blocks have equal 180 meter sides in x and y directions. The reservoir simulation case has been modelled with corner point geometry with a Carter-Tracy aquifer. The complete data set for this reservoir is available online [29].

The geological parameters are the horizontal (k_h) and vertical (k_v) permeabilities and the porosities (φ). We parameterise the PUNQ-S3 model with nine homogenous regions per layer. As mentioned earlier, PUNQ-S3 case has five layers. We estimate the porosities in each homogenous region and layer of the reservoir using the optimisation algorithms. Five layers times nine regions per layer makes 45 porosity values that will be estimated in the assisted history matching framework. Horizontal and vertical permeability values are then calculated from every 45 porosity values using relations 10 and 11. These correlations are obtained from least square fitting of the crossplots of porosity and permeability data in well locations [30]. Calculation of horizontal and vertical permeabilities completes the necessary parameters required to build the reservoir model and perform flow simulations.

$$\ln(k_h) = 0.77 + 9.03\phi \tag{10}$$

$$k_v = 3.124 + 0.306k_h \tag{11}$$

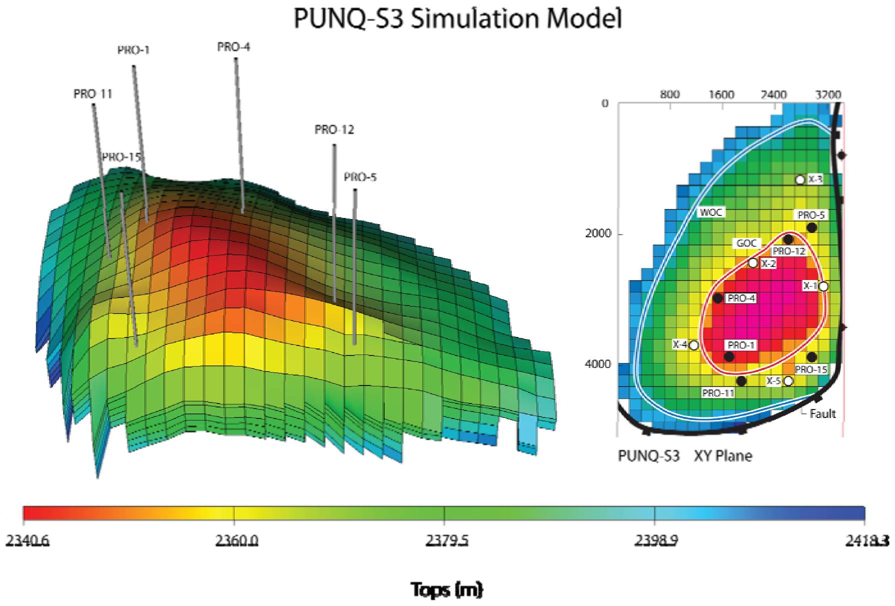


Fig. 8 PUNQ-S3 reservoir top surface map and well positions in this reservoir.

The prior ranges for unknown parameters in each layer of PUNQ-S3 reservoir are given in Table 2. These values are based on the geological description of the reservoir. Layers 1, 3 and 5 are high quality and layers 2 and 4 have lower porosity and permeability values.

Table 2 Initial ranges for parameters in PUNQ-S3 reservoir

Layer	Porosity (φ)	Horizontal Permeability (k_h) (mD)	Vertical Permeability (k_v) (mD)
1	0.15 - 0.3	133 - 3013	44 - 925
2	0.05 - 0.15	16 - 133	8 - 44
3	0.15 - 0.3	133 - 3013	44 - 925
4	0.1 - 0.2	47 - 376	17 - 118
5	0.15 - 0.3	133 - 3013	44 - 925

Simulated production history for the first 8 years from the six wells has been generated by the Netherlands Organization for Scientific Research (TNO). The production history includes pressure, water cuts and gas-oil ratios for each well. In order to reflect the real world measurement errors, Gaussian noise has been added to the well data. After 8 years, we continue the production for total period of 16.6 years from current wells. By estimating porosity and permeability values using different optimisation algorithms, input files of simulation are ready and the reservoir simulation can be performed. After running the simulation, we obtain different simulated field responses such as fluid production rates. By estimating proper porosity and permeability values, we try to minimise the following objective function in PUNQ-S3 case [31]:

$$SoS(o^{obs}, p) = \frac{1}{n_w}\sum_i \frac{1}{n_p}\sum_j \frac{1}{n_t}\sum_k \left(w_{ijk} \frac{o_{ij}^{obs}(t^k) - o_{ij}^{sim}(t^k; p)}{\sigma_{ijk}} \right)^2 \tag{12}$$

where n_w is number of wells with subscript i running over the wells, n_p is number of production data types with subscript j running over them. Subscript k runs over production data report times and n_t is the respective number of samples. Observed data (o^{obs}) (bottomhole pressures, gas oil ratio and watercuts) and simulated ones (o^{sim}) for each of the parameters (p) are being reported at time steps t^{k} with measurement error of σ. At each time step for the parameters, there are extra weighting factors denoted with w. These weights reflect the importance of some of data types at some specific time steps and are specifically indicated in the provided online dataset [29].

5 Results

The results of this work are presented in two sections. In section 5.1 we show the performance of algorithms for history matching of two reservoir models and in section 5.2, we study the uncertainty of predictions made by ensemble of models obtained with these algorithms.

5.1 History Matching

In this section we present the results of obtaining history matched models using evolutionary optimisation algorithms. We first start with Teal South reservoir, which is a simple model with 8 unknown parameters. Teal South is used as a proof-of-concept example. In the next section, we will discuss the results of history matching in PUNQ-S3 model, where we have 45 parameters to match.

5.1.1 Teal South Reservoir

For history matching of the Teal South reservoir, we have experimented with different combinations of the tuning parameters for optimisation algorithms in order to obtain the best results. A full study on effect of tuning parameters of algorithms is beyond the scope of this work; however, we have tried to set these parameters in such a way that a fair comparison of algorithms can be made. Tables 3-6 summarize the best results obtained for history matching and corresponding algorithm settings.

Table 3 Algorithm parameters and best misfit obtained for Differential Evolution (DE)

Algorithm	N_p	F	C_r	Generations	Best Misfit
DE-Rand	30	0.2	0.9	46	10.86
DE-Best	30	0.5	0.5	46	10.37

Table 4 Algorithm parameters and best misfit obtained for Particle Swarm Optimisation (PSO)

Swarm size	Inertial Weight	c_1	c_2	Generations	Best Misfit
30	0.8-0.4	2	2	46	8.42

Table 5 Algorithm parameters and best misfit obtained for Ant Colony Optimisation (ACO_R)

Number of ants	k	q	ξ	Generations	Best Misfit
30	30	0.3	0.3	46	11.27

Table 6 Algorithm parameters and best misfit obtained for Neighbourhood Algorithm (NA)

n_{si}	n_s	n_r	Generations	Best Misfit
30	30	15	46	8.42

A look at best misfits in the above tables reveals that all the methods can get to reasonably low objective function values, with PSO achieving slightly better final misfit. Fig. 9 shows the oil production rate as a result of history matching of Teal South reservoir model with different algorithms. As we can see from Fig. 9, in low dimensional history matching problem such as Teal South reservoir, performance of all algorithms is satisfactory. They all can provide an acceptable level of match and there is a marginal difference in terms of the final misfit value.

5.1.2 PUNQ-S3 Model

After testing a simple example to illustrate the concept of using evolutionary and swarm-based algorithms for history matching of petroleum reservoir models, let's move to a more complex problem with 45 unknown parameters. The goal of this study is to investigate the performance of these algorithms in a real life problem, where we may deal with tens of parameters to estimate. Tables 7-10 show the tuning parameters used in the different algorithms for history matching of PUNQ-S3 reservoir. These parameters are obtained after trying different settings and choosing the best results among these runs. The population size is same in all algorithms.

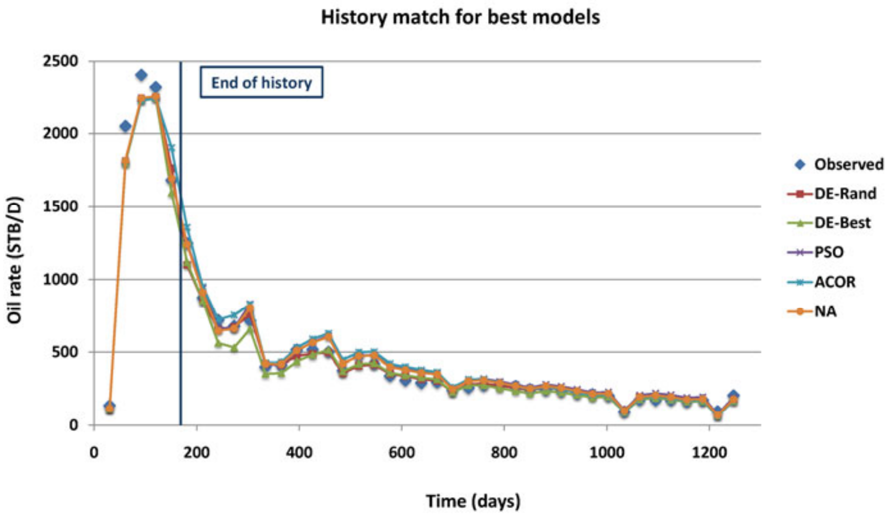


Fig. 9 Field oil production rate results obtained by each algorithm in history matching of Teal South reservoir.

Table 7 Algorithm parameters and best misfit obtained for Differential Evolution (DE)

Algorithm	N_p	F	C_r	Generations	Best Misfit
DE-Rand	50	0.5	0.7	60	1.95
DE-Best	50	0.5	0.5	60	1.45

Table 8 Algorithm parameters and best misfit obtained for Particle Swarm Optimisation (PSO)

Swarm size	Inertial Weight	c_1	c_2	Generations	Best Misfit
50	0.8-0.4	1	2	60	1.51

Table 9 Algorithm parameters and best misfit obtained for Ant Colony Optimisation (ACO_R)

Number of ants	k	q	ξ	Generations	Best Misfit
50	50	0.4	0.7	60	1.83

Table 10 Algorithm parameters and best misfit obtained for Neighbourhood Algorithm (NA)

n_{si}	n_s	n_r	Generations	Best Misfit
50	50	50	60	4.07

From the best misfits reported in the above tables, we see that DE, PSO and ACO$_R$ provide good final history matching results. The performance of the NA is not satisfactory in this high dimensional history matching problem. We used extreme exploration settings for NA (n_s/n_r=1) to perform widest search. Less explorative NA settings obtained larger misfit values, possibly due to over-refinement of local minima. DE-Best obtains a slightly lower misfit value than other algorithms for this case. Figs. 10-13 show the best history matching results for selected wells in PUNQ-S3 model. These figures show the quality of match for bottomhole pressure (BHP) in well 1, gas oil ratio (GOR) for wells 4 and 11 and water cut (WWC) in well 11. In these figures black filled circles show the observed data used to calculate misfit values during history matching. Solid black line shows the simulation performed using the truth case provided in the online dataset [29] for full production period (6025 days). Note that the observations may vary from the truth case due to the added noise.

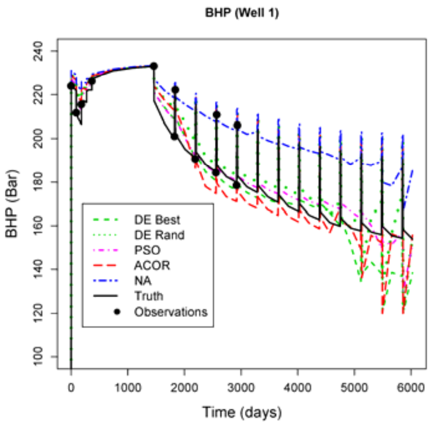


Fig. 10 BHP match result for well 1.

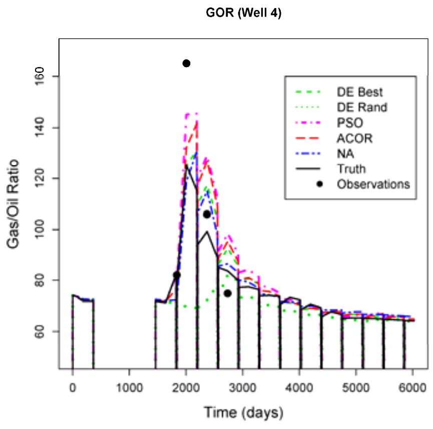


Fig. 11 GOR match result for well 4.

Figs 10-13 show different matching quality in wells. For example, both DE-Best and PSO algorithms have obtained very good misfit values (1.45 and 1.51). In Fig. 12 we see that PSO is the only algorithm that can capture the behaviour of truth model in prediction period (days 2936 to 6025). In Fig. 13 we see that PSO is not able to reproduce the water cut data and other algorithms are better in predicting water cut in well 11. We can also see the poor match quality of model obtained by NA. Different optimisation algorithms are obtaining history-matched models in different regions of the search space. These models represent diverse representations of the reservoir model, thus leading to different predictions.

Fig. 14 shows the convergence graph for the algorithms. It plots the best misfit obtained in each generation of the algorithms versus the generation number.

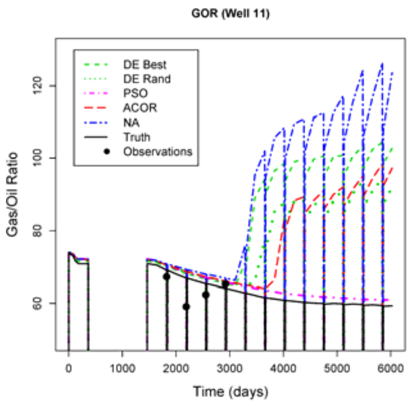


Fig. 12. GOR match result for well 11.

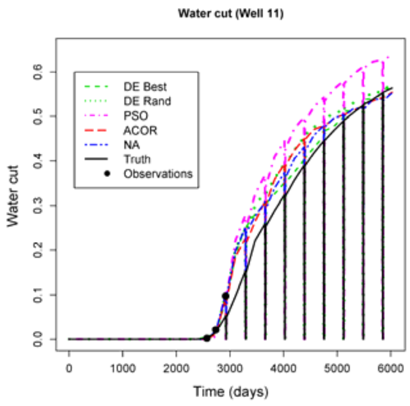


Fig. 13. WWC match result for well 11.

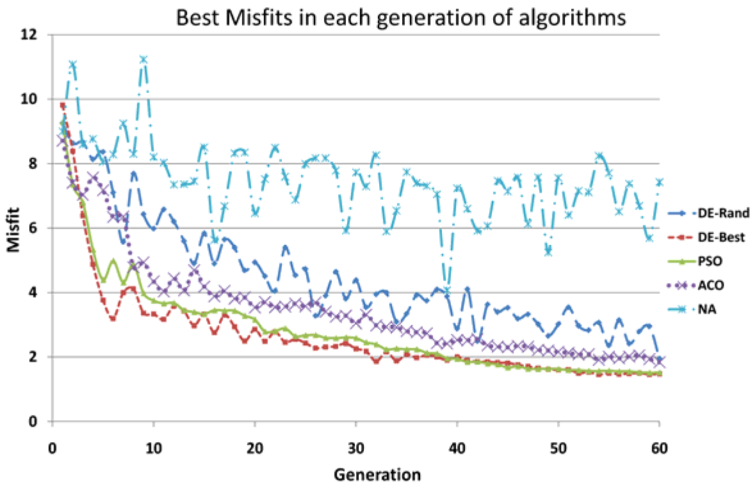


Fig. 14 Comparison of convergence speeds for different algorithms in history matching of PUNQ-S3 model.

As we can see in Fig. 14, PSO and DE-Best algorithms have the fastest convergence. ACO_R and DE-Rand algorithms are ranked next in terms of convergence speed. NA demonstrated a very poor convergence, because the n_s/n_r ratio is set to 1 which corresponds to the most exploratory mode of NA [26]. DE-Best has a fast convergence due to its base vector selection strategy where the best member in the population is selected for adding the difference vector and building new candidate solution. The effect of the base vector selection is also reflected in the convergence of DE-Rand algorithm. As we can see in Fig. 14, this algorithm has more fluctuation in best generational value and a slower convergence in comparison with DE-Best algorithm.

Fig.14 can be used to quantify the computational efficiency of the algorithms. Most of the algorithms (except NA) achieve their 50% efficiency in misfit reduction in first 10 generations (500 simulations) by getting to misfit values less than 6. After this period, the effort of the algorithms is mostly devoted to find global minimum and also discover more local minima in the search space. Table 11 shows the first simulation for each of the algorithms that a model with misfit value less than 4 is found. Also this table shows the number of models that have misfits less than 4 in the next 250 simulations (5 generations) after getting the first model with a misfit below 4. NA is not represented in this table because the minimum misfit value for NA is 4.07. We see that DE-Best obtains the first model in 224th simulation while PSO gets to the first model after 415 simulations. DE-Rand spends more than 1200 simulations to get a model with a misfit value less than 4.

Table 11 Performance of different algorithms in obtaining first model with misfit value less than 4 and number of models obtained with this threshold in next 250 simulations

Algorithm	DE-Rand	DE-Best	PSO	ACO
First simulation with misfit < 4	1285	224	415	790
models with misfit < 4 in next 250 simulations	2	9	16	11

Sampling history trails show how the algorithms navigate through parameter space in each dimension (see Figs. 15-19 for DE, PSO, ACO_R and NA). In each figure variables of best individuals in each generation are plotted versus the generation number. Each figure has 45 tiles showing the 45 parameters we are optimising. We have the scaled parameters value (0, 1) in the vertical axis and generation number (0, 60) in the horizontal axis.

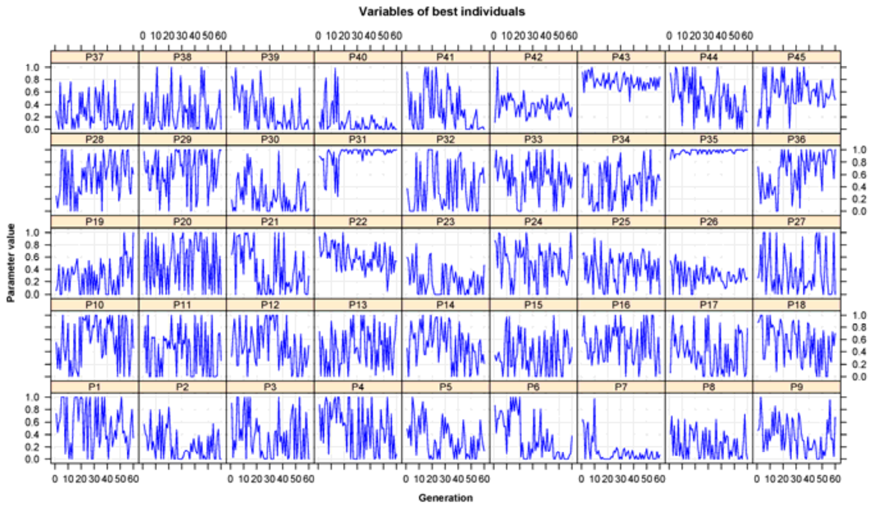


Fig. 15 Sampling history of DE-Rand for PUNQ-S3 model.

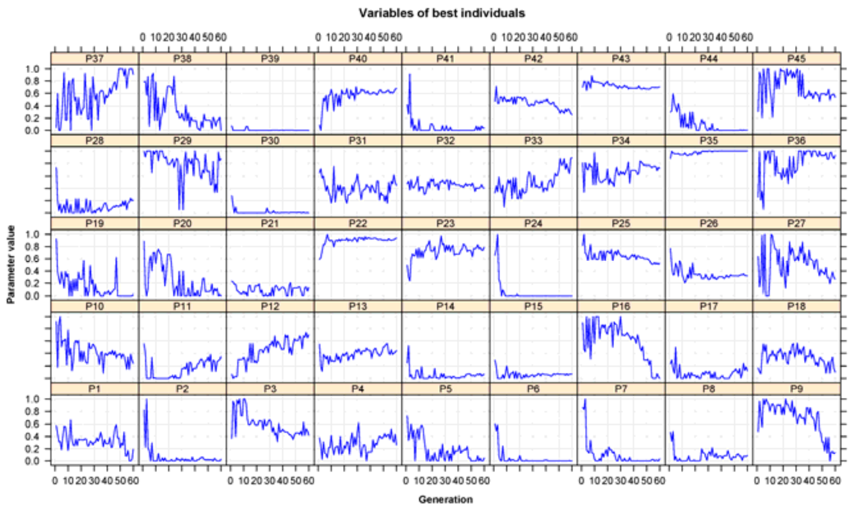


Fig. 16 Sampling history of DE-Best for PUNQ-S3 model.

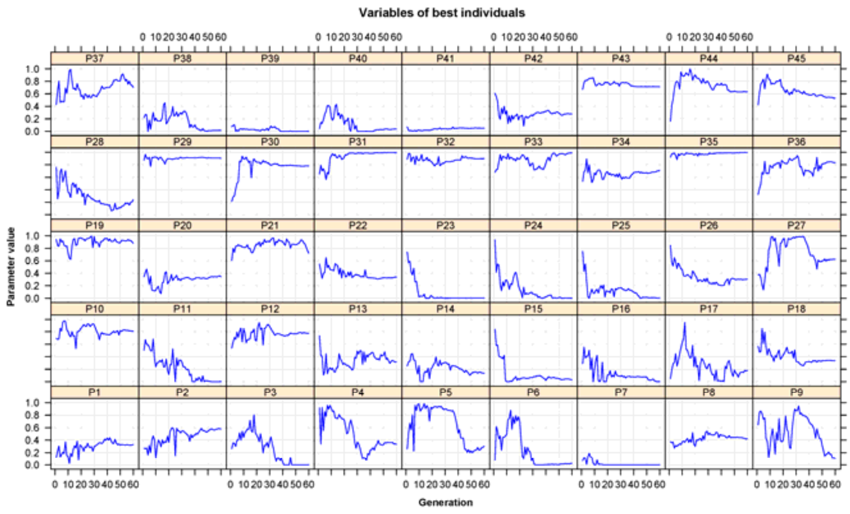


Fig. 17 Sampling history of PSO for PUNQ-S3 model.

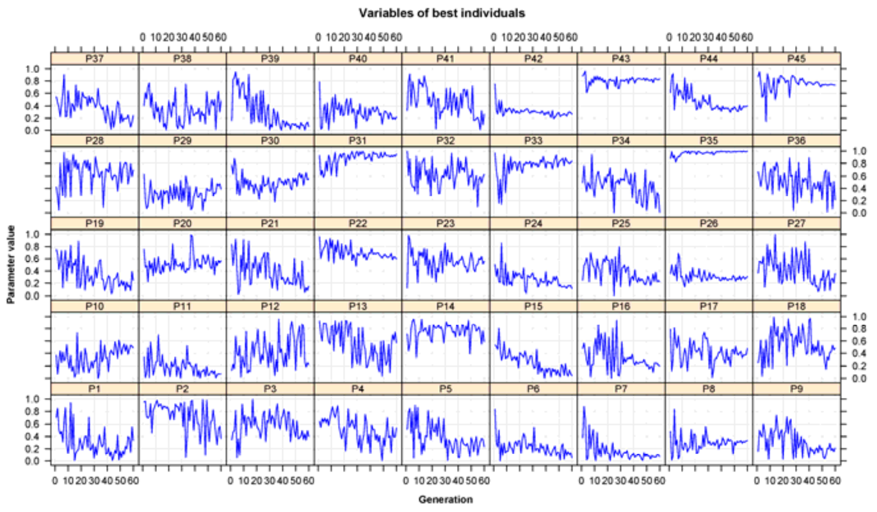


Fig. 18 Sampling history of ACO_R for PUNQ-S3 model.

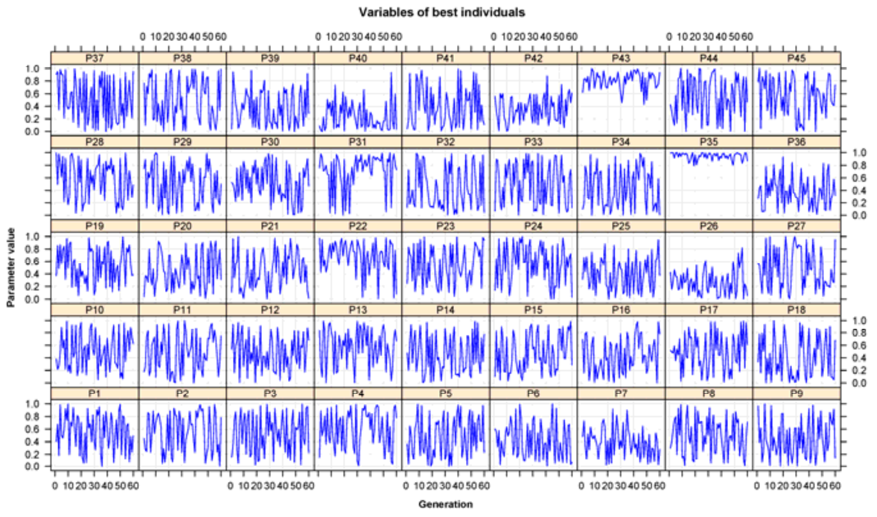


Fig. 19 Sampling history of NA for PUNQ-S3 model.

Sampling history trails give us more insight into the performance of the algorithms. For example let's compare sampling history for PSO and DE-Best algorithms in Figs. 16 and 17. Looking at second row from top in these figures, which show parameters 28-36, we can see that DE-Best has more fluctuations in sampling performance. Also, we can see that DE-Best is mostly sampling from middle range of the parameters 28-36, while PSO keeps its search in upper boundary of these parameters. Sampling figures show that DE-Best and PSO algorithms are obtaining different representations of reservoir model by sampling in different regions of the search space (multiple local minima) during history matching. This diversity of the reservoir models explain different gas oil ratio and water cut predictions in Figs. 12-13 for PSO and DE-Best algorithms. DE-Rand has a wider sampling performance (Fig. 15) in comparison with PSO and DE-Best algorithms. Neighbourhood Algorithm although covers most of the search space, but considering its convergence performance in Fig. 14, it is not successful in locating the good fitting models in the search space in this problem. An ideal situation is that we effectively reduce misfit value during optimisation while keeping the diversity of the obtained solutions. In section 5.2.2 we fill further explore the effect of sampling history in uncertainty quantification.

5.2 Uncertainty of Predictions

In the previous section we successfully showed that evolutionary and swarm intelligence algorithms can be used for finding multiple history-matched models. These algorithms sample from the prior rather than from the posterior. The resulting models have different probabilities which need to be evaluated from inference. Thus, a Markov Chain Monte Carlo (MCMC) technique is applied to the ensemble of models based on the likelihood values computed during the sampling stage.

Here we use NA-Bayes (NAB) algorithm for posterior inference [32]. NAB is a MCMC technique which uses Gibbs sampler to make steps in one dimension of the parameter space at a time. A proxy likelihood surface is also needed to make the inference computationally efficient once the flow simulation is not feasible to compute the likelihood evaluation for every MCMC step. The algorithm is based on the proxy likelihood surface represented by Voronoi cells that are centered around the sampled models in the parameter space. The likelihood within each Voronoi cell is assumed constant and equal to the one computed at the sampling stage for the corresponding model in the centre of the cell. NAB performs resampling with the probability of accepting the Metropolis step, which is proportional to the volume of the Voronoi cell and likelihood. The cell volume is evaluated by MCMC integration. The main benefits of using NAB algorithm for this purpose are:

1) It infers the information from the all models in the ensemble. The idea is that even bad models with high misfit values help to get more information about the uncertainty of predictions.
2) There is no need to run forward reservoir simulations for all the models generated by the sampling algorithm, but only for the ones resampled by NAB. This helps to save computational resources.

As a result, the resampled models from the ensemble at the inference stage are assigned the evaluated posterior probabilities. Prediction confidence intervals are derived from the probability density function based on the multiple model predictions taken with the evaluated posterior probability.

5.2.1 Teal South Reservoir

In Teal South reservoir the history matching is done for 181 days and we extend the forecast for 1247 days. The ensemble of 1380 models generated using each of the algorithms is submitted to the NAB routine and Fig. 20 shows the uncertainty intervals of the prediction made for 1247 days. We can see that the predictions of all methods encapsulate the truth total production after 1247 days. PSO algorithm has the closest value to the truth production in this example.

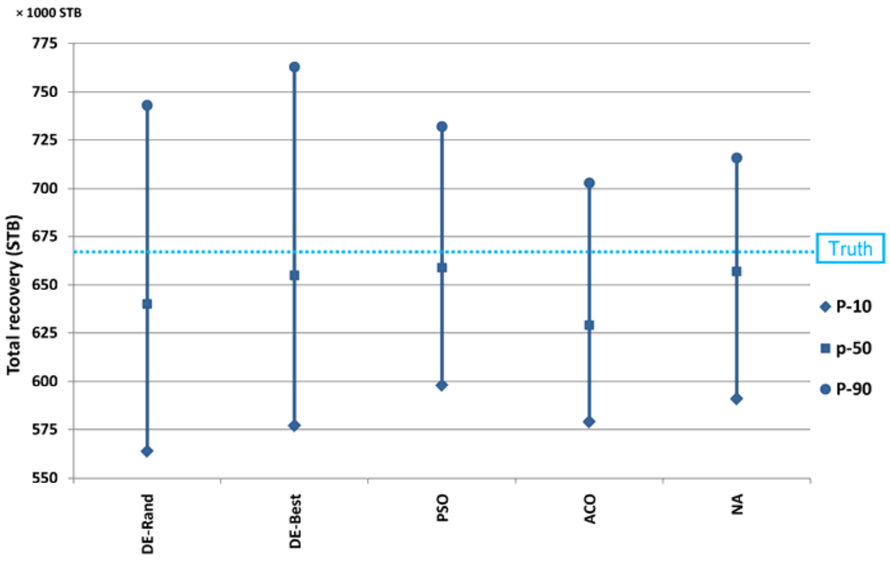


Fig. 20 Uncertainty intervals for total oil recovery after 1247 days in Teal South reservoir model obtained by different algorithms compared with the truth solution.

5.2.2 PUNQ-S3 Model

Prediction of total oil recovery after 16.5 years for PUNQ-S3 model is one of challenges of this problem. It has been shown that many methods give a good history matching result, yet fail to correctly predict the total oil recovery after 16.5 years from this reservoir [28,33].

Figs. 21 and 22 show the resampled models obtained by NAB routine and their match results for BHP of well 1 and cumulative oil production of the PUNQ-S3 reservoir. Resampled models by NAB have lower misfit values and thus better fit to the observations. We use these models and their corresponding posterior probability values to quantify the uncertainty associated with the future predictions.

Figs. 23 and 24 compare the predictions of total oil recovery after 16.5 years obtained by the algorithms used in this work and other approaches reported previously [28].

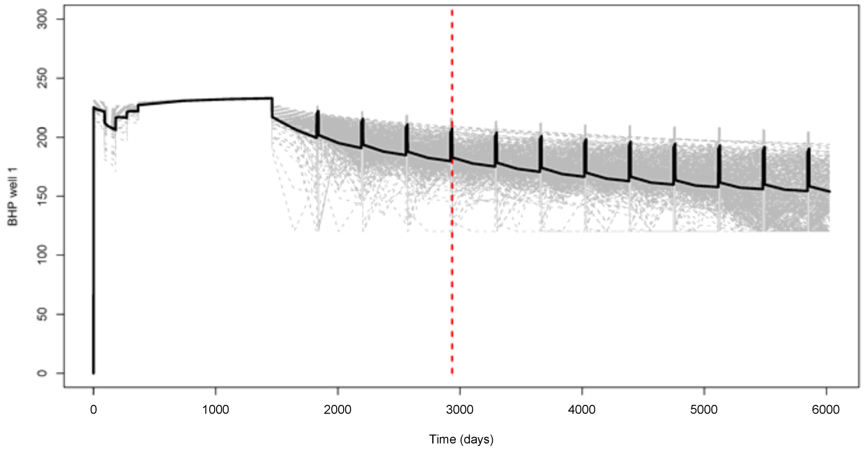


Fig. 21 Resampled modes and their match results for BHP of well 1(vertical dashed line shows the end of history period, solid line represents truth case solution).

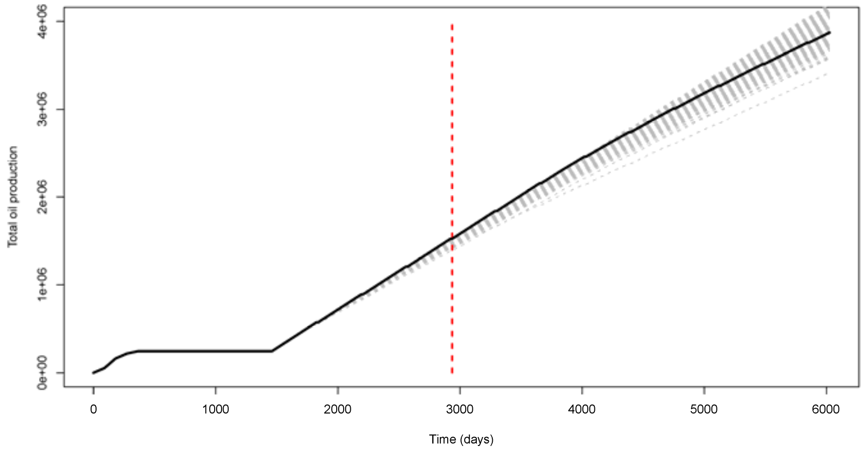


Fig. 22 Resampled modes and their match results for total oil recovery from the field (vertical dashed line shows the end of history period, solid line represents truth case solution).

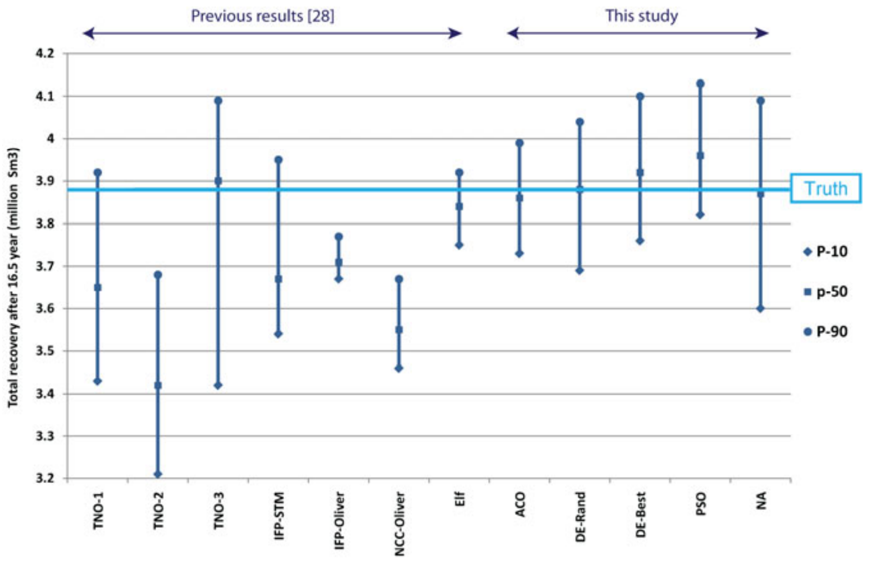


Fig. 23 Uncertainty intervals in PUNQ-S3 model and its comparison with gradient methods and truth solution.

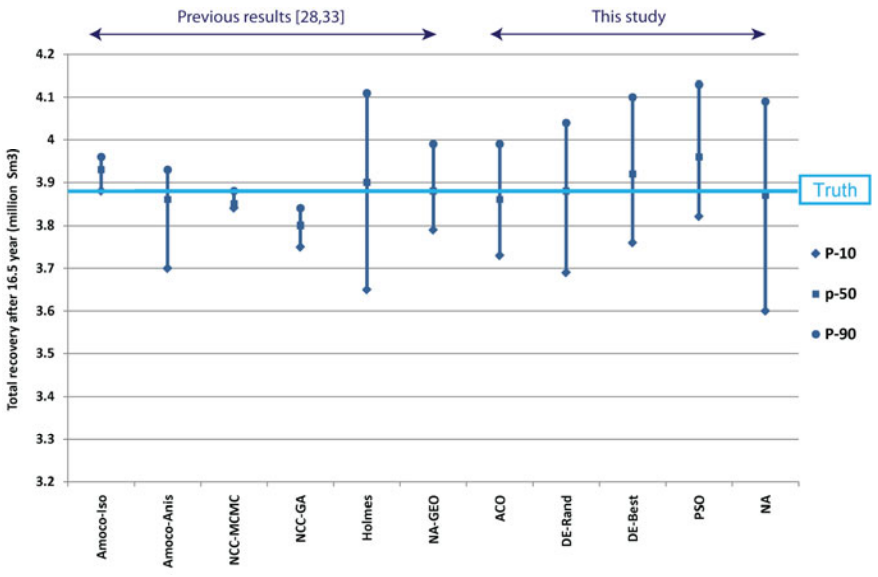


Fig. 24 Uncertainty intervals in PUNQ-S3 model and its comparison with stochastic methods and truth solution.

Methods used for history matching and uncertainty quantification of PUNQ-S3 which are reported in Figs. 23 and 24 are different in the following 3 aspects. The first difference is the parameterisation of the reservoir model, the second is that they use different methods to find good matching models (gradient, stochastic) and the third is the approach they take to quantify the uncertainty. These three aspects are equally important in reservoir engineering studies. In Figs. 23 and 24, we compare the results of published works for prediction of oil recovery in PUNQ-S3 model and our work. Fig. 23 shows this comparison between our work and the approaches that use a gradient-based optimisation technique for finding good fitting models [28] and Fig. 24 compares the stochastic-based methods used in previous publications [28, 33] and this work. Gradient-based methods in this study tend to provide wider and less accurate credible intervals than stochastic algorithms, though some exceptions are possible. Also most of the stochastic algorithms achieve P50 prediction closer to the true solution. However, it is important not to underestimate uncertainty which may be the case when credible intervals are based on too few inferred models.

As we can see from above figures, all algorithms used in this work forecast an uncertainty range which covers the truth total production from PUNQ-S3 model. It is interesting to note that NA, which had a relatively poor performance in history matching, can still give a very good prediction for ultimate oil recovery. PSO and DE-Best slightly over-estimate the oil recovery from reservoir. Uncertainty estimate of DE-Best is closer to the truth solution in comparison with PSO. Looking back to the sampling figures (Figs. 16 and 17), we can see that DE-Best samples a wider range in the search space than PSO. Performance of the algorithms in sampling of the search space is reflected in their ultimate recovery predictions. In PUNQ-S3 case, a more diverse sampling of the search space (DE-Rand) leads to a recovery estimate which is very close to truth value. NA did not find very good fitting regions in the search space, but it is able to make good predictions with wide sampling of the search space.

6 Conclusions

This chapter describes tackling uncertainty quantification problem using recently developed evolutionary and swarm intelligence optimisation algorithms. We describe the Bayesian framework to quantify uncertainty of petroleum reservoir models with multiple history-matched models. We search the space of parameters based on prior beliefs using stochastic optimisation algorithms and evaluate the posterior probability of the models via a Bayesian framework. We have reviewed and compared performance of different advanced adaptive stochastic algorithms: Ant Colony Optimisation, Differential Evolution, Particle Swarm Optimisation and Neighbourhood Algorithm for this purpose. The peculiarities of each algorithm are discussed by describing different flavours and tuning parameters of each algorithm.

The application is illustrated by two case studies: a simple model of a real reservoir (Teal South), and a more complex synthetic model (PUNQ-S3) with more degrees of freedom and variables. All the reviewed algorithms have performed well in both case studies. The predicted uncertainty was compared with the result of the earlier studies where different types of optimisation algorithms have been used, both gradient and stochastic.

The specific conclusions from this study are:

1) For the Teal South reservoir the difference between the performance of the studied algorithms was marginal due to a relatively simple model with just eight free parameters and the univariate objective function based on the data from a single-well.
2) In the PUNQ-S3 high dimensional problem, all algorithms also perform reasonably well in terms of the best history match models obtained. However, DE-Best algorithm and PSO with absorbing boundary strategy achieve slightly better fits then NA and ACO. Also, both DE and PSO demonstrate faster convergence rates to the optimal solution than ACO and NA.
3) Different algorithms are able to find different models of similar history match quality located in different regions of the parameter space. Such diverse models are essential for adequate representation of uncertainty and can be found because of the stochastic nature of the optimisation algorithms. The NA demonstrates a highly explorative behaviour by sampling from all regions of the parameter space to achieve a reasonable fit with good predictions. PSO and DE demonstrate the ability to jump from one local minimum to another, in search for good history matched models.
4) All algorithms provide the confidence bounds which comfortably includes the true solution. The P50 prediction is close enough to the true solution in particular for PSO, DE and NA. Sampling behaviour of the optimisation algorithm has a direct impact on its prediction. Thus, the algorithms performing more exploratively tend to provide wider prediction confidence bounds.

Uncertainty quantification using stochastic evolutionary algorithms in the Bayesian framework can be used for a wide range of prediction models beyond petroleum reservoir modelling.

Acknowledgements. The support for Uncertainty Quantification project at Heriot Watt University is provided by BG, BP, ConocoPhillips and JOGMEC which are gratefully acknowledged here. We also thank Schlumberger GeoQuest for providing ECLIPSE reservoir simulation software.

References

1. Coats, K., Dempsey, J., Henderson, J.: A New Technique for Determining Reservoir Description from Field Performance Data. In: SPE 2344, 43rd SPE Annual Fall Meeting, Houston, Texas, U.S.A, September 29 - October 2 (1968)
2. Slater, G., Durrer, E.: Adjustment of Reservoir Simulation Models to Match Field Performance. In: SPE 2983, SPE 45th Annul Fall Meeting, Houston, Texas, U.S.A, October 4-7 (1970)
3. Bush, M.D., Carter, J.N.: Application of a Modified Genetic Algorithm to Parameter Estimation in Petroleum Industry. In: Intelligent Engineering Systems through Artificial Neural Networks, vol. 6, p. 397. ASME Press, New York (1996)
4. Palatnic, B., Zakirov, L., Haugen, S., van Roosmalen, J.: New Approaches to Multiple History Matching. To be presented at the seventh European Symposium on improved Oil Recovery, Moscow (1993)
5. Tavasolli, Z., Carter, J.N., King, P.: Errors in History Matching, paper 96883. SPE Journal 9(3), 352–361 (2004)
6. Romero, C., Carter, J.N., Gringarten, A.C., Zimmerman, R.W.: A Modified Genetic Algorithm for Reservoir Characterization, paper 64765. In: International Oil and Gas Conference and Exhibition, Beijing, China, November 7-10 (2000)
7. Erbas, D., Christie, M.: Effect of Sampling Strategies on Prediction Uncertainty Estimation, paper 106229. In: SPE Reservoir Simulation Symposium, Houston, Texas, U.S.A, February 26-28 (2007)
8. Christie, M., MacBeth, C., Subbey, S.: Multiple History-Matched Models for Teal South. The Leading Edge 21(3), 286–289 (2002)
9. Mohamed, L., Christie, M., Demyanov, V.: Comparison of Stochastic Sampling Algorithms for Uncertainty Quantification, paper 119139. SPE Journal 15(1), 31–38 (2010)
10. Mohamed, L., Christie, M., Demyanov, V.: Reservoir Model History Matching with Particle Swarms, paper 129152. In: Oil and Gas India Conference and Exhibition, Mumbai, India, January 20-22 (2010)
11. Kathrada, M.: Uncertainty Evaluation of Reservoir Simulation Models using Particle Swarms and Hierarchical Clustering. Heriot Watt University, Edinburgh (2009)
12. Schulze-Riegert, R.W., Axmann, J.K., Haase, O., Rian, D.T., You, Y.: Optimisation Methods for History Matching of Complex Reservoirs, paper 66393. In: SPE Reservoir Simulation Symposium, Houston, Texas, U.S.A, February 11-14 (2001)
13. Hajizadeh, Y., Christie, M., Demyanov, V.: Ant Colony Optimisation for History Matching, paper121193. In: EUROPEC/EAGE Conference and Exhibition, Amsterdam, The Netherlands, June 8-11 (2009)
14. Hajizadeh, Y., Christie, M., Demyanov, V.: Application of Differential Evolution as a New Method for History Matching, paper 127251. In: International Petroleum Engineering Conference, Kuwait, December 14-16 (2009)
15. Christie, M., Demyanov, V., Erbas, V.: Uncertainty Quantification for Porous Media Flows. Journal of Computation Physics 217, 143–158 (2006)
16. Dorigo, M.: Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Milan, Italy (1992)
17. Socha, K., Dorigo, M.: Ant Colony Optimisation for Continuous Domains. European Journal of Operational Research 185(3), 1155–1173 (2008)

18. Madadgar, S., Afshar, A.: An Improved Continuous Ant Algorithm for Optimisation of Water Resource Problems. Water Resource Management 23, 2119–2139 (2009)
19. Schluter, M., Egea, J., Antelo, L., Alonso, A., Banga, R.: An Extended Ant Colony Optimisation Algorithm for Integrated Process and Control System Design. Ind. Eng. Chem. Res 48, 6723–6738 (2009)
20. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimisation over Continuous Spaces, Technical Report for International Computer Science Institute, Berkeley, TR-95-012 (1995)
21. Kennedy, J., Eberhart, R.: Particle Swarm Optimisation. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
22. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimiser. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 69–73. IEEE Press, Piscataway (1998)
23. Kennedy, J.: The Particle Swarm: Social Adaptation of Knowledge. In: Proceedings of the International Conference on Evolutionary Computation, pp. 303–308. IEEE Service Center, Piscataway (1997)
24. Kennedy, J.: The Behaviour of Particles. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, Springer, Heidelberg (1998)
25. Suganthan, P.N.: Particle Swarm Optimiser with Neighbourhood Operator. In: Proceedings of the Congress of Evolutionary Computation, vol. 3, pp. 1958–1962. IEEE Press, Washington D.C., USA (1999)
26. Sambridge, M.: Geophysical Inversion with a Neighbourhood Algorithm - I Searching a Parameter Space. Geophysics J. Int. 138, 479–494 (1999)
27. Subbey, S., Christie, M., Sambridge, M.: A Strategy for Rapid Quantification of Uncertainty in Reservoir Performance Prediction, paper 79678. In: SPE Reservoir Simulation Symposium, Houston, Texas, U.S.A, February 3-5 (2003)
28. Floris, F.J.T., Bush, M.D., Cuypers, M., Roggero, F., Syversveen, A.-R.: Methods for Quantifying the Uncertainty of Production Forecasts. Petroleum Geoscience 7, 87–96 (2001)
29. PUNQ-S3 model, Department of Earth Science and Engineering,, Imperial College London, `http://www3.imperial.ac.uk/earthscienceandengineering/research/perm/punq-s3model` (last accessed April 2010)
30. Boss, C.: Production forecasting with UNcertainty Quantification, Netherlands Institute of Applied Geoscience TNO (1999)
31. Barker, J.W., Cuypers, M., Holden, L.: Quantifying Uncertainty in Production Forecasts: Another Look at the PUNQ-S3 Problem, paper 74707. SPE Journal 6(4), 433–441 (2001)
32. Sambridge, M.: Geophysical Inversion with a Neighbourhood Algorithm - II Appraising the Ensemble. Geophysics Journal International 138, 727–745 (1999)
33. Demyanov, V., Subbey, S., Christie, M.: Neighbourhood Algorithm with Geostatistical Simulations for Uncertainty Quantification Reservoir Modelling: PUNQ-S3 Case study. In: Proceedings of the 9th European Conference on Mathematics in Oil RecoveryECMOR IX 2004, Cannes, France (2004)

Optimization of Multiple Traveling Salesmen Problem by a Novel Representation Based Genetic Algorithm

András Király and János Abonyi

Abstract. The Vehicle Routing Problem (VRP) is a complex combinatorial optimization problem that can be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several requests by the customers, find a route plan for the vehicles with overall minimum route cost (eg. distance traveled by vehicles), which service all the demands. It is well known that multiple Traveling Salesman Problem (mTSP) based algorithms can also be utilized in several VRPs by incorporating some additional constraints, it can be considered as a relaxation of the VRP, with the capacity restrictions removed. The mTSP is a generalization of the well known traveling salesman problem (TSP), where more than one salesman is allowed to be used in the solution. Because of the fact that TSP is already a complex, namely an NP-hard problem, heuristic optimization algorithms, like genetic algorithms (GAs) need to be taken into account. The extension of classical GA tools for mTSP is not a trivial problem, it requires special, interpretable encoding and genetic operators to ensure efficiency. The aim of this chapter is to review how genetic algorithms can be applied to solve these problems, and propose a novel, easily interpretable and problem-oriented representation and operators, that can easily handle constraints on the tour lengths, and the number of salesmen can vary during the evolution. The elaborated heuristic algorithm is demonstrated by a complete realistic example.

András Király
University of Pannonia, Department of Process Engineering,
P.O. Box 158. Veszprém H-8200, Hungary
e-mail: kandras85@gmail.com

János Abonyi
University of Pannonia, Department of Process Engineering,
P.O. Box 158. Veszprém H-8200, Hungary
e-mail: abonyij@fmt.uni-pannon.hu

1 Introduction

The aim of logistics is to get the right materials to the right place at the right time, while optimizing a given performance measure (e.g. minimizing total operating cost) and satisfying a given set of constraints (e.g. time and capacity constraints). Supply chain management includes the planning and management of all activities involved in sourcing, procurement, conversion, and logistics management as well as crucial components of coordination and collaboration. It deals with several problems, like Distribution Network Configuration, Trade-Offs in Logistical Activities, Inventory Management or Distribution Strategy. In most distribution systems goods are transported from various origins to various destinations. For example, many retail chains manage distribution systems in which goods are transported from a number of suppliers to a number of retail stores. It is often economical to consolidate the shipments of various origin-destination pairs and transport such consolidated shipments in the same truck at the same time. There are many ways in which such consolidation can be accomplished. Obviously the challenge is to find the optimal i.e. the best consolidation according to some objective functions. This is a numerical optimization problem, commonly an NP-hard task. In logistics, several types of problems could come up; one of the most remarkable is the set of route planning problems.

One of the most studied problems is the Vehicle Routing Problem (VRP), which is a complex combinatorial optimization problem that can be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several requests by the customers, find a route plan for the vehicles with overall minimum route cost (eg. distance traveled by vehicles) which service all the demands. The complexity of the search space and the number of decision variables makes this problem notoriously difficult. There are several commercial software on the market like IBM's ILOG[1], which is based on a constraints programming optimizer, or the ArcLogistics, a modular system by ESRI[2], which includes a complete framework for optimization and scheduling. VRP problems can be solved by ILOG indirectly by the help of constraint programming. On the other hand, ArcLogistics provides a complete solution for VRP including graphical interface, visualization component or vehicle tracking. The available free softwares present only source codes, or a programming library (like OR-object at http://opsresearch.com/), thus these implementations provide only a programming interface instead of a directly usable software. A relaxation of the VRP is the multiple Traveling Salesman Problem (mTSP), where the capacity of the vehicles is infinite, i.e. capacity restrictions are removed. This means that all the solution procedures and formulations for VRP are also valid for mTSP, by assigning accordingly large capacities to the vehicles (salesmen). If there is only one salesman in the problem, the mTSP reduces to the well-studied Traveling Salesman Problem (TSP). Since TSP is a special case of mTSP, all the

[1] http://www.ilog.com/

[2] http://www.esri.com/software/arclogistics/

formulations and solution procedures remain valid for TSP. The available solvers for TSP are more in number than the others above, a collection of them can be found in [21]. In most cases, the distance between two nodes in the TSP network is the same in both directions. The case where the distance from A to B is not equal to the distance from B to A is called asymmetric TSP. A practical application of an asymmetric TSP is route optimisation using street-level routing (asymmetric due to one-way streets, slip-roads and motorways). Naturally every problem above is an optimization problem where the solution will be optimal, if its cost is minimal. Because of the fact that TSP belongs to the class of NP-hard problems, it is evident that mTSP and VRP are NP-hard problems too. Thus the approximate solutions need to take into account, and the optimization procedures of them could necessitate some heuristic improvements. The number of solvers for mTSP is much smaller than for VRP. Some MATLAB implementation can be found in MATLAB Central[3], and a TSP solver, the Concorde package[4] includes a linear programming based ANSI C code to solve mTSP. These implementations are applicable for classical mTSP problems without special constraints and use approximate method to find the optimal solution.

The main motivation of the research presented in this paper was that there was no available algorithm that is "intelligent" enough to handle constraints on tour lengths, asymmetric distances, and the number of salesmen is not predefined, and can vary during the evolution of the individual solutions, furthermore the representation is so transparent that supports not only the implementation, but the initialization and heuristic fine-tuning of the individual routes. Obviously every optimization is an intelligent procedure, but they can be differentiating from each other by the type of intelligence. The solution can include some heuristics to improve its efficiency, like the appropriate choice of the initial solution set, or decision making according to predefined aspects. Another manifestation of intelligence could be a nature-inspired approach, like a nature-inspired representation of the problem. It is certified in several papers in the literature that a representation elected correctly could improve the effectiveness of the solution procedure. Hence, this chapter presents a novel, quite complex representation of the problem, and proposes genetic operators that can be used for the effective optimization of constrained mTSP problems.

Firstly, the definition of the problem will be given, the general theory of genetic algorithms (GAs), particularly for mTSP, will be discussed, and the known solutions, especially the GA-based approaches to solve mTSP, will be reviewed. It is manifest from the earlier approaches, that the existing representations could not represent the nature of mTSP sufficiently. Following this detailed introduction a more sophisticated representation, and a novel genetic algorithm (GA) based solution using this representation will be proposed. Finally some details about the implementation and utilization of the algorithm will be presented.

[3] http://www.mathworks.com/matlabcentral/fileexchange/?term=tag:"mtsp"

[4] http://www.tsp.gatech.edu/concorde/

2 Mathematical Representations and Optimization Algorithms to Solve mTSP Based Problems

The multiple Traveling Salesman Problem (mTSP) [5] is a generalisation of the well-known Traveling Salesman Problem (TSP) [21]. It consists of determining a set of routes for m salesmen who all starts from, and get back to the home city (depot). In the rest of the section, a problem definition, a review of variations and application areas, and mathematical definitions will be given.

2.1 Problem Definition and Variations

In case of mTSP, a set of n nodes (locations or cities) are given and m salesmen are located at a *single depot* node. The remaining nodes or cities that are to be visited are the *intermediate nodes*. Then, the goal is to find tours for all m salesmen, who all start and end at the central depot, such that each intermediate city is visited exactly once, and the total travelling cost (the cost of visiting all nodes) is minimised. The cost metric can be defined in terms of distance, time, etc.

The possible variations of the problem are as follows:

- *Single depot*: In the single depot case, all salesmen start from and end their tours at a single central depot (see [29]).
- *Multiple depots*: If there exist multiple depots with a number of salesmen located at each, the final destination of the salesmen can be at their original depot or they can return to any depot, with the restriction, that the initial number of salesmen at each depot remains the same after all the travel. The former is the so-called *fixed destination* case, whereas the latter is named as the *nonfixed destination* case (see [29]).
- *Number of salesmen*: The number of salesmen appearing in the problem can be a bounded variable or determined a priori.
- *Fixed charges*: If the number of salesmen in the problem is a bounded variable, then each salesman usually has an associated fixed cost, incurring whenever this salesman is used in the solution. In this case, the minimisation of this bounded variable may be involved in the solution process ([23]).
- *Time windows*: In this variation, certain cities need to be visited in specific time periods, named as *time windows*. This extension of mTSP is referred to as the multiple Traveling Salesman Problem with Time Windows (mTSPTW). mTSPTW has immediate applications in school bus, ship and airline scheduling problems ([11]).
- *Other restrictions*: These additional special restrictions can consist of bounds of the number of cities each salesman can visit, the maximum or minimum distance or travelling duration a salesman travels, or other special constraints.

Further information about the variations of TSP and mTSP can be found in [5] and [21]. In our case, the problem can be defined as an asymmetric multiple Traveling Salesman Problem with Time Windows (mTSPTW) with additional special constraints, where the number of salesmen is an upper bounded variable. The determined constraints are the following:

- maximum number of salesmen
- maximum travelling time / distance of each salesman
- time window at each location

2.2 Practical Applications of mTSP

mTSP is more capable to model real life applications than TSP, since it handles more than one salesman. This section describes some applications to demonstrate the practical importance of the problem. Further information about applications can be found in [21].

One of the principal applications of mTSP is the scheduling of a printing press for a periodical with multi-editions [19]. Five pairs of cylinders exist here between which the paper can roll, and both side of the paper printed concurrently. There exist three type of forms, and the scheduling problem consists of choosing which form will be on which run, and the duration of each run. The inter-city costs is identified by the plate change costs. One of the primary applications is the school bus scheduling, given by Angel at al. [2], which is a modified mTSP with additional side constraints. The goal is to obtain a bus loading pattern while the number of routes is minimized, the total travelling distance by all the buses is kept at minimum, no bus is overloaded, and there exist a maximum travelling time for each route. Another early application was presented by Beltrami et al. [6]. This paper focuses on municipal waste collection, and it addresses the problem of efficiently routing garbage trucks for residential collection. Another application, a crew-scheduling problem can be found in [39], which reports an application for deposit carrying between different branch banks. Here, trusts need to be picked up at banks and returned to the central office by a crew of messengers. The problem is to determine the routes of messengers with a total minimum cost.

mTSP can be effectively used in waste collection problem, which can be found in the work of Bautista et al. in [4]. In this paper, authors solve the NP-hard problem by the help of a constructive metaheuristic approach: the ant colony optimisation method. Mission planning includes the scheduling of autonomous mobile robots, like warehouse automation, military reconnaissance or post-office automation. The mission plan consists of determining the optimal path for each robot to accomplish the goals of the mission, in the smallest possible time. For this problem, a variation of mTSP can be used, where n robots exist and n goals need to be visited by some robot, and a home city also exists to where all the robots need to return. Planning of autonomous robots is modelled as a variant of the mTSP by Yu et al. [42] in the field of cooperative robotics.

A very recent application arises in the design of global navigation system surveying networks [38]. A GNSS is a space-based satellite system which provides coverage for all locations worldwide, and are quite crucial in real-life applications such as early warning and management for disasters, environment and agriculture monitoring, etc. The objective is to determine the geographical positions of unknown points on and above the earth using satellites.

2.3 Mathematical Formulations of mTSP

Usually, mTSP is formulated by different type of integer programming formulations. Before presenting them, some technical definitions will be presented. The mTSP is defined on a graph $G = (V,A)$, where V is the set of n nodes (vertices) and A is the set of arcs (edges). Let $C = (c_{ij})$ be a cost (distance or duration) matrix associated with A. The matrix C is *symmetric* when $c_{ij} = c_{ji}, \forall (i,j) \in A$ and *assymetric* otherwise. If $c_{ij} + c_{jk} \geq c_{ik}, \forall i,j,k \in V$, C is said to satisfy the *triangle inequality*.

2.3.1 Assignment-Based Integer Programming Formulations

mTSP is usually defined using an assignment-based double-index integer linear programming formulation. Let's define the following binary variable:

$$x_{ij} = \begin{cases} 1 \text{ if arc } (i,j) \text{ is used on the tour} \\ 0 \text{ otherwise} \end{cases}$$

A general scheme of the assignment-based directed integer linear programming formulation of the mTSP can be given as follows:

$$\text{minimize} \quad \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} \cdot x_{ij} \tag{1}$$

s.t.

$$\sum_{j=1}^{n} x_{1j} = m, \tag{2}$$

$$\sum_{j=1}^{n} x_{j1} = m, \tag{3}$$

$$\sum_{i=0}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \tag{4}$$

$$\sum_{j=0}^{n} x_{ij} = 1, \quad i = 1, \ldots, n, \tag{5}$$

$$+ \text{ subtour elimination constraints,} \tag{6}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{7}$$

where (4), (5) and (7) are the usual assignment constraints, (2) and (3) ensure that exactly m salesmen start end return back to the central depot (node 0). Constraints (6) are used to forbid subtours, which are degenerate tours that are formed between intermediate nodes, and not connected to the depot. The constraints are the so-called *subtour elimination constraints* (SECs). Several SECs have been proposed for mTSP in the literature, some are discussed in [5].

2.3.2 Modified Assignment-Based Formulations Related to Present Problem

As it was mentioned in section 2.1, the problem which is analysed in this chapter, is more complex than the traditional mTSP problem. It is a so-called mTSPTW with additional constraints, which can be formulated as follows. Let us define the following binary variable:

$$x_{ijk} = \begin{cases} 1 \text{ if arc } (i,j) \text{ is used on the tour of the } k^{th} \text{ salesman} \\ 0 \text{ otherwise} \end{cases}$$

Let's define M as the maximum number of salesmen, and S as the maximum length of any tour in the solution. Furthermore, let's define the cost (distance or duration) matrix associated with A as $C^t = (c^t_{ij})$, where $c^t_{ij} = c_{ij} + c^{tw}_j$, and c_{ij} is the ordinary cost (e.g. distance) of the arc_{ij}, and c^{tw}_j is the cost of the time window. Time window means, that every salesman has to wait in each location, which can be e.g. the duration of loading the goods. Obviously, C^t can't be a symmetric matrix, since in a real life application $c_{ij} \neq c_{ji}, \forall (i,j) \in A$, because of there can exist e.g. one-way roads. Thus, the optimisation problem can be given as follows:

$$\text{minimize} \quad \sum_{i=0}^{n}\sum_{j=0}^{n} c^t_{ij} \cdot \sum_{k=1}^{m} x_{ijk} + m \cdot c_m \tag{8}$$

s.t.

$$\sum_{j=1}^{n}\sum_{k=1}^{m} x_{1jk} = m, \tag{9}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{m} x_{j1k} = m, \tag{10}$$

$$\sum_{i=0}^{n}\sum_{k=1}^{m} x_{ijk} = 1, \quad i = 2,\ldots,n, \tag{11}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{m} x_{ijk} = 1, \quad j = 2,\ldots,n, \tag{12}$$

$$+ \text{ subtour elimination constraints,} \tag{13}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} c^t_{ij} \cdot x_{ijk} \leq S, \quad k = 1,\ldots m, \tag{14}$$

$$x_{ijk} \in \{0,1\}, \forall (i,j) \in A, 1 \leq k \leq m, 1 \leq m \leq M \tag{15}$$

If we use the newly introduced binary variable, equation (1) is altered into equation (8), where the cost of the involvement of a salesmen appears too (c_m). (9) - (12) are equal with equation (2) - (5) using the binary variable x_{ijk}, and (14) ensures that the tour length of each salesmen is under the specified bound, S.

Furthermore, if we want to add penalty for the salesperson who reaches the maximal tour length, the above formalism could change slightly. Let

$$\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}^{t} \cdot x_{ijk} = E_k, \quad k = 1, \dots m, \tag{16}$$

Thus, equation (8) is changed in the following way:

$$\text{minimize} \sum_{k=1}^{m} (E_k + \lambda \cdot \max(E_k - S, 0)) + m \cdot c_m \tag{17}$$

In (17), the penalty is proportional to the tour length of a salesmen above the upper bound S, while the degree of the punishment is determined by the constant λ, which value much depends on the range of c_{ij}. Note that another sort of penalty could be a cutoff of the route of a salesmen who reaches the upper bound (see section 5.2).

2.4 Approaches to Solve mTSP

In the last two decades the traveling salesman problem received quite big attention, and various approaches have proposed to solve the problem, e.g. Branch-and-Bound [13], cutting planes [28], neural network [7] or tabu search [17]. Some of these methods are exact algorithms, while others are near-optimal or approximate algorithms. The exact algorithms usually use integer linear programming approaches with additional constraints.

The mTSP is much less studied like TSP problem. [5] gives a comprehensive review of the known approaches. There are several exact algorithms of the mTSP with relaxation of some constraints of the problem, like [25], which is the first approach to solve the mTSP directly, without any transformation of the TSP. In this problem, each salesman has a fixed cost f, which is activated whenever a salesman is activated in the solution. The solution in [1] is based on Branch-and-Bound algorithm, which is applicable for asymmetric, as well as symmetric problems. Another exact solution method is in [20]. This approach is based on a quasi-assignment relaxation obtained by relaxing the subtour elimination constraints (SECs).

Recent research can be found in [12], where mTSP is optimised by mixed method. Authors combined Particle Swarm Optimization with Ant Colony Optimization to find the best solution of the problem. Another recent solution is presented in [30] where authors used K-Means Clustering, Shrink Wrap Algorithm and Meta-Heuristics to solve the mTSP. A multi-objective approach can be found in [15], where the multiple objective ant colony optimization is used for the bi-criteria TSP.

Lately GAs are also used for optimization of mTSP. The previous GA-based solutions will be discussed in section 3. In the literature there are several examples that a good problem-specific representation can dramatically improve the efficiency of genetic algorithms. A problem-specific individual design can reduce the search-space, and in this case, it is needed to implement special operators which can simulate the nature of the problem. These properties can make the problem-specific genetic algorithm more effective for the given task, and it becomes more easily interpretable.

GAs are direct, random search algorithms, based on the evolutionary model [18], related with Darwin's evolutionary theory. The researches of GAs have begun in the sixties by J.H. Holland [22]. GAs belong to the evolutionary computation (EC) [3] methods, thus their terminology is closely related to biology. Each solution of the problem, or equivalently, each point in the search space is represented by an individual which consists of chromosomes, and chromosomes consist of genes. Individuals constitute a population, which contains all possible solutions. The method is based on the collective learning process of the population. The individuals are improved in the course of iterations by the partway forthcoming operators, like selection, crossover and mutation.

More recently, GAs are successfully implemented to solve TSP [16]. Potvin presents a survey of GA approaches for the general TSP [33]. In case of mTSP, due to its combinatorial complexity, it is necessary to apply some heuristic in the solution, especially in real-sized applications. One of the first heuristic approach were published by Russell [37] and an other procedure is given by Potvin et al. [34]. The algorithm of Hsu et al. [24] presented a Neural Network-based solution. Lately GAs are used for the solution of mTSP too. The first result can be bound to Zhag et al. [43]. Most of the work on solving mTSPs using GAs has focused on the Vehicle Scheduling Problem (VSP) [26, 31]. VSP typically includes additional constraints, like the capacity of a vehicle (it also determines the number of cities each vehicle can visit), or time windows for the duration of loadings.Recent application can be found in [40], where GAs were developed for hot rolling scheduling. There are no constraints on the route lengths of the salesmen, and it introduces a lot of dummy nodes and some additional binary variable, thus it can convert the mTSP into a single TSP and apply a modified GA to solve the problem. You et al. [42] use GAs to solve the mTSP in path planning. A new approach of chromosome representation, the so-called *two-part chromosome* technique can be found in [9], which reduces the size of the search space by the elimination of redundant solutions. According to the referred paper, this representation is the most effective one so far. The related representations for mTSP will be studied in section 3.2.2.

Although the salesmen in mTSP are separated from each other "physically", every previous solutions of mTSP with GA has used a single chromosome to represent a whole solution, i.e. to represent each salesman. This type of representation requires several "transformation steps" to extract the individual tours of the salesmen, and does not support genetic operators defined among solutions with different number of salesmen. Rely on the previous considerations, the chapter introduces a new representation for this problem class, which can separate the salesmen from each other, thus it is more similar to the characteristic of mTSP.

3 Application of Genetic Algorithms to Solve mTSP

3.1 Introduction to Traveling Salesman Specific GAs

GA starts with an initial solution set, which contains individuals created randomly. This is called initial population. The initial step can mightily improve the efficiency of the algorithm, thus a new start strategy can be momentous. The new population is always generated form the actual population's participants by the genetic operators. The generation of new populations is continued until a predefined stop criteria is satisfied.

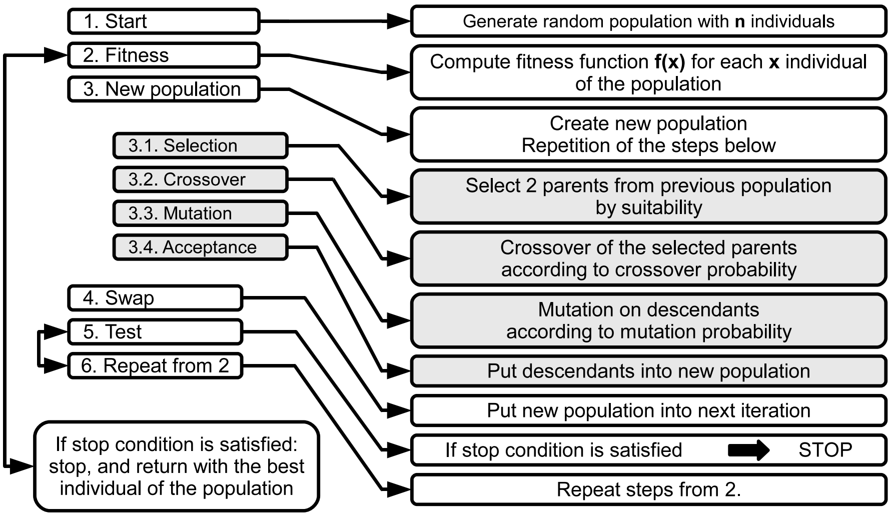

1. Start
2. Fitness
3. New population
3.1. Selection
3.2. Crossover
3.3. Mutation
3.4. Acceptance
4. Swap
5. Test
6. Repeat from 2
If stop condition is satisfied: stop, and return with the best individual of the population
Generate random population with **n** individuals
Compute fitness function **f(x)** for each **x** individual of the population
Create new population Repetition of the steps below
Select 2 parents from previous population by suitability
Crossover of the selected parents according to crossover probability
Mutation on descendants according to mutation probability
Put descendants into new population
Put new population into next iteration
If stop condition is satisfied
STOP
Repeat steps from 2.

Fig. 1 The life cycle of genetic algorithms.

Fig. 1 shows the general case of GA's life cycle. Obviously in a specific problem, this process can be much more complicated, almost in every step specific realization can be required. The first important task is to choose the encoding of the chromosomes, considering crossover and mutation operators. A very important problem is the determination of parents. Several opportunities exist, but most often the algorithm selects the participants with a better attribute with (or with better fitness value) bigger probability. The reason of this consideration is that individuals with better fitness could produce descendants with better properties.

If the new population was composed from the newly created descendants only, the old population's best individual may be lost. To eliminate this deficiency, a new approach, the so-called *elitism* was introduced. This method ensures that the previous population's best individual will get into the new population without any modification, thus the best solution found so far will survive during the whole evolutionary process.

3.2 Encoding

The *encoding* of the problem is the mapping of the *phenotype* to the *genotype*, while decoding is the inverse operator, calculating the parameters of phenotype from the genotype. Genotype codes the genetic information of the individual, this is the representation of the problem. The crossover and mutation operators operates on the genotype. The related encoding techniques to mTSP will be reviewed below.

3.2.1 Permutation Encoding

Permutation encoding is only used in ordering problems, such as Traveling Salesman Problem or task ordering problem. Every chromosome is a string of numbers, which represents numbers in a sequence (see [16]). This technique can be useful for ordering problems, however, special operators are needed to keep the new individuals consistent after crossover and mutation.

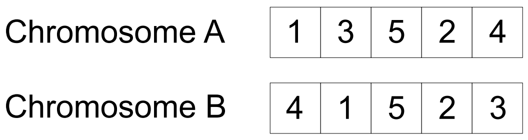


Fig. 2 Permutation encoded chromosomes.

3.2.2 Encoding Related to mTSP

In this section, a review of the existing GA representations related to mTSP will be presented. A simple example route-system is represented in Fig. 3. The following representations will encode this problem into the genes of the chromosomes.

The first approach is the so-called *one chromosome technique* [43], which is illustrated in Fig. 4. It uses a single chromosome of length $(n+m-1)$ (n is the number of locations and m is the number of salesmen). The cities are represented by a permutation of integers from 1 to n. This permutation is divided into m subtours by the insertion of $m-1$ negative integer values, which represents the turn from one salesmen to the next. The cities in a subtour is in the order of the visitation of the salesman. The corresponding route-system is shown in Fig. 3. Using this chromosome representation, there are $(n+m-1)!$ possible solutions of the problem.

Fig. 5 illustrates another approach for chromosome representation of solutions in mTSP (with $n = 15$ and $m = 4$), the so-called *two chromosome technique* [26, 31]. This method requires 2 chromosomes, each of length n. The first chromosome

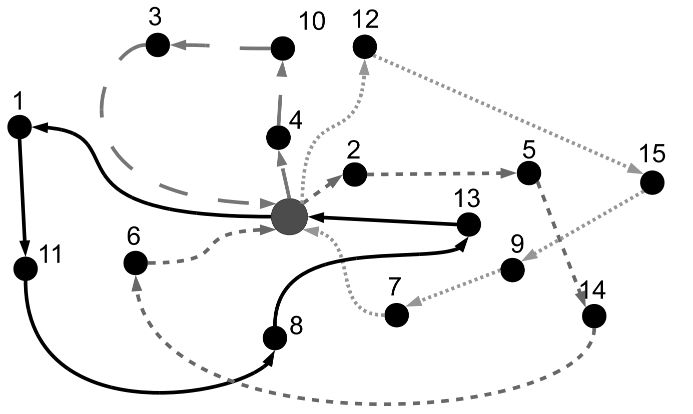


Fig. 3 Example route-system with 15 cities and with 4 salesmen.

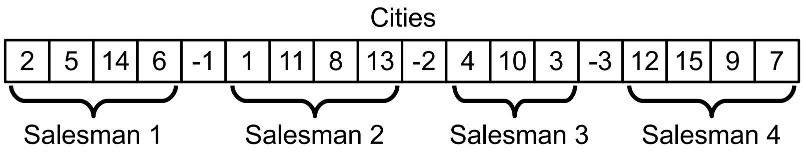


Fig. 4 Example of one chromosome representation for a 15 city mTSP with 4 salesperson ([9]).

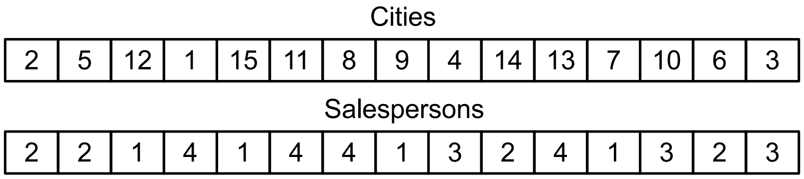


Fig. 5 Example of two chromosome representation for a 15 city mTSP with four salesperson ([9]).

contains a permutation of the n cities, and the second one assigns a salesperson to each locations in the same position of the first chromosome. Using this representation, the search space (i.e. the number of possible solutions) is $n! \cdot m^n$.

A quite new approach of chromosome representation, the so-called *two-part chromosome* technique can be found in [9], which reduces the size of the search space by the elimination of redundant solutions. As Fig. 6 shows this approach represents a solution by a single chromosome. The first part is a permutation of integers

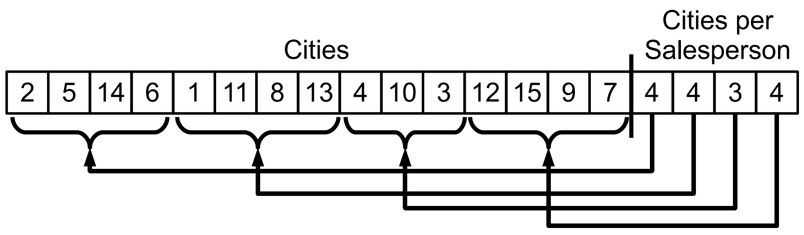


Fig. 6 Example of two-part chromosome representation for a 15 city mTSP with 4 salesmen ([9]).

from 1 to n (number of locations), representing the n cities, and the second part of the chromosome represents the number of cities assigned to each of the m salesperson. The related routes are shown in Fig. 3. According to the cited paper, this representation is the most effective one so far.

3.3 Evaluation of Population

The evaluation of population is done by calculating the fitness value for each individual, which is a real number. Each individual has an *objective-score* which is calculated by the algorithm. The fitness is calculated from the objective score with a possibility of taking the other individuals into account. The objective score is an intrinsic parameter to the optimization problem, thus could not be modified to enhance the evolution process. However, the mapping of objective score to fitness value makes it available to adjust the goodness of an individual for selection.

The type of objective-score to fitness mapping is either *scaling* or *ranking*. In case of scaling, the fitness is a function of the objective-score, while in the case of ranking, the population are sorted according to the objective-score, and the fitness value of the individuals depend on the position in the ranking. Note that in many cases, objective-score and fitness value are identical ($f(x) = x$).

In case of mTSP usually the objective-score and equivalently the fitness value of an individual is the sum of distances (durations) travelled by each salesman. The additional constraints like maximal overall travelling distance refers to this value. If a solution exceeds this constraints, some punishment will be applied, like a proportionately huge fitness value, or the application of a special penal operator.

3.3.1 Operators

A big number of genetic operators can be found in the literature, general ideas are presented in [8, 16, 18], operators for sequencing problems are in [14]. Expressly multi-chromosomal approach can be found in [32], and operators refer to TSP are presented in [27]. In the following sections only a theoretical overview will be given.

Selection

During GA, two kind of selection exist: selection for reproduction and selection for survival. The former selects the individuals from the population for reproduction (parents), and the latter selects the individuals of the new population. This section presents a widely used selection techniques for reproduction, which is used by the novel algorithm presented in section 4. A detailed description of selection schemes is presented in [8].

Tournament Selection

Individuals are chosen from the population randomly for the so-called *tournament*, in which the individual with best fitness is selected as the winner. The number of chosen members for the tournament is determined by the *tournament size* (t), which is between 2 and μ, where μ is the size of the population. The winner can either be removed from, or kept in the population, if it is allowed or disallowed to select an individual multiple times. Tournament selection has a time complexity of $O(N)$. The selection pressure is adjustable through the size of the tournament.

Crossover

Crossover or recombination creates new individuals from the genes of the parents. The easiest way is the one-point crossover, which is shown on the left hand side of Fig. 7. One crossover point is randomly selected (the 3th gene in the example), and the two descendants are created by interchanging the parents' genes after the crossover point. Similarly, the course of two-point crossover (right hand side of Fig. 7) two crossover point is randomly selected, and the genes of the parents are interchanged before and after the crossover points.

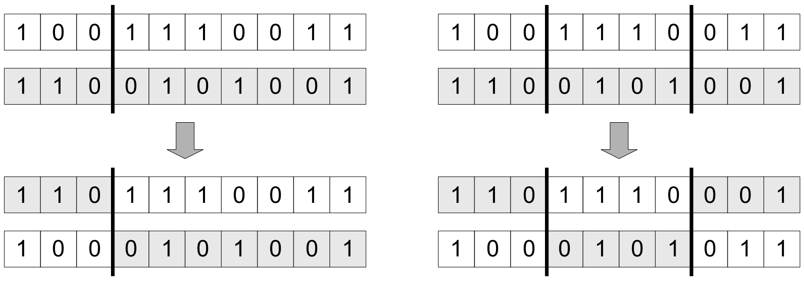


Fig. 7 One-, and two-point crossover of binary encoded individuals.

Mutation

After crossover happened, during the mutation randomly chosen genes are selected and the operator changes their value into an other possible value. An example can be seen in Fig. 8. Mutation can prevent the algorithm from the convergence to a local extrema. Mutation like crossover largely depends on the encoding of the problem.

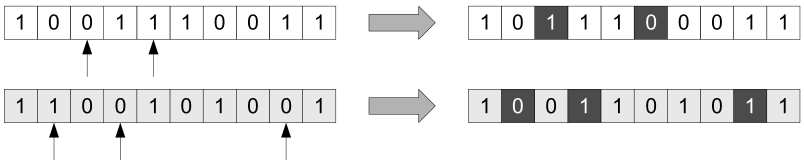


Fig. 8 Mutation of binary encoded individuals.

Genetic algorithms have further parameters, which could effect the efficiency of the GA. *Crossover probability* determines how often the crossover occurs. If no crossover happens, descendants will be equivalent with their parents, otherwise the descendant will consist of the copy of the parents' genetic parts. If the crossover probability is 100%, every offspring will created by crossover, however if it is 0%, the new individuals will be the exact copy of the old population's members (note that it doesn't mean that the two population are equal). It is advisable to transmit the best individuals into the new population without any modification.

Mutation probability determines how often the mutation is used on the offsprings. If no mutation happens, the offspring will be the result of the crossover, or of the copy. If mutation happens, some part of the chromosome will change, in case of 100%, every descendant will change, otherwise (0%) no modification will occur.

The *population size* defines the number of individuals in the population. If it is too small, the algorithm couldn't cover the whole search space. When population size is too big, the GA will slow down.

4 The Proposed Approach to Solve mTSP

There are several representations of mTSP (see Sect. 2.4), like one chromosome technique [43], the two chromosome technique [26, 31] and the latest two-part chromosome technique [9]. As mentioned in the previous section, every GA-based approach for solving the mTSP has used single chromosome for representation so far. The new approach presented here is a so-called multi-chromosome technique, which separates the salesmen from each other thus may provide more effective approach.

Authors should mention that the representation which is presented here is not a "classical" multi-chromosomal representation, because in evolutionary computation, every chromosome of the population represents a global solution of the problem. However, the classical nomenclature of evolutionary computation couldn't give an opportunity to the authors for a proper denomination, since inside an individual, the salesmen are separated from each other. These separations are named as chromosomes (namely a chromosome represents a salesman), as it will be presented in the next sections.

4.1 The Novel Representation for mTSP

The selection and the evaluation of the representation of a problem may have many aspects. Obviously most of the different genetic representations of the same optimization problem could be transformed into each other. For example, the two-part chromosome representation can be easily transformed into the two chromosome representation. However, the transformation can be very expensive computationally. The design of a novel representation can base on several respects. Authors have developed a novel genetic representation for mTSP which is suitable for the implementation of the limit handling, which is easy to initialize, and which makes the local search inside a route of a salesman possible. This novel approach will be presented in the rest of this section.

The multi-chromosome approach is used in notoriously difficult problems to decompose complex solution into simpler components. It was used in mixed integer problem [32] or in order problems [41]. A usage of routing problem optimization can be seen in [35] and a lately solution of a symbolic regression problem in [10]. This section discusses the usage of multi-chromosomal genetic programming in the optimization of mTSP.

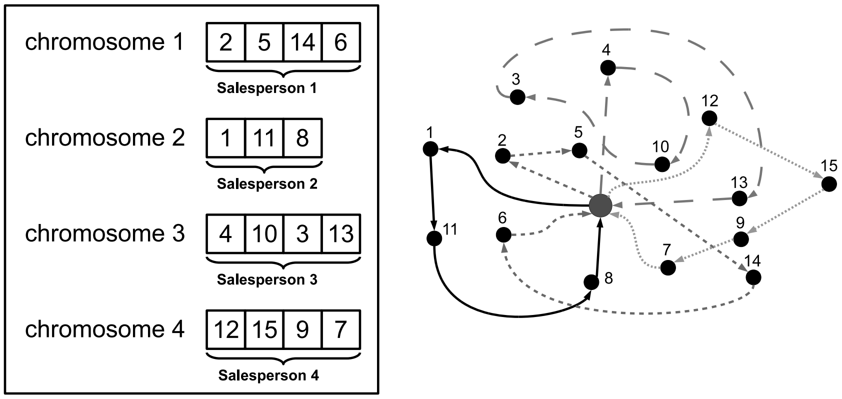


Fig. 9 Example of the multi-chromosome representation for a 15 city mTSP ($n = 15$) with 4 salesperson ($m = 4$).

Fig. 9 illustrates the proposed chromosome representation for mTSP with 15 cities ($n = 15$) and with 4 salesmen ($m = 4$). It shows a single individual of the population, which represents a single solution of the problem. The first chromosome represents the first salesman itself, thus each gene denotes a city (cities were numbered previously, depot is not presented here, it is the first and the last station of each salesman). This is the so-called permutation encoding, because a sequence of numbers are encoded into the genes (section 3.2.1). The order of the cities was defined previously. It is in evidence in the example that salesperson 1 visits 4 cities: city 2, 5, 14 and 6, respectively. In the same way, chromosome 2 represents salesperson 2, which visits city 1, 11 and 8 respectively, and so on. This representation is much similar to the characteristic of the problem, because salesmen are separated from each other "physically". Furthermore, authors consider this representation more easily interpretable.

4.2 *Special Operators for the Proposed Representation*

As it was mentioned in subsection 3.3.1, a lot of genetic operators are presented in the literature. Most of them can be created from other operators, e.g. a multi-chromosomal mutation can be constructed from multiple single-chromosome mutations. The examples in the following subsections have the same properties, but the new representation necessitates the introduction of new genetic operators, like mutation operators. There are two sets of mutation operators, the so-called *In-route mutations* and the *Cross-route mutations*. Authors have implemented several operators for the novel representation, but only an overview of them are given in the following subsections.

4.2.1 In-Route Operators

In-route mutation operators work inside one chromosome. The first operator chooses a random subsection of a chromosome and inverts the order of the genes inside it (Fig. 10). The second operator reverses two randomly chosen genes in the given chromosome (Fig. 11) and the third put a randomly chosen gene into a given place as it can be seen in Fig. 12.

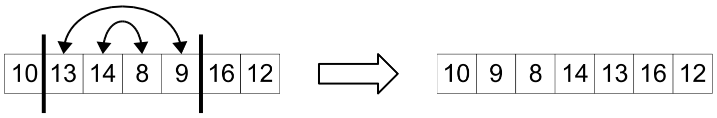


Fig. 10 In-route mutation - gene sequence inversion.

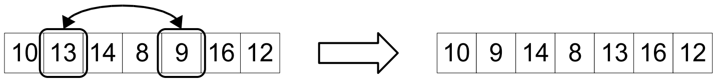


Fig. 11 In-route mutation - gene transposition.

4.2.2 Cross-Route Operators

Cross-route mutation operates on multiple chromosomes. If we think about the distinct chromosomes as individuals, this method could be similar to the regular crossover operator. Fig. 13 illustrates the method when randomly chosen subparts

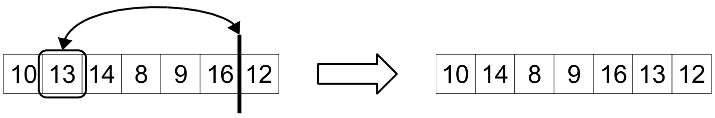


Fig. 12 In-route mutation - gene insertion.

of two chromosomes are transposed. If the length of one of the chosen subsections is equal to zero, the operator could transform into an interpolation.

In Fig. 14 it can be seen a contraction of two chromosomes. In this situation the number of routes in the newly created individual is decreased by one. Fig. 15 illustrates the inverse operation of chromosome contraction. In this case a single chromosome is partitioned into two new chromosomes in the newly created individual, thus the number of salesmen is incremented by one.

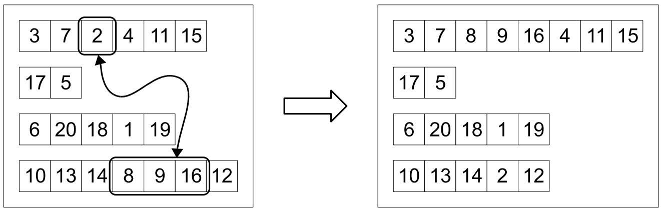


Fig. 13 Cross-route mutation - gene sequence transposition.

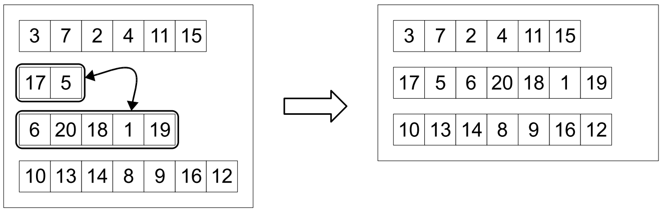


Fig. 14 Cross-route mutation - chromosome contraction.

4.3 Complexity Analysis of the Proposed Representation

Using the multi-chromosome technique for the mTSP reduces the size of the overall search space of the problem. Let the length of the first chromosome be k_1, let the length of the second be k_2 and so on. Of course $\sum_{i=1}^{m} k_i = n$. Determining the genes of the first chromosome is equal to the problem of obtaining an ordered subset of

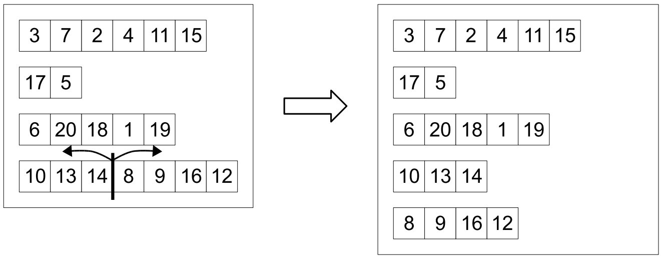


Fig. 15 Cross-route mutation - chromosome partition.

k_1 element from a set of n elements. There are $\frac{n!}{(n-k_1)!}$ distinct assignment. This number is $\frac{(n-k_1)!}{(n-k_1-k_2)!}$ for the second chromosome, and so on. So the total search space of the problem can be formulated as equation (18).

$$\frac{n!}{\cancel{(n-k_1)!}} * \frac{\cancel{(n-k_1)!}}{\cancel{(n-k_1-k_2)!}} * \ldots * \frac{\cancel{(n-k_1-\ldots-k_{m-1})!}}{(n-k_1-\ldots-k_m)!} = \frac{n!}{(n-n)!} = n! \qquad (18)$$

It is necessary to determine the length of each chromosome too. It can be represented as a positive vector of the lengths $(k_1, k_2, \ldots, k_m)$ that must sum to n. There are $\binom{n-1}{m-1}$ distinct positive integer-valued vectors that satisfy this requirement [36]. Thus, the solution space of the new representation is $n!\binom{n-1}{m-1}$. It is equal with the solution space in [9], but this approach is more similar to the characteristic of the mTSP, so it can be more problem-specific therefore more effective, as it will be proven in section 5.3.

5 Implementation of the Proposed Representation

To analyze the new representation, a GA using this approach was developed in MATLAB. The new algorithm is capable to optimise the traditional mTSP problems, furthermore, it can handle the additional constraints and time windows (see Sect. 2). The new approach was compared with the best known one (the two-part chromosome technique) which is available on MATLAB Central[5]. Some example code fragments will be presented in the next sections, but the complete actual MATLAB implementation of the algorithm is available on the website of the authors[6].

[5] http://www.mathworks.com/matlabcentral/
[6] http://www.fmt.uni-pannon.hu/softcomp/

5.1 General Description of the Novel Algorithm

The algorithm requires two input sets, the coordinates of the locations (for visualisation), and the distance matrix which contains travelling distances (in kilometres or in minutes) between any two cities. Furthermore, it requires some parameter determination, like population size, iteration number or the additional constraints. The depot is not presented here, because of complexity reduction (see Sect. 4.3). After these steps, the initial population can be created, which consists of randomly created individuals.

The fitness function simply summarizes the overall route lengths for each salesman inside an individual. The selection is *tournament selection*, where tournament size i.e. the number of individuals who compete for survival is 8. Therefore population size must be divisible by 8. The winner of the tournament is the member with the smallest fitness, this individual is selected for new individual creation, and this member will get into the new population without any modification. The implemented operators will be discussed in the next section.

The penalty of routes which exceed the constraints (too long routes) is realised in an uncommon way. Not a proportionately big fitness value (or a large cost) is assigned, but these chromosomes are split by the chromosome partition operator (Sect. 4.2.2). In this way, too long routes are separated into smaller routes, which do not exceed the constraints (but the number of salesmen is incremented). Because there exists a constraint for the number of the salesmen, the algorithm involves the minimization of this amount, hence this penalty has a remarkable effect on the optimization process.

5.2 Implementation Issues

The creation of initial population is shown in MATLAB code 1. For simplicity, two separated population has been created for routes and for breaks, filled with random data (line 1-6).

In line 8-16 this representation is transformed into the multi-chromosome format (discussed in Sect. 4.1), thus a single *pop* cell array exists. In this *pop* variable, every member is a candidate solution, and every member has several chromosomes (line 14).

MATLAB code 1 Initialiazion of the population

```
1  pop_rte = zeros(pop_size,n);   % population of routes
2  pop_brk = cell(pop_size,1);    % population of breaks
3  for k = 1:pop_size
4       pop_rte(k,:)   = randperm(n);
5       pop_brk{k}     = randbreak();
6  end
7  ⋮
8  for k = 1:pop_size
9      p_rte      = pop_rte(k,:);
10     p_brk      = pop_brk{k};
11     salesmen   = length(p_brk)+1;
12     rng        = [[1 p_brk+1];[p_brk n]]';
13     for j=1:salesmen
14        pop{k}.chromosome{j}=p_rte(rng(j,1):rng(j,2));
15     end
16 end
```

The fitness assignment can see in MATLAB code 2 in line 1-4. *dindist* is an external function which calculates the data of each tour inside a candidate solution. *total_dist* and *total_time* variables contain the overall length and duration of the routes for an individual.

From line 7 a redesign happens, which depends on the constraint violation. If one of the routes in a solution exceeds the constraint of maximal length (line 10-14), the route in question is partitioned into 2 routes by the *chromosome partition* operator (line 12-13, and 16-18). Note that the penetration which was mentioned in equation (17) can be implemented inside the *dindist* function (it is available on the authors website, see footnote 6). Finally the individuals are evaluated again.

MATLAB code 2 Fitness assignment and penalty

```
1  for p = 1:pop_size
2      output{p}      = dindist(pop{p}); %Eval the member of pop
3      total_dist(p)  = output{p}.totalDist;
4      total_time(p)  = output{p}.totalTime;
5  end
6  %Redesign
7  for p = 1 : pop_size
8      dum = [];
9      for i = 1 : length(output{p}.dist)
10         ddum1 = find(cumsum(output{p}.dist{i})>max_route);
11         if size(ddum1)>0
12            dum = [dum pop{p}.chromosome{i}(ddum1(1):end)];
```

```
            pop{p}.chromosome{i}(ddum1(1):end) = [];
        end
    end
    if size(dum)>0
        pop{p}.chromosome{i+1} = dum;
    end
    %Eval the member of pop
    output{p}      = dindist(pop{p});
    total_dist(p) = output{p}.totalDist;
end
```

MATLAB code 3 shows some example of the implemented operators. Line 2 and 3 present In-route mutations, and lines 6-7 show a Cross-route mutation.

MATLAB code 3 Examples of the implemented genetic operators

```
1 % Flip
2 tmp{k}.chromosome{s}(I:J)=tmp{k}.chromosome{s}(J:-1:I);
3 % Swap
4 tmp{k}.chromosome{s}([I J])=tmp{k}.chromosome{s}([J I]);
5 %  Crossover
6 tmp{k}.chromosome{s}=[dum1a dum2b];
7 tmp{k}.gchromosomeene{sc}=[dum2a dum1b];
```

5.3 *Evaluation of the Proposed Approach*

To analyse the effectiveness of the new representation, it was tested by several examples. Two of them are presented here. The first example has created in order to test the algorithm. It is shown in Fig. 16, it consists of 3 loops with 10-10 points, and depot is denoted by a circle. The second example is a realistic problem, where input is defined by a Google Maps map, it can be seen in Fig. 17. It contains 1 depot (lighter marker) and 24 additional locations.

The results are presented in Fig. 18 and in Fig. 19. For comparison, the new representation was compared with an existing MATLAB implementation[7]. This implementation realises the so-called *two-part chromosome* approach [9], which is the best technique for mTSP using GA in many cases so far. Obviously the comparison of two different genetic algorithms couldn't be totally adequate, because the performance of GAs greatly vary as the function of their parameters. Thus, one parameter setting which is appropriate for one algorithm, could be disadvantageous for the other. Hence, only some experimental results will be presented here to show the possible capabilities of the novel representation and algorithm.

[7] This can be downloaded from MATLAB Central (*http://www.mathworks.com/matlabcentral/fileexchange/?term=tag%3A"multiple+traveling+salesmen+problem"*).

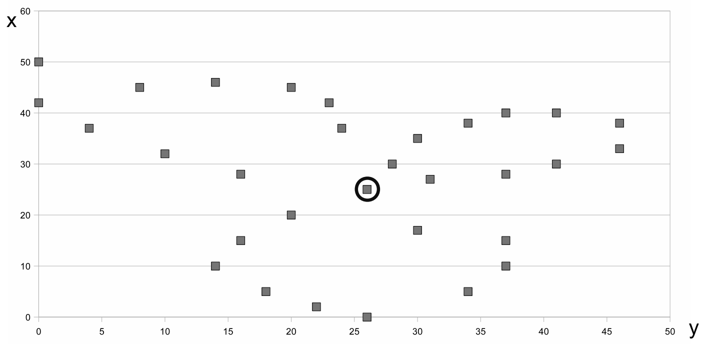


Fig. 16 Simple example for testing.

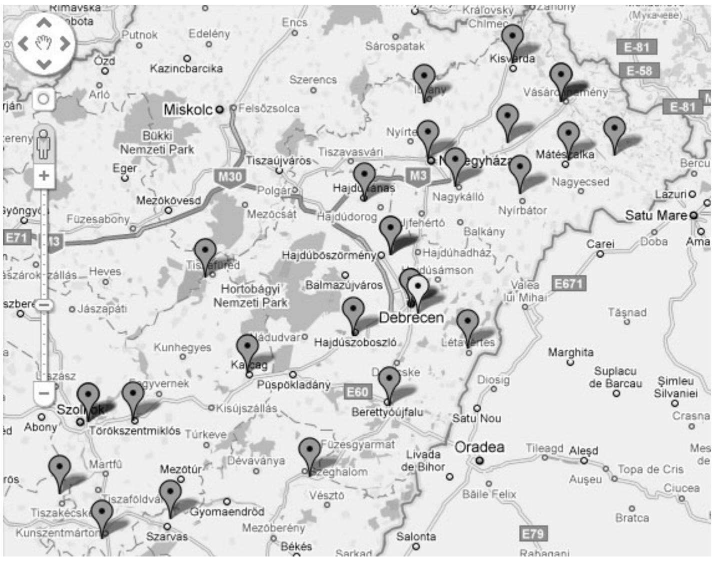


Fig. 17 Example of input data for distance matrix calculation.

In each cases the population size was 160 and iteration number was 500. The figures below show an average result of 60 runs of the algorithm, during a single run, the initial population of the two variety of the algorithm was the same. Figures show unambiguously that the new approach produces better results in these cases. The founded minimum is equal in both cases, but the multi-chromosome technique can converge to the optima faster. In the example above (Fig. 18) multi-chromosome approach needed only 167 iterations to find the optimal value (210.46), while two-part chromosome technique required 314 iterations. At the realistic example (Fig. 19) with a little bit smaller location number, the rate of the iteration numbers are quite the same, the new approach required 60 iterations, while the other method needed 134 to find the optimum. As we expected, at the synthetic example the algorithm

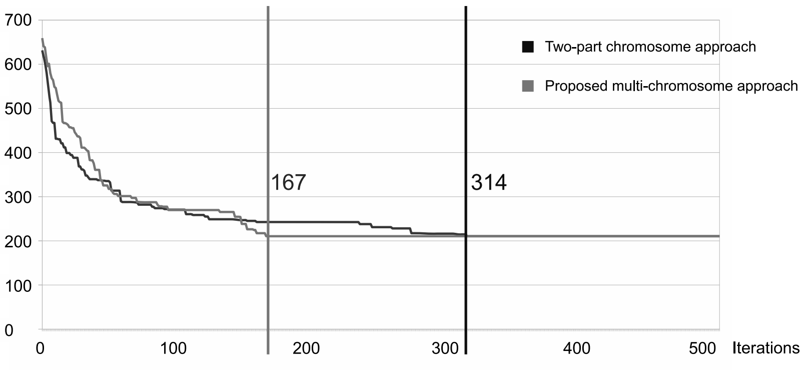


Fig. 18 Result of efficiency analysis - synthetic example.

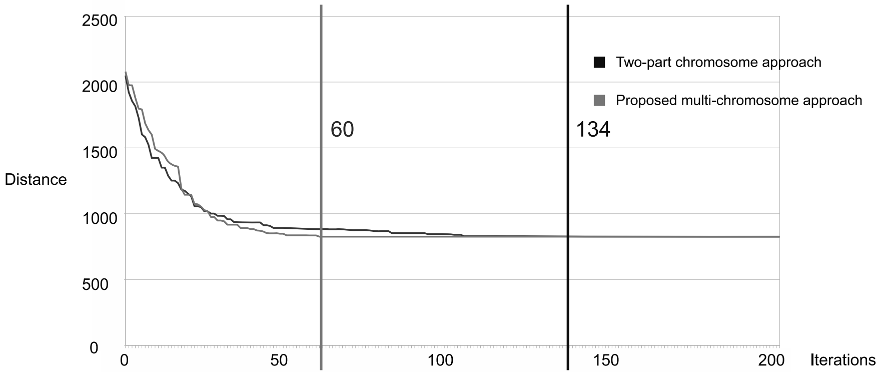


Fig. 19 Result of efficiency analysis - realistic example.

resulted that 3 is the optimal number of salesperson, while in case of the realistic example this number was 2.

The execution time of the algorithms was equal in most cases, which is due to the equal complexity of the representations. Thus, the representations can be comparable only in case of best objective function value per iteration. These and other test issues denote that the novel representation, which is presented in this chapter, can be more effective for the solution of mTSP problems with GAs, than any other approaches so far.

Summarised, a novel representation based GA was developed to solve the mTSP, which solution shows a more efficient approach than the previous ones, proved by several test issues. Furthermore, the new algorithm can handle the constraints for the routes and the time windows for the locations too, as well as it's representation is more similar for the characteristic of the problem than the ones until now, thus it can be more easily understandable and realisable.

6 Application Studies

In this section a whole process of a real problem's solution will be presented. The problem is given in a Google Maps map, and the final output is a route system defined by a Google Maps map also.

The first step is the determination of the distance matrix. The input data is given by a map as it can see in Fig. 20 and a portion of the resulted distance table is shown in Table 1. It contains 25 locations (with the depot). The task is to determine the optimal routes for these locations with the following constraints:

- maximum number of salesmen is 5
- maximum travelling distance of each salesman is 450 km

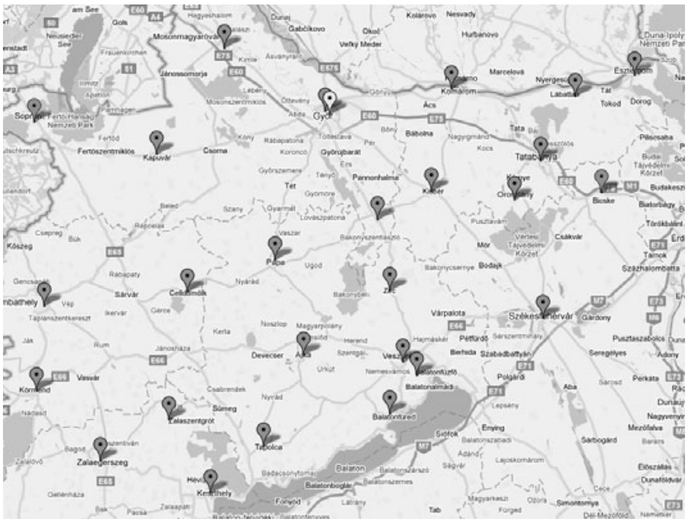


Fig. 20 The map of the studied application (initial input).

Table 1 Example distance table - kilometres.

Kilometres	Adony	Celldömölk	Kapuvár
Adony	0	169.89	147.53
Celldömölk	169.41	0	44.42
Kapuvár	146.56	44.43	0

After distance matrix determination the algorithm computes the solution with the new representation. The GA ran with population size 320 and it did 200 iterations. The result of the optimisation is shown on the upper side of Fig. 21.

It resulted that 4 salesman is enough to satisfy the constraints. With the visualiser component we can visualise the results, as it is shown on the lower side of Fig. 21.

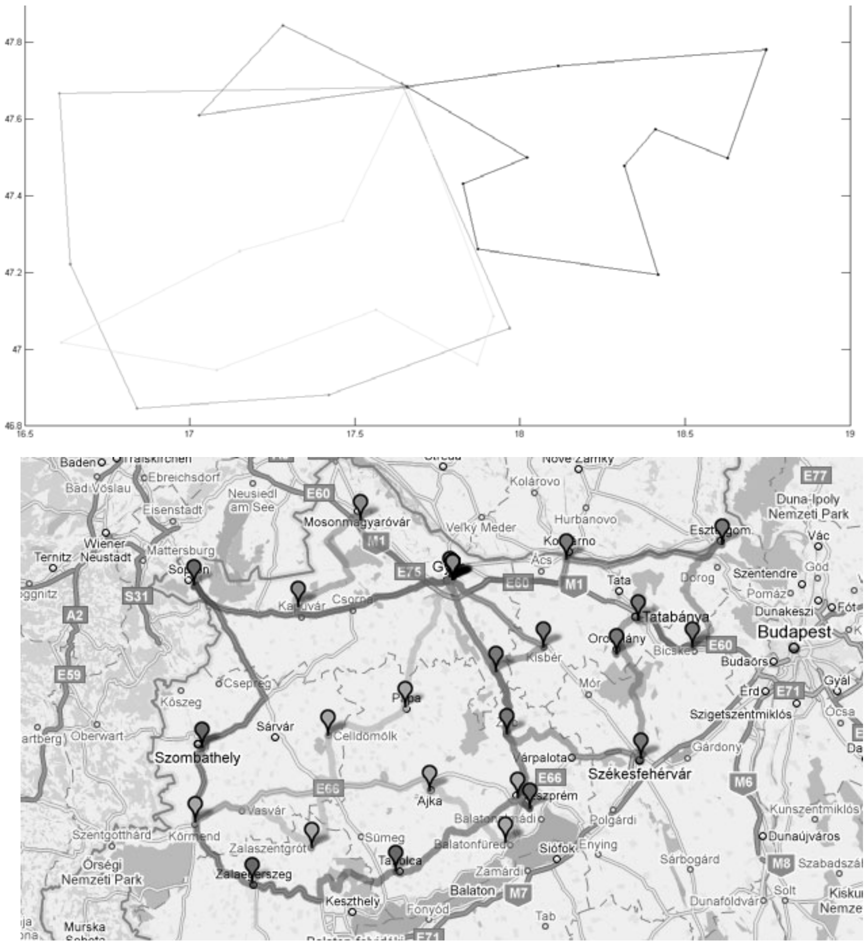


Fig. 21 Results of the optimization by MATLAB and by the Visualiser component for 25 locations with at most 5 salesmen and at most 450 km tour length per salesman.

The length of the routes are 364 km, 424 km, 398 km and 149 km respectively, i.e. they satisfy the constraints, thus the algorithm provided a feasible solution of the problem.

7 Conclusions

The Vehicle Routing Problem (VRP) is a complex combinatorial optimization problem arising at different type of engineering and logistic systems. The multiple Traveling Salesmen problem (mTSP) can be considered as a relaxation of the VRP, with the capacity restrictions removed. The modified mTSP problem with additional constraints on the tour length of the individual salesman, asymmetric distances, unknown and varying number of salesman was introduced and solved by a novel approach. The literature review has showed unambiguously that the existing representations for solving mTSP by the help of GA uses only one chromosome to represent the whole problem, although the nature of the problem could necessitate the separation of the salesmen on chromosome design level. This observation motivated the authors to introduce a novel representation in the individual design, where a separate chromosome is assigned to all salesmen. The approach presented here is innovative in the representation of individuals, in the handling of the constraints, and it gives a whole methodology to solve an NP-hard problem, the mTSP. This novel approach, the so-called multi-chromosome technique was presented for solving mTSPs, and the new representation proved to be more effective in terms of flexibility, complexity and transparency, and also in efficiency than the previous methods. The algorithm was implemented in MATLAB and integrated with Google Maps to provide a framework for distance calculation, definition of the initial routes and visualization of the resulted solutions. Clearly, holistic testing is needed to offer a valid opinion about the efficiency of the novel method. In this chapter some illustrative application examples illustrate that the methodology presented in the chapter supports a fast, near-optimal solution for route planning, setting out from a single map and from constraints defined on the number and tour lengths of the available salesmen.

References

[1] Ali, A.I., Kennington, J.L.: The asymmetric m-traveling salesmen problem: a duality based branch-and-bound algorithm. Discrete Applied Mathematics 13, 259–276 (1986)

[2] Angel, R.D., Caudle, W.L., Noonan, R., Whinston, A.: Computer-assisted school bus scheduling. Management Science 18(6), 279–288 (1972)

[3] Back, T., Fogel, D.B., Michalewicz, Z.: Handbook of evolutionary computation. IOP Publishing Ltd (1997)

[4] Bautista, J., Fernández, E., Pereira, J.: Solving an urban waste collection problem using ants heuristics. Computers & OR 35(9), 3020–3033 (2008)

[5] Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 34, 209–219 (2006)

[6] Beltrami, E.J.B.: Networks and vehicle routing for municipal waste collection. Networks 4(1), 65–94 (1972)

[7] Bhide, S., John, N., Kabuka, M.R.: A boolean neural network approach for the traveling salesman problem. IEEE Transactions on Computers 42(10), 1271 (1993)

[8] Blickle, T., Thiele, L.: A comparison of selection schemes used in evolutionary algorithms. Evolutionary Computation 4(4), 361–394 (1996)

[9] Carter, A.E., Ragsdale, C.T.: A new approach to solving the multiple traveling salesperson problem using genetic algorithms. European Journal of Operational Research 175, 246–257 (2006)
[10] Cavill, R., Smith, S., Tyrrell, A.: Multi-chromosomal genetic programming. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1753–1759. ACM, New York (2005)
[11] Desrosier, J., Sauve, M., Soumis, F.: Lagrangian relaxation methods for solving the minimum fleet size multiple traveling salesman problem with time windows. Management Science 34(8), 1005–1022 (1988)
[12] Feng, H., Bao, J., Jin, Y.: Particle swarm optimization combined with ant colony optimization for the multiple traveling salesman problem. Materials science forum, Trans. Tech. 626, 717–722 (2009)
[13] Finke, G.: Network flow based branch and bound method for asymmetric traveling salesman problems. In: Symposium, X.I. (ed.) on Operations Research, Darmstadt, pp. 117–119 (1986)
[14] Fox, B., McMahon, M.: Genetic operators for sequencing problems. In: Rawlins, G.J. (ed.) Foundations of Genetic Algorithms, pp. 284–300. Morgan Kaufmann, San Francisco (1991)
[15] Garcia-Martinez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp. European Journal of Operational Research 180(1), 116–148 (2007)
[16] Gen, M., Cheng, R.: Genetic algorithms and engineering design. John Wiley and Sons, Inc., New York (1997)
[17] Glover, F.: Artificial intelligence, heuristic frameworks and tabu search. Managerial and Decision Economics 11(5), 365–375 (1990)
[18] Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
[19] Gorenstein, S.: Printing press scheduling for multi-edition periodicals. Management Science 16(6), 373–383 (1970)
[20] Gromicho, J., Paixão, J., Bronco, I.: Exact solution of multiple traveling salesman problems. Combinatorial optimization: new frontiers in theory and practice, pp. 291–292 (1992)
[21] Gutin, G., Punnen, A.P.: The Traveling Salesman Problem and Its Variations. Combinatorial Optimization. Kluwer Academic Publishers, Dordrecht (2002)
[22] Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Cambridge (1975)
[23] Hong, S., Padberg, M.W.: Note on the symmetric multiple traveling salesman problem with fixed charges. Operations Research 25(5), 871–874 (1977)
[24] Hsu, C.-Y., Tsai, M.-H., Chen, W.-M.: A study of feature-mapped approach to the multiple travelling salesmen problem. In: IEEE International Symposium on Circuits and Systems, vol. 3, pp. 1589–1592 (1991)

[25] Laporte, G., Nobert, Y.: A cutting planes algorithm for the m-salesmen problem. Journal of the Operational Research Society 31, 1017–1023 (1980)
[26] Malmborg, C.J.: A genetic algorithm for service level based vehicle scheduling. European Journal of Operational Research 93(1), 121–134 (1996)
[27] Mathias, K., Whitley, D.: Genetic operators, the fitness landscape and the traveling salesman problem. Parallel Problem Solving from Nature 2, 219–228 (1992)
[28] Miliotis, P.: Using cutting planes to solve the symmetric travelling salesman problem. Mathematical Programming 15(1), 177–188 (1978)
[29] Nagy, G., Salhi, S.: Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. European Journal of Operational Research 162(1), 126–141 (2005)
[30] Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R., Parthiban, P.: Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics. International Journal of Nonlinear Science 8(4), 480–487 (2009)
[31] Park, Y.B.: A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. International Journal of Productions Economics 73(2), 175–188 (2001)
[32] Pierrot, H.J., Hinterding, R.: Multi-chromosomal genetic programming. In: Sattar, A. (ed.) Canadian AI 1997. LNCS, vol. 1342, pp. 137–146. Springer, Heidelberg (1997)
[33] Potvin, J.Y.: Genetic algorithms for the traveling salesman problem. Annals of Operations Research 63(3), 337–370 (1996)
[34] Potvin, J.Y.P., Lapalme, G., Rousseau, J.: A generalized k-opt exchange procedure for the mtsp. INFOR 27, 474–481 (1989)
[35] Ronald, S., Kirkby, S.: Compound optimization. solving transport and routing problems with a multi-chromosome genetic algorithm. In: The 1998 IEEE International Conference on Evolutionary Computation, ICEC 1998, pp. 365–370 (1998)
[36] Ross, S.M.: Introduction to Probability Models. Academic Press, New York (1984)
[37] Russell, R.A.: An effective heuristic for the m-tour traveling salesman problem with some side conditions. Operations Research 25(3), 517–524 (1977)
[38] Saleh, H.A., Chelouah, R.: The design of the global navigation satellite system surveying networks using genetic algorithms. Engineering Applications of Artificial Intelligence 17(1), 111–122 (2003)
[39] Svestka, J.A., Huckfeldt, V.E.: Computational experience with an m-salesman traveling salesman algorithm. Management Science 19(7), 790–799 (1973)
[40] Tanga, L., Liu, J., Rongc, A., Yanga, Z.: A multiple traveling salesman problem model for hot rolling scheduling in shangai baoshan iron & steel complex. European Journal of Operational Research 124(2), 267–282 (2000)
[41] Yoshiji, F., Yuki, A., Tsuyoshi, Y.: Applying the genetic algorithm with multi-chromosomes to order problems. In: Proceedings of the Annual Conference of JSAI, vol. 13, pp. 468–471 (2001)
[42] Yu, Z., Jinhai, L., Guochang, G., Rubo, Z., Haiyan, Y.: An implementation of evolutionary computation for path planning of cooperative mobile robots. In: Proceedings of the 4th World Congress on Intelligent Control and Automation, pp. 1798–1802 (2002)
[43] Zhang, T., Gruver, W., Smith, M.: Team scheduling by genetic search. In: Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials, vol. 2, pp. 839–844 (1999)

Part III

Applications

Out-of-the-Box and Custom Implementation of Metaheuristics. A Case Study: The Vehicle Routing Problem with Stochastic Demand

Paola Pellegrini and Mauro Birattari

Abstract. Metaheuristics are a class of effective algorithms for optimization problems. A basic implementation of a metaheuristic typically requires rather little development effort. With a significantly larger investment in the design, implementation, and fine-tuning, metaheuristics can often produce state-of-the-art results. According to the amount of development effort, we say that an implementation of a metaheuristic is either an *out-of-the-box* version or a *custom* one. The possibility of implementing metaheuristics in such a flexible way is one of the major strengths of these algorithms. Nonetheless, it also hides some possible catches. In particular, it should be noticed that results obtained with *out-of-the-box* implementations cannot be always generalized to *custom* ones, and vice versa. The goal of this analysis is to stress that these two ways of using metaheuristics are different. As a case study, we focus on the vehicle routing problem with stochastic demand and on five among the most successful metaheuristics—namely, tabu search, simulated annealing, genetic algorithms, iterated local search, and ant colony optimization. We show that the relative performance of these algorithms strongly varies whether one considers *out-of-the-box* implementations or *custom* ones, in which the parameters are accurately fine-tuned. Moreover, we underline the relevance of clearly stating the framework in which the results reported in the literature have been obtained. To this aim, we consider also an implementation of the same algorithms as described in the literature.

Paola Pellegrini
Department of Applied Mathematics, Università Ca' Foscari Venezia,
Cannaregio 873, 30121, Venice, Italy
e-mail: paolap@unive.it

Mauro Birattari
IRIDIA, CoDE, Université Libre de Bruxelles, 50, Av. F. Roosevelt,
CP 194/6, B-1050 Brussels, Belgium
e-mail: mbiro@ulb.ac.be

1 Introduction

The term *metaheuristics* [1] is nowadays widely adopted for designating a class of approaches to tackle optimization problems.

A metaheuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems.

Dorigo and Stützle, 2004 [2, p. 25]

The generality of metaheuristics and the ease with which they can be applied to the most diverse combinatorial optimization problems is definitely the main reason for their success. Indeed, compared to exact algorithms and problem-specific heuristics, metaheuristics typically require a much lower design and implementation effort. This is particularly true if one does not necessarily aim at state-of-the-art results but has the main goal of obtaining a fairly good performance, while minimizing the development costs. In these cases, an *out-of-the-box* implementation of a metaheuristic is typically the solution of choice for many practitioners. On the other hand, in a number of applications it has been shown that state-of-the-art performance can be obtained through metaheuristics, provided that a *custom* version is developed by taking extra care in the design, implementation, and fine-tuning. This, quite naturally, implies higher development costs.

This flexibility of metaheuristics is definitely one of their appealing traits: In practical applications, one can start with an *out-of-the-box* version of a metaheuristic for quickly having some preliminary results and for gaining a deeper understanding of the problem at hand. Then one can move to a *custom* version for obtaining a better performance without having to switch to a completely different technology.

Nonetheless, the fact that metaheuristics can be flexibly used either in their *out-of-the-box* or *custom* versions, can be reason of misunderstanding. Indeed, results obtained with *out-of-the-box* implementations do not always generalize to *custom* ones, and vice versa. In particular, it could well happen that, as we show in the case study proposed in this work, a metaheuristic M_1 performs better than a metaheuristic M_2 on a given problem when *out-of-the-box* versions of M_1 and M_2 are considered; whereas M_2 performs better that M_1 on the very same problem when *custom* versions are considered.

This issue is unfortunately overlooked in the literature: Many research papers propose comparisons of metaheuristics without providing any measure of the development effort devoted to the algorithms under analysis or, in other words, without clearly stating whether the metaheuristics considered are *out-of-the-box* versions or rather high-performing *custom* versions. Without this piece of information, the usefulness of these comparisons is somehow impaired[1].

The lack of specification about the context in which empirical studies are performed can be partially justified by the fact that, admittedly, measuring the amount of development effort is not a simple and well-defined task. Much of the ambiguity

[1] In a similar way the performance assessment method used may have an impact on the relative performance of metaheuristics. For the analysis of this impact see for example [3, 4, 5, 6, 7].

comes from the fact that there is no such thing as the *standard developer*: What costs a great effort to somebody with limited experience in the domain, might be effortless for a seasoned practitioner. The issue is further complicated by the fact that researchers and practitioners often specialize on one metaheuristic (or on few). For example, if an expert in genetic algorithms devotes the same time and attentions to the development of a genetic algorithm and of a tabu search, the resulting algorithms will have a relative performance that is expectedly much different from the one that would be obtained if the algorithms had been developed by a tabu search expert.

Following the preliminary analysis proposed by Pellegrini and Birattari [8], in this study we show the difference that may result between two experimental studies, one performed in the *out-of-the-box* context, and the other in the *custom* one. Moreover, we highlight the implications of using algorithms as they are described in the literature. In this case, the question is whether it makes sense to use parameters that have been selected as reasonably high performing in some context that is possibly different from the one under analysis. To this aim, we consider as a case study the vehicle routing problem with stochastic demand, and five of the most successful metaheuristics—namely, tabu search, simulated annealing, genetic algorithm, iterated local search, and ant colony optimization. The goal is to show that the relative performance of the above metaheuristics depends on the implementations considered. With this work, we wish to draw the attention of the research community on this issue and contribute to establish a better practice for the empirical analysis and comparison of metaheuristics.

What we wish to underline is that the scope of the studies should be made clear. Different researches may have different goals, that may justify the use of either *out-of-the-box* or *custom* versions of metaheuristics. Nonetheless, we should be aware of the fact that the results achieved may not be generalizable. If we use for our experiments an implementation that was built with a goal that is different from ours, we may be mislead. For supporting this conclusion, we will consider also implementations of the five metaheuristics as they are described in the literature. We will show that the relative performance of these versions is different from the one of both the *out-of-the-box* and the *custom* versions of the same metaheuristics.

In order to attenuate the problem concerning the different ability of a single designer in implementing various approaches, we consider the implementations of the five metaheuristics produced within the *Metaheuristics Network*,[2] a EU funded research project started in 2000 and accomplished in 2004. In the *Metaheuristics Network*, five academic groups and two companies, each specialized in the development and application of one or more of the above metaheuristics, joined their research efforts with the aim of gathering a deeper insight into the theory and practice of metaheuristics. For a detailed description of the metaheuristics developed by the *Metaheuristics Network* for the vehicle routing problem with stochastic demand, we refer the reader to Bianchi et al. [9]. The availability of this reference makes the

[2] `http://www.metaheuristics.net/`

vehicle routing problem with stochastic demand, and the five metaheuristics above mentioned, particularly suitable to our analysis.

In our analysis, these implementations are considered as *black-box* metaheuristics: By modifying their parameters, we obtain the *out-of-the-box*, the *custom*, and what we call the *literature* versions. The first ones are obtained by randomly drawing the parameters from a defined range. The second ones are obtained by fine-tuning the parameters through an automatic procedure based on the F-Race algorithm [10, 11, 12]. This removes some of the ambiguity connected with the measure of the development effort and guarantees that equal attention is devoted to all metaheuristics under analysis. The last ones are exactly the implementations described in the literature: They are obtained considering the implementations and the values of the parameters reported by Bianchi et al. [9]. To the best of our knowledge this paper represents the most recent and successful application of metaheuristics to the problem considered.

The fact of reducing the difference between the *out-of-the-box* and the *custom* version of metaheuristics to the fine-tuning of the parameters is not free of implications and needs to be further justified. The following two observations in favor of the validity and significance of our analysis should be sufficient in order to convince our reader. Although many research papers fail to provide an exhaustive account on how the parameters of the algorithms under analysis are obtained, it is widely recognized that an accurate fine-tuning has a major impact on the performance of algorithms [11, 13, 3, 14]. Selecting the best values for the parameters, given the class of instances that are to be tackled, is definitely a sort of customization. Other elements, as for example an advanced design and implementation of critical data structures, clearly play a major role in the *custom* implementation of a metaheuristic. Nonetheless, the goal of the study is to show that an analysis based on *custom* implementations might produce radically different results from one based on *out-of-the-box* implementations. If we succeed to show this fact when even one single element characterizing *custom* implementations is considered, namely the fine-tuning of parameters, we have nevertheless reached our goal. The use of advanced data structures in the *custom* implementation could only enhance the difference observed.

The rest of the chapter is organized as follows. In Section 2, we present a panoramic view of the literature concerning the vehicle routing problem with stochastic demand, the metaheuristics considered, and the tuning problem. In Section 3, we describe the specific characteristics of these elements as they appear in our analysis. In Section 4, the experimental study is reported. Finally, in Section 5, we make some conclusions.

2 Literature Overview

In this section, we provide the reader with a general overview of the available literature concerning the three main topics of interest of our analysis: i) the vehicle routing problem with stochastic demand, ii) the five metaheuristics we consider in this study, and iii) the problem of fine-tuning metaheuristics.

The problem we consider in our analysis is the vehicle routing problem with stochastic demand (VRPSD). It can be described as follows: Given a fleet of vehicles with finite capacity, a set of customers has to be served at minimum cost. The peculiarity of this variant of the vehicle routing problem is that the demand of each customer is *a priori* unknown and only its probability distribution is available. The actual demand is revealed only when the customer is reached. In this probabilistic setting, the objective of the VRPSD is the minimization of the total expected traveling cost.

Optimal methods, heuristics, and metaheuristics have been proposed in the literature for tackling this problem. In particular, the problem is first addressed by Tillman [15] in 1969. Stewart and Golden [16], Dror and Trudeau [17], Laporte and Louveau [18] and Laporte et al. [19] used techniques from stochastic programming to solve optimally small instances. Bertsimas [20] and Bertsimas and Simchi-Levi [21] proposed different heuristics for solving the VRPSD. They considered the construction of an *a priori* TSP-wise tour. This tour is then split according to precise rules. Yang et al. [22] proposed a strategy for splitting the *a priori* tour allowing the restocking before a stockout, when this is profitable. Secomandi [23, 24, 25] analyzed different possibilities for applying dynamic programming to this problem. Teodorović and Pavković [26] and Gendreau et al. [27] tackled the VRPSD using metaheuristic approaches. In particular, Teodorović and Pavković [26] adopted simulated annealing while Gendreau et al. [27] used tabu search. Finally, an extended analysis on the behavior of different metaheuristics has been proposed by Bianchi et al. [9].

Two classical local search algorithms have been used for the VRPSD: the Or-opt and the 3-opt procedures. The first was proposed by Or [28] in 1976. It consists in the extraction of a string of consecutive nodes from the starting sequence representing a solution, and in its insertion at a different position. Yang et al. [22] presented an approximated way for computing the value of each move. In particular, the cost saving allowed by a move is the difference between the approximated saving obtained removing the string of consecutive customers from its original position, and the approximated cost of inserting it somewhere else in the tour: Instead of evaluating the complete solution before and after the move, the saving and cost are computed only with respect to the immediate neighbors of the customers shifted. This method has been adopted also by Bianchi et al. [9], who proposed also another approximation based on delta values calculated in a TSP-wise fashion, that is, considering the variation of the length of the *a priori* tour. Moreover, they extended this TSP-wise approximation also to the 3-opt local search [29]. In this case, three edges belonging to the starting solution are removed and replaced by three different ones.

Following Bianchi et al. [9], we focus on five of the most popular metaheuristics: tabu search (TS), simulated annealing (SA), genetic algorithm (GA), iterated local search (ILS), and ant colony optimization (ACO).

Tabu search has been introduced by Glover [1] in 1986, on the basis of early ideas formulated a decade before [30]. It consists in the exploration of the solution space via a local search procedure. Tabu search accepts non-improving moves and uses a short term memory. The latter expedient is introduced to avoid sequences of moves that constantly repeat themselves [31].

Simulated annealing takes inspiration from the annealing process in crystals, which assume a low energy configuration when cooled with an appropriate cooling schedule [32, 33, 34, 35, 36]. The principal idea is the exploration of the search space via a local search procedure. Simulated annealing escapes from local minima by allowing moves to worsening solutions. The parameter that controls this mechanism is the *temperature*. It is slowly decreased during the search with the consequence that at the beginning the probability of accepting non-improving solutions is higher, and then it decreases over time. This technique helps in quitting the basin of attraction of high-cost local minima that might be encountered in the early stages of the search.

Genetic algorithms are inspired by the ability shown by populations of living beings to evolve and adapt to changing conditions, under the pressure of natural selection [37]. This metaheuristic is based on the selection of individuals representing candidate solutions. From generation to generation, individuals evolve through *recombination* and *crossover*, and using *mutation* or *modification* operators that lead to self-adaptation [38, 39, 40, 41, 42, 43].

Iterated local search is one of the simplest metaheuristics. It is based on the reiteration of a local search procedure: It explores the neighborhoods of a sequence of solutions obtained via successive perturbations [44].

Ant colony optimization is a metaheuristic based on the foraging behavior of ants [2, 45]. It constructs solutions using a pheromone model, that is, a parameterized probability distribution over the solution space. The solutions found are used to modify the pheromone values biasing the search toward high quality solutions [46].

Tuning is a critical issue when working with metaheuristics. Each metaheuristic can be seen as a modular structure coming with a set of components, each typically provided with a set of free parameters. The tuning problem is the problem of properly instantiating this algorithmic template by choosing the best among the set of possible components and by assigning specific values to all free parameters [12]. Although this problem is generally recognized to be very important when dealing with metaheuristics, only in recent years it has been the object of extensive studies [12, 13, 14, 47, 48, 49, 50]. Some authors adopt a methodology based on factorial design, which is characteristic of a *descriptive* analysis. Therefore, rather than solving directly the tuning problem, they pass through the possibly more complex intermediate problem of understanding the relative importance of each parameter of the algorithm. For example, Xu and Kelly [51] tried to identify the relative contribution of five different components of a tabu-search. Furthermore, the authors considered different values of the parameters of the most effective components and select the best one. Parson and Johnson [52] and Breedam [53] used a similar approach. Xu et al. [54] described a more general technique, which is nonetheless based on factorial analysis. Another approach to tuning that has been adopted for example by Coy et al. [14] and by Adenso-Díaz and Laguna [13] is based on the method that in the statistical literature is known as *response surface methodology*. Bartz-Beielstein and Markon [48] proposed a method to determine relevant parameter settings. It is based on statistical design of experiments, classical regression analysis, tree based regression and design and analysis of computer experiments (a.k.a. DACE) models. Hutter et al. [49] provided methods for optimizing a target algorithms performance on a given class of problem instances by varying a set of ordinal and/or categorical parameters. The authors exploited a family of local-search-based algorithm configuration procedures and presented novel techniques for accelerating them by adaptively limiting the time spent for evaluating individual configurations. Some procedures for tackling the tuning problem have been proposed by Birattari [12]. Among them, the F-Race method is the best performing one and has been used in a number of works on metaheuristics [55, 56, 57, 58, 59].

For the sake of completeness, we mention here another approach to tuning that goes under the name of *on-line tuning*. The key idea behind this second family of techniques is to modify some parameters of the search algorithm while performing the search itself. This approach is particularly appealing when one is supposed to solve one single instance, typically large and complex. One of the most influential descriptions of *on-line* adjustment of the parameters of an algorithm has been given by Battiti and Tecchiolli [60]. The authors introduced a tabu search where the length of the tabu list is optimized on-line.

3 Main Elements of the Analysis

This section provides details on the three main elements of the case study considered in the work. In particular, Section 3.1 describes the algorithmic framework of the vehicle routing problem with stochastic demand and the local search procedures that we consider. Section 3.2 describes the specific implementations of the five metaheuristics under analysis. Section 3.3 describes F-Race, that is, the tuning algorithm that is used for fine-tuning the metaheuristics in our study.

3.1 The Problem

The vehicle routing problem (VRP) consists in finding the set of tours of minimum cost for visiting a given number of customers exactly once, starting and ending each tour at the depot. A fleet of identical vehicles with finite-capacity is to be used for delivering goods to customers, each having a predefined demand. Moving from a customer to another has a cost that is known *a priori*. Typically, problem instances are represented on graphs in which nodes correspond to customers and a cost is associated to each edge.

The VRP can be formulated using the following notation: Let $G = (V, E)$ be a complete undirected graph, with $V = \{0, ..., n\}$ set of nodes. Each node $i \in V \setminus \{0\}$ represents a customer having a nonnegative demand q_i. Node 0 corresponds to the depot. A travel cost c_{ij} is associated to each edge $(i, j) \in E$. Let k be the number of identical vehicles available, each with capacity Q. Let $r(S)$ denote the minimum number of vehicles needed to serve the customers of a subset S of customers. A lower bound of $r(S)$ is $\lceil \sum_{i \in S} q_i / Q \rceil$. For each $s \subset V$ let $\delta(S) = \{(i, j) : i \in S, j \notin$

$S7 \vee i \notin S, j \in S\}$, i.e. the set of edges connecting a node belonging to S with one belonging to $V \setminus S$. Let x_{ij} be the integer variable indicating the number of times edge (i, j) is traversed in the solution. The formulation proposed by Laporte et al. [61] is then:

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{1}$$

s.t.

$$\sum_{(i,j)\in\delta(\{i\})} x_{ij} = 1, \qquad \forall i \in V \setminus \{0\}, \tag{2}$$

$$\sum_{(i,j)\in\delta(\{0\})} x_{ij} = 2k, \tag{3}$$

$$\sum_{(i,j)\in\delta(S)} x_{ij} \geq 2r(S), \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset, \tag{4}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \notin \delta(\{0\}), \tag{5}$$

$$x_{ij} \in \{0,1,2\}, \qquad \forall (i,j) \in \delta(\{0\}). \tag{6}$$

$$\tag{7}$$

Constraints (2) impose that each customer is visited exactly once. Constraint (3) states that k routes are created. Capacity constraints (4) ensure both connectivity of the tours and respect of vehicles capacity. This is done by imposing a sufficient number of edges to enter each subset of nodes S. Constraints (5) and (6) imply that each edge connecting two customers is traversed at most once, while each edge connecting a customer to the depot is traversed at most twice.

In the vehicle routing problem with stochastic demand, the quantities demanded by customers are not known *a priori* but their probability distributions are given.

As in most of the previously published works on VRPSD [9, 20, 21, 22], in this work the problem is addressed by considering only one vehicle. This element was proved to give the best solution in absence of additional constraints [22]. The solution technique consists in constructing an *a priori* TSP-wise tour. This tour is then split according to the specific realizations of the random variables representing the demand of the customers. The objective is finding the *a priori* tour with minimum expected cost.

The computation of the expected cost of the solutions follows Yang et al. [22] and Bianchi et al. [9]. In particular, it is based on a dynamic programming recursion that moves backward from the last node of the sequence. At each node, the decision of restocking or proceeding is based on the expected cost-to-go in the two cases.

In this analysis, we consider the two local search procedures that can be found in the VRPSD literature: Or-opt and 3-opt [9, 22]. Different methods are used for computing the cost of a move in the local search. In this way, five procedures are obtained: Or-opt(TSP-cost), Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), 3-opt(EXACT-cost). For a detailed description of these techniques we refer the reader to Bianchi et al. [9].

In order to reach some significant conclusion with our empirical analysis, a rather large set of instances is needed. The set of instances considered in Bianchi et al. [9] is too small for the aim of our research. To the best of our knowledge, these are the only benchmark instances available for the vehicle routing problem with stochastic demand. For our experiments, we use instances created with the instance generator described in Pellegrini and Birattari [62]. We consider instances with either 50 or 60 nodes.

Following [9], we consider instances in which the demand of each customer is uniformly distributed. The average and the spread of these distributions are reported in Table 1.

Table 1 Parameters used for generating the instances. $U(min, max)$ means that the value is randomly extracted from a uniform distribution in the range between *min* to *max*.

Instance class	average demand	spread
I	$U(20,30)$	$U(5,10)$
II	$U(20,30)$	$U(5,15)$
III	$U(20,35)$	$U(5,10)$
IV	$U(20,35)$	$U(5,15)$

The capacity of the vehicle is 80. In this way the average number of customers that can be served before returning to the depot is about three. Analysis of cases with such a low ratio between capacity of the vehicle and average customer demand are not very frequent in the literature. On the other hand it is a situation that can be easily encountered in reality. A previous study of the performance of algorithms when tackling instances with this peculiarity can be found in Bianchi et al. [9]. Being this paper the main reference for our work, we decided to use instances generated according to the ratio used there.

3.2 Metaheuristics

The implementation of the metaheuristics we consider is based on the code written by Bianchi et al. [9][3]. In the following, we give a short description of the main element characterizing each algorithm. The parameters of the algorithms are briefly explained. As a reference algorithm, following Bianchi et al. [9], we considered a random restart local search (RR). It uses the randomized furthest insertion heuristic plus local search. It restarts every time a local optimum is found, until the stopping criterion is met—in our case, the elapsing of a fixed computational time.

In the **tabu search**, the tabu-list stores partial solutions. An *aspiration criterion* allows forbidden moves if the new solution is the new best one. The *tabu tenure*, that is, the length of the tabu list, is variable [9]: At each step it assumes a random value between $t(m-1)$ and $m-1$, where $0 \leq t \leq 1$ is a parameter of the algorithm. When

[3] Available at `http://iridia.ulb.ac.be/vrpsd.ppsn8`.

3-opt is used, m is equal to the number of customers. When Or-opt is used, m is equal to the number of customers minus the length of the string of consecutive nodes that are shifted in a move. During the exploration of the neighborhood, solutions that include forbidden components are evaluated with probability p_f and the others with probability p_a. The difference between the EXACT-cost, the VRPSD-cost, and the TSP-cost implementations concerns only the local search procedure.

Concerning the **simulated annealing**, the probabilistic acceptance criterion consists in accepting a solution s' either if it has a lower cost than the current solution s or, independently of its cost, with probability

$$p(s'|T_k, s) = exp\left(-\frac{Cost(s')Cost(s)}{T_k}\right). \tag{8}$$

The relevant parameters of the algorithm are related to the initial level of the *temperature* and to its evolution. The starting value T_0 is determined by considering one hundred solutions randomly chosen in the neighborhood of the first one, by computing the variation of the cost in this set, and by multiplying this result for the parameter f. At every iteration k, the *temperature* is decreased according to the formula $T_k = \alpha T_{k-1}$, where the parameter α, usually called *cooling rate*, is such that $0 < \alpha < 1$. If after $n \cdot q \cdot r$ iterations the quality of the best solution is not improved, the process known as *re-heating* [32] is applied: the temperature is increased by adding T_0 to the current temperature. Besides the local search procedure used, the difference between the EXACT-cost, the VRPSD-cost and the TSP-cost implementations consists in the way $Cost(s')$ and $Cost(s)$ in Equation 8 are computed. In the TSP-cost, only the length of the *a priori* tour is considered.

In the implementation of the **genetic algorithm**, edge recombination [63] consists in generating a tour starting from two solutions by using edges present in both of them, whenever possible. Mutation swaps adjacent customers with probability p_m. If mutation is *adaptive*, p_m is equal to the product of the parameter mr (*mutation-rate*) and a similarity factor. The latter depends on the number of times the n-th element of the first parent is equal to the n-th element of the second one. If the mutation is not *adaptive*, p_m is simply equal to mr. The difference between the EXACT-cost, the VRPSD-cost and the TSP-cost implementations concerns only the local search procedure adopted.

The **iterated local search** is characterized by a function that performs a perturbation on solutions. It returns a new solution obtained after a loop of n random moves (with n number of nodes of the graph) of a 2-exchange neighborhood. They consist in subtour inversions between two randomly chosen nodes. The loop is broken if a solution with quality comparable to the current one is found. We say that the quality of a solution is comparable to the quality of the current one if its objective function value is not greater than the objective function value of the current solution plus a certain value ε. The difference between the EXACT-cost, the VRPSD-cost and the TSP-cost implementations concerns only the local search procedure adopted.

In this implementation of **ant colony optimization**, the pheromone trail is initialized to $\tau_0 = 0.5$ on every arc. The first population of solutions is generated and refined via the local search. Then, a *global pheromone update* is performed r times. At each following iteration, p new solutions are constructed by p artificial ants on the basis of the information stored in the pheromone matrix. After each step, the *local pheromone update* is performed on the arc just included in the route. Finally, the local search is applied to the p solutions and the *global pheromone update* is executed. Local and global pheromone updates are performed as follows:

Local pheromone update: the pheromone trail on the arc (i, j) is modified according to the following formula:

$$\tau_{ij} = (1-\psi)\tau_{ij} + \psi\tau_0,$$

with ψ parameter such that $0 < \psi < 1$.

Global pheromone update: the pheromone trail on each arc (i, j) is modified according to the following formula:

$$\tau_{ij} = (1-\rho)\tau + \rho\Delta\tau_{ij}^{bs}$$

where

$$\Delta\tau_{ij}^{bs} = \begin{cases} Q/Cost\_Solution\_bs & \text{if arc } (i,j) \in Solution\_bs \\ 0 & \text{otherwise,} \end{cases}$$

ρ is a parameter such that $0 < \rho < 1$ and $Solution\_bs$ is the best solution found so far.

3.3 The Tuning Process

The parameters of all algorithms considered in the study are tuned through the F-Race procedure [10, 11, 12]. F-Race is a racing algorithm for choosing a candidate configuration, that is, a combination of values of the parameters, out of predefined ranges.

F-race runs the optimization algorithm multiple times testing on several instances a given set of candidate configurations. On the basis of the results achieved, a configuration can be discarded if it appears suboptimal: For each instance (each representing one step of the race) the ranking of the results obtained using the different configurations is computed and a statistical test is performed for deciding whether to discard some candidates or not. The set of configurations considered at a specific step h contains all the candidates that survived after step $h-1$ (Figure 1). F-Race is based on the Friedman two-way analysis of variance by ranks [64]. An important advantage offered by this statistical test is connected with the nonparametric nature of a test based on ranking, which does not require to formulate hypothesis on the distribution of the observations.

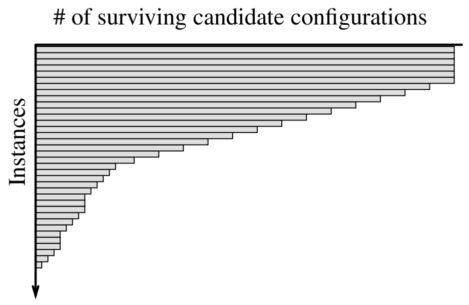


Fig. 1 Graphical representation of the computation performed by the racing approach. As the evaluation proceeds, the racing algorithm focuses more and more on the most promising candidates, discarding a configuration, as soon as sufficient evidence is gathered that it is suboptimal [11, 12].

4 Experimental Analysis

The main goal of the computational experiments proposed in this section is to show that a remarkable difference exists between the results obtained by *out-of-the-box* and *custom* versions of the metaheuristics. Moreover, we wish to study the performance of a *literature* version of the same approaches, that is, a version that reproduces as precisely as possible implementation described in the literature. We aim at showing that by using published versions that have been optimized for solving possibly different problem instances, the results that one might obtain are not necessarily state-of-the-art. In particular, they might significantly differ from those that can be obtained through *custom* implementations specifically tailored to the class of problem instances at hand.

As we mentioned in the introduction, the various versions differ one from the other in the values of the parameters. In the *literature* versions, the values of the parameters are those proposed by Bianchi et al. [9]. In the *custom* versions, the parameters are accurately fine-tuned with the F-Race automatic procedure. As mentioned in Section 3.3, F-Race selects the best values of the parameters out of a given set of candidate ones. Finally, in the *out-of-the-box* versions, the values of the parameters are randomly drawn from the same set of candidate values that is considered by F-Race for *custom* versions. Equal probability has been associated to each configuration and, for each instance considered in the analysis, a random selection has been performed. The choice of randomly drawing the values of the parameters is motivated by the observation that an *out-of-the-box* implementation is often based on an experience–based selection: Given a set of reasonable values, one makes decisions that will end up being more or less “lucky”. In order to prescind from our fortune while testing the thesis at the basis of this work, a sort of average performance is sought. Such a selection criterion is anyway much different from the random picking of admissible values for the parameters: Following our experience in the field, if we did not have the possibility of tuning parameters, we could have reasonably chosen any of the available combinations for running experiments.

For each of the metaheuristics, besides the methods used for setting the parameters, the implementations considered in the *out-of-the-box*, *custom*, and *literature* versions are identical.

The values of the parameters represent only one of the elements that may be customized when implementing a metaheuristic. By considering only this element in our study, we somehow underestimate the difference between *out-of-the-box* and *custom* implementations. Nonetheless, if we succeed to show that the results of an analysis performed on *out-of-the-box* implementations cannot be generalized to *custom* implementations when the difference between the two simply consists in the values of the parameters, we have reached our goal: Any other element that can be fine-tuned and customized would simply further reduce the possibility of generalizing results observed in one context to the other.

All experiments are run on a cluster of AMD Opteron$^{\text{TM}}$ 244, and 1000 instances are considered. We run each algorithm once on each instance [65]. The computation time is used as a stopping criterion for all the algorithms and it is set to 30 seconds.

In order to obtain the *custom* versions of the metaheuristics through F-Race, a number of different configurations ranging from 1200 to about 1600 were considered for each of them. Table 2 reports, for each metaheuristic, the parameters considered for optimization, the range of values allowed, and the values selected. A set of 500 instances of the vehicle routing problem with stochastic demand was available for the tuning. These instances have the same characteristics of the ones used for the experimental analysis, but the two sets of instances are disjoint [5]. While tuning a metaheuristic, the F-Race procedure was allowed to run the metaheuristic under consideration for a maximum number of times equal to 15 times the number of configurations considered for that metaheuristic. Also for the random restart local search, a *custom* version has been considered. It has been obtained by selecting, through the F-Race procedure, the best performing local search. In other words, the parameter that has been optimized in this case is the underlying local search.

A first analysis of the performance of the algorithms in the two contexts, *custom* and *out-of-the-box*, consists in comparing the results achieved in terms of cost of the best solution returned. Figure 2 reports the distributions of the difference between the costs of the solutions obtained in the two contexts by each metaheuristic. To be precise, we report the distribution of the cost of the solutions found by each *custom* version minus the one of its *out-of-the-box* counterpart. In Figure 2(a), the whole distributions are shown. In Figure 2(b), the detail of the area around zero is reported. We can observe that, even if the tails of the distributions are sometimes very long[4], almost 75% of the observations fall below the zero line for all metaheuristics. This means that, in the strong majority of the cases, the difference is in favor of the *custom* version. Again, we can observe that some metaheuristics are more sensitive

[4] The long tails of the distributions do not appear to be dependent on any specific aspect of the instances solved. In the same way, it is not possible to find a correlation between the parameter configurations use in the *out-of-the-box* versions and the presence of outliers.

Table 2 Range of values considered for the parameters of the metaheuristics. The values reported in bold are the ones selected by F-Race for the *custom* versions.

Tabu search – total number of candidates = 1460	
parameter	range
p_f	0.1, **0.2**, 0.25, 0.3, 0.35, 0.4
p_a	0.5, **0.6**, 0.7, 0.75, 0.8, 0.85, 0.9
t	**0.3**, 0.4, 0.5, 0.7, 0.8, 0.9, 1
local_search	Or-opt(TSP-cost), Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), **3-opt(EXACT-cost)**

Simulated annealing – total number of candidates = 1200	
parameter	range
α	**0.3**, 0.5, 0.7, 0.9, 0.98
q	**1**, 5, 10
r	10, 20, 30, **40**
f	0.01, **0.03**, 0.05, 0.07
local_search	**Or-opt(TSP-cost)**, Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), 3-opt(EXACT-cost)

Genetic algorithm – total number of candidates = 1360	
parameter	range
pop. size	10, 12, 14, 16, 18, **20**, 22, 24
mr	0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65,**0.7**, 0.75, 0.8, 0.85, 0.9
adaptive	**Yes**, No
local_search	**Or-opt(TSP-cost)**, Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), 3-opt(EXACT-cost)

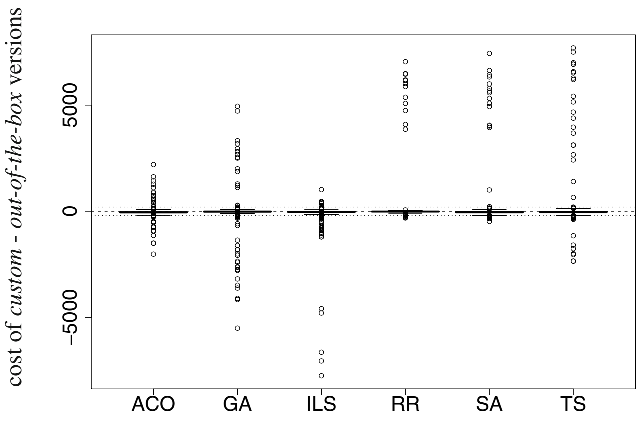
Iterated local search – total number of candidates = 1520	
parameter	range
ε	$n/x, x \in$ {0.005,0.01,0.05,0.1,0.5, 1.0,**1.5**,2.0, all multiples of 0.5 up to 150.0}
local_search	**Or-opt(TSP-cost)**, Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), 3-opt(EXACT-cost)

Ant colony optimization – total number of candidates = 1620	
parameter	range
p	**5**,10, 20
ρ	0.1, 0.5, **0.7**
r	100, **150**, 200
Q	$10^5, 10^6, 10^7, \mathbf{10^8}, 10^9$
local_search	Or-opt(TSP-cost), Or-opt(VRPSD-cost), Or-opt(EXACT-cost), **3-opt(TSP-cost)**, 3-opt(EXACT-cost)

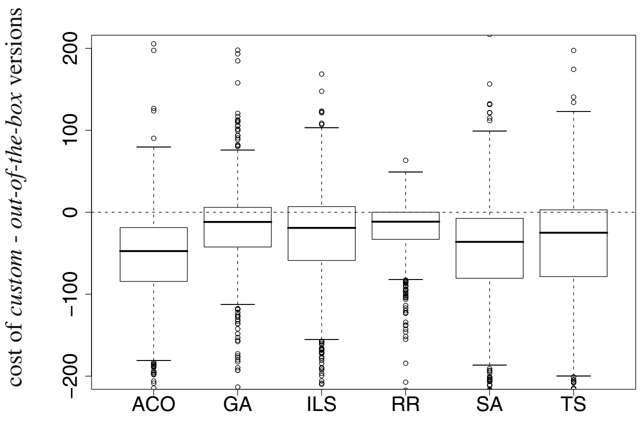
Random restart – total number of candidates = 5	
parameter	range
local_search	Or-opt(TSP-cost), Or-opt(VRPSD-cost), Or-opt(EXACT-cost), 3-opt(TSP-cost), **3-opt(EXACT-cost)**

to the value of their parameters and therefore benefit more than others from an accurate fine-tuning. Observing these results, it is immediately clear that, as expected, the performance achieved by algorithms depend strongly on the values chosen for the parameters, and then on the contexts considered.

Some further observations can be made considering the distribution of the ranking achieved by each algorithm. These results are reported in Figures 3(a), 3(b), and 3(c), for the *custom*, *out-of-the-box*, and *literature* versions respectively. On the left of each graph, the names of the algorithms are given. The order in which they appear reflects the average ranking: The lower the average ranking, the better the general behavior, and the higher the metaheuristic appears in the list. On the right, the boxplots represent the distributions of the ranks over the 1000 instances. Between the names and the boxplots, vertical lines indicate if the difference in the behavior of the



(a)



(b)

Fig. 2 Difference between the costs of the solutions obtained by the *custom* and the *out-of-the-box* versions of the metaheuristics under analysis. In Figure 2(a), the entire distribution is shown for each metaheuristic. Since the distributions are characterized by long tails, in Figure 2(b) the detail of the more interesting central area is given. For all metaheuristics, the median of the distribution is below the zero: Being the VRPSD a minimization problem, the results obtained by the *custom* versions are in general better than those obtained by their *out-of-the-box* counterpart.

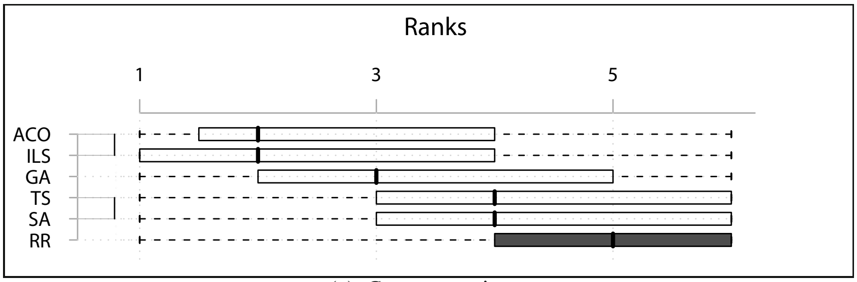
metaheuristics is significant according to the Friedman test: If two metaheuristics are not comprised by the same vertical line, their behavior is significantly different according to the statistical test considered, with a confidence of 95%. The difference in the denomination of the algorithms between the first two figures and the third one depends on the fact that in Bianchi et al. [9], the local search procedure is not considered as a parameter of the algorithms. For this reason, the metaheuristics are presented in Figure 3(c) with the name of the metaheuristic paired with the one of the variant of the local search used. In Bianchi et al. [9], the 3-opt local search has been used only in association with iterated local search and genetic algorithms.

As it can be observed, the ranking of algorithms varies in the three contexts. First of all, let us focus on the graphics representing the *out-of-the-box* and the *custom* versions. The two main differences concern RR and ACO. The former performs the worst in the *custom* context, while this is not the case in the *out-of-the-box* context. The case of a metaheuristic performing worse than the random restart local search is to be considered as a major failure for the metaheuristic itself. We consider this point as a remarkable difference between the two contexts: In the *out-of-the-box* context, three out of five metaheuristics perform significantly worse than the random restart local search; in the *custom* context, all metaheuristics achieve better results than the random restart local search.

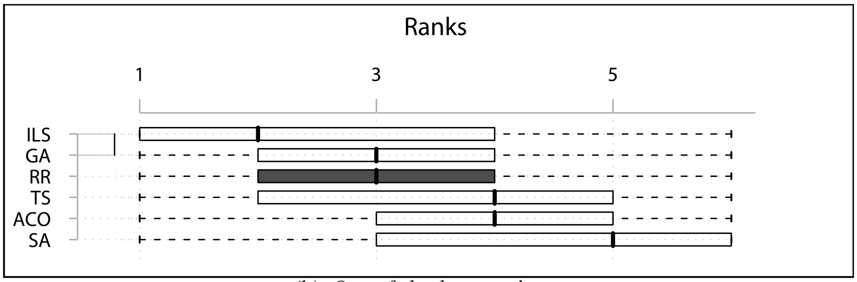
As far as ACO is concerned, we can observe that the relative performance is visibly different in the two contexts. The *out-of-the-box* version behaves significantly worse than RR, and is among the worst in the set. On the contrary, the *custom* version achieves the best average ranking. This difference shows that this metaheuristic is more sensitive than the others to variations of the parameters, possibly due to the large number of parameters of the algorithm. This might be seen as a drawback of ACO. Anyway, we think that this fact should be read in a different way: If one is interested in an *out-of-the-box* metaheuristics, a high sensitivity to the parameters is definitely an issue; on the other hand, if one wishes to implement a *custom* metaheuristic, the sensitivity is an opportunity that can be exploited in order to finely adapt the algorithm to the class of instances to be tackled.

Let us consider now Figure 3(c), where the performance of the *literature* version is reported. As it can be noted by comparing this graph with Figures 3(a) and 3(b), the general trend is very similar to the one obtained in the *out-of-the-box* context.

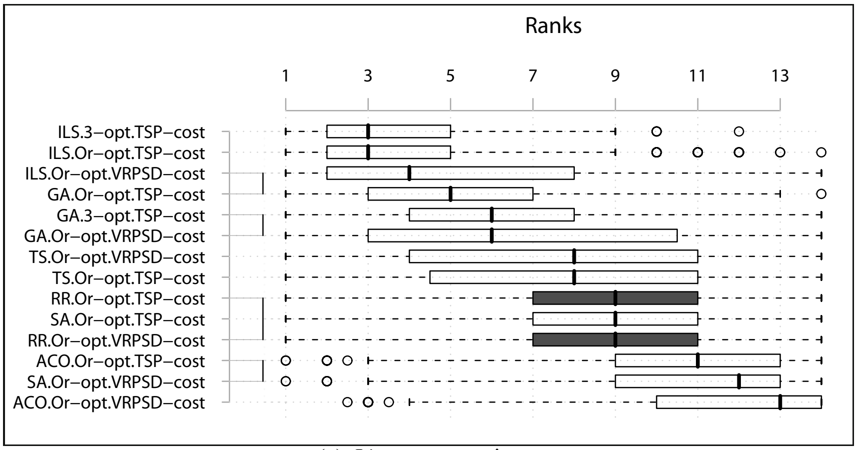
In order to provide a more precise picture of the sensitivity of each metaheuristics to its parameters, Figure 4 reports, for each metaheuristic, the comparison of the results obtained in the three contexts. What clearly emerges is that, as expected, all metaheuristics achieve the best results in their *custom* version. The difference is always statistically significant according to the Friedman test. Moreover, it can be observed that, less expectedly, *literature* versions, that is, those in which the parameters are set according to Bianchi et al. [9], obtain results that are comparable with those of the *out-of-the-box* versions. In particular, while for iterated local search and tabu search the values reported in the literature appear to be better than those drawn at random, for genetic algorithms and simulated annealing this is not always the case. Even more striking, in the case of ant colony optimization, the parameters used in Bianchi et al. [9] yield results that are significantly worse than those drawn



(a) *Custom* versions.



(b) *Out-of-the-box* versions.



(c) *Literature* versions.

Fig. 3 Results over 1000 instances of the metaheuristics in the three variants considered. Two main observations can be made: In the *out-of-the-box* and *literature* versions we can see that some metaheuristics are outperformed by the random restart local search. This represents a major failure. In the *custom* versions all metaheuristic achieve better results. Moreover, the relative behavior of metaheuristics changes. For example, ACO ranks first in Figure 3(a), the fifth in Figure 3(b), and last in Figure 3(c).

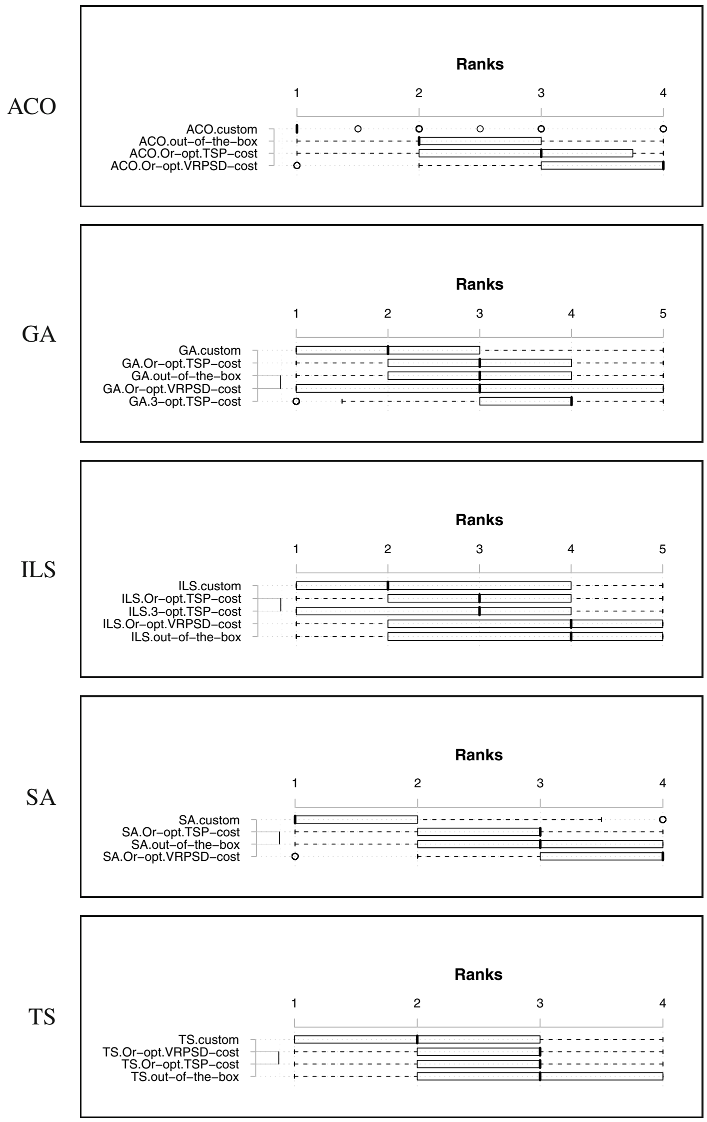


Fig. 4 Comparison of the results obtained in the three sets of experiments. As it can be observed, the *custom* versions are always significantly better than all the others. For iterated local search and tabu search, *literature* versions are better than their *out-of-the-box* counterparts, that is, the values chosen by Bianchi et al. [9] behave better than the random ones. On the contrary, the *out-of-the-box* ant colony optimization works better than the *literature* version. Finally, in genetic algorithms and simulated annealing the results are mixed: the *out-of-the-box* versions is better than one of the two *literature* versions and worse that the other one, or of the other two in the case of genetic algorithms for which three versions where proposed in Bianchi et al. [9].

at random. These results clearly support our claim according to which there is a strong difference between the performance of metaheuristics used *out-of-the-box* or in a *custom* way. Moreover, they show that the versions that can be found in the literature are not necessarily state-of-the-art, when applied to problem instances that differ from those considered in the original study.

5 Conclusions

In this chapter, five of the most successful metaheuristics, namely tabu search, simulated annealing, genetic algorithm, iterated local search, and ant colony optimization, have been compared on the vehicle routing problem with stochastic demand. These five metaheuristics applied to the vehicle routing problem with stochastic demand have been the focus of a research published by Bianchi et al. [9].

Our goal is to highlight that results obtained with *out-of-the-box* versions of metaheuristics cannot be directly generalized to *custom* versions. In our analysis, what differentiates a *custom* version of a metaheuristic from the corresponding *out-of-the-box* one, is that the parameters of the former are fine-tuned through the F-Race algorithm, while those of the latter are drawn at random.

As it could be expected, the empirical results show that the *custom* version of each metaheuristic achieves better results than the corresponding *out-of-the-box* one. The difference is always statistically significant according to the Friedman test. Moreover, the relative performance of algorithms differs greatly in the two contexts. This can be ascribed to the fact that different metaheuristics might be more or less sensitive to variations of their parameters.

A second element on which the analysis focuses is whether the results that are reported in the literature can be *a priori* associated with to one of the two contexts. From our experiments it appears clear that this is not the case.

On the basis of this case study, we can conclude that there may be a strong difference in the results achievable by using the *out-of-the-box* or the *custom* version of metaheuristics. This difference may concern both the quality of the solutions returned by an approach, and the relative performance of algorithms. As a consequence, one should clearly describe the implementation criteria followed in the design of an algorithm, in order to allow the readers to focus on the more suitable implementations, given their specific goals.

The lack of this piece of information cannot be filled by considering all the implementation studied in the literature as custom: as we show, they may refer alternatively to either context. This element confirms the relevance of our research. In this precise sense, the analysis presented in this work is strongly related to a subject that has an actual impact on the current research in the field of metaheuristics.

Acknowledgments. This work was partially supported by the ARC projects ANTS and META-X, funded by the Scientific Research Directorate of the French Community of Belgium. Mauro Birattari acknowledges support from the fund for scientific research F.R.S.-FNRS of Belgium's French Community, of which he is a research associate.

References

[1] Glover, F.: Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 13, 533–549 (1986)

[2] Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)

[3] Barr, R.S., Kelly, J.P., Resende, M.G.C., Stewart, W.R.: Designing and reporting computational experiments with heuristic methods. Journal of Heuristics 1(1), 9–32 (1995)

[4] Birattari, M., Dorigo, M.: How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? Optimization Letters 1(3), 309–311 (2006)

[5] Birattari, M., Zlochin, M., Dorigo, M.: Towards a theory of practice in metaheuristics design: A machine learning perspective. Theoretical Informatics and Applications 40(2), 353–369 (2006)

[6] Eiben, A.E., Jelasity, M.: A critical note on experimental research methodology in EC. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 582–587. IEEE Press, Los Alamitos (2002)

[7] Hooker, J.N.: Testing heuristics: We have it all wrong. Journal of Heuristics 1, 33–42 (1995)

[8] Pellegrini, P., Birattari, M.: Implementation effort and performance. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) SLS 2007. LNCS, vol. 4638, pp. 31–45. Springer, Heidelberg (2007)

[9] Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O., Schiavinotto, T.: Hybrid metaheuristics for the vehicle routing problem with stochastic demands. Journal of Mathematical Modelling and Algorithms 5(1), 91–110 (2006)

[10] Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B., et al. (eds.) GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 11–18. Morgan Kaufmann, San Francisco (2002)

[11] Birattari, M.: The problem of tuning metaheuristics as seen from a machine learning perspective. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium (2005)

[12] Birattari, M.: Tuning Metaheuristics: A Machine Learning Perspective. Springer, Berlin (2009)

[13] Adenso-Díaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research 54(1), 99–114 (2006)

[14] Coy, S.P., Golden, B.L., Runger, G.C., Wasil, E.A.: Using experimental design to find effective parameter settings for heuristics. Journal of Heuristics 7(1), 77–97 (2001)

[15] Tillman, F.: The multiple terminal delivery problem with probabilistic demands. Transportation Science 3, 192–204 (1969)

[16] Stewart, W., Golden, B.: Stochastic vehicle routing: a comprehensive approach. European Journal of Operational Research 14, 371–385 (1983)

[17] Dror, M., Trudeau, P.: Stochastic vehicle routing with modified saving algorithm. European Journal of Operational Research 23, 228–235 (1986)

[18] Laporte, G., Louveau, F.: Formulations and bounds for the stochastic capacitated vehicle routing problem with uncertain supplies. Technical Report G-87-23, Ecole des Hautes Etudes Commerciale, University of Montreal, Montreal, Canada (1987)

[19] Laporte, G., Louveau, F., Mercure, H.: Models and exact solutions for a class of stochastic location-routing problems. Technical Report G-87-14, Ecole des Hautes Etudes Commerciale, University of Montreal, Montreal, Canada (1987)

[20] Bertsimas, D.J.: A vehicle routing problem with stochastic demand. Operations Research 40(3), 574–585 (1992)
[21] Bertsimas, D.J., Simchi-Levi, D.: A new generation of vehicle routing research: robust algorithms, addressing uncertainty. Operations Research 44(3), 286–304 (1996)
[22] Yang, W.H., Mathur, K., Ballou, R.H.: Stochastic vehicle routing problem with restocking. Transportation Science 34(1), 99–112 (2000)
[23] Secomandi, N.: Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Computers & Operations Research 27, 1201–1225 (2000)
[24] Secomandi, N.: A rollout policy for the vehicle routing problem with stochastic demands. Operations Research 49, 796–802 (2001)
[25] Secomandi, N.: Analysis of a rollout approach to sequencing problems with stochastic routing applications. Journal of Heuristics 9, 321–352 (2003)
[26] Teodorović, D., Pavković, G.: A simulated annealing technique approach to the VRP in the case of stochastic demand. Transportation Planning and Technology 16, 261–273 (1992)
[27] Gendreau, M., Laporte, G., Séguin, R.: A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. Working paper, CRT, University of Montreal, Montreal, Canada (1994)
[28] Or, I.: Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking. PhD thesis, Northwestern University, Evanston, IL, USA (1976)
[29] Lin, S.: Computer solutions of the traveling salesman problem. Bell System Tech. Journal 44, 2245–2269 (1965)
[30] Glover, F.: Heuristics for integer programming using surrogate constraints. Decision Sciences 8, 156–166 (1977)
[31] Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)
[32] Aarts, E.H.L., Korst, J.H.M., van Laarhoven, P.J.M.: Simulated annealing. In: Aarts, E., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 91–120. John Wiley & Sons, Inc., New York (1997)
[33] Cerny, V.: A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications 45, 41–51 (1985)
[34] Fleischer, M.: Simulated annealing: past, present and future. In: Lilegdon, W.R., Alexopoulos, C.L., Kang, K., Goldsam, G. (eds.) Proceedings of the 1995 Winter Simulation Conference, pp. 155–161 (1995)
[35] Ingber, L.: Adaptive simulated annealing (ASA): lessons learned. Control and Cybernetics 26(1), 33–54 (1996)
[36] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
[37] Darwin, C.R.: On the Origin of Species by Means of Natural Selection. Or the preservation of favoured races in the struggle for life. John Murray, London (1859)
[38] Back, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. IOP Publishing Ltd., Bristol (1997)
[39] Fogel, L.J.: Toward inductive inference automata. In: Proceedings of the International Federation for Information Processing Congress, Munich, Germany, pp. 395–399 (1962)
[40] Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligent through Simulated Evolution. John Wiley & Sons, New York (1966)

[41] Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading (1989)
[42] Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor (1975)
[43] Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
[44] Laurenço, H.R., Martin, O., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 321–353. Kluwer Academic Publishers, Norwell (2002)
[45] Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization: Artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine 1(4), 28–39 (2006)
[46] Zlochin, M., Birattari, M., Meuleau, N., Dorigo, M.: Model-based search for combinatorial optimization: A critical survey. Annals of Operations Research 131(1-4), 373–395 (2004)
[47] Bartz-Beielstein, T., Preuss, M., Reinholz, A.: Evolutionary algorithms for optimization practitioners. Technical Report CI-151/03, Interner Bericht des Sonderforschungsbereichs 531 Computational Intelligence, Universität Dortmund, Dortmund, Germany (2003)
[48] Bartz-Beielstein, T., Markon, S.: Tuning search algorithms for real-world applications: A regression tree based approach. In: Greenwood, G.W. (ed.) Proc. 2004 Congress on Evolutionary Computation (CEC 2004), pp. 1111–1118. IEEE Press, Piscataway (2004)
[49] Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: Paramils: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36, 267–306 (2009)
[50] Favaretto, D., Moretti, E., Pellegrini, P.: On the explorative behavior of MAX–MIN ant system. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) SLS 2009. LNCS, vol. 5752, pp. 115–119. Springer, Heidelberg (2009)
[51] Xu, J., Kelly, J.: A network flow-based tabu search heuristic for the vehicle routing problem. Transportation Science 30, 379–393 (1996)
[52] Parson, R., Johnson, M.: A case study in experimental design applied to genetic algorithms with applications to DNA sequence assembly. American Journal of Mathematical and Management Sciences 17, 369–396 (1997)
[53] Van Breedam, A.: An analysis od the effect of local improvement operators in genetic algorithms and simulated annealing for the vehicle routing problem. Technical Report TR 96/14, Faculty of Applied Economics, University of Antwerp, Antwerp, Belgium (1996)
[54] Xu, J., Chiu, S.Y., Glover, F.: Fine-tuning a tabu search algorithm with statistical tests. International Transactions on Operational Research 5(3), 233–244 (1998)
[55] Chiarandini, M.: Stochastic local search for overconstrained problems. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany (2005)
[56] Chiarandini, M., Stützle, T.: Experimental evaluation of course timetabling algorithms. Technical Report AIDA-02-05, FG Intellektik, FB Informatik, Technische Universität Darmstadt, Darmstadt, Germany (2002)
[57] den Besten, M.L.: Simple metaheuristics for scheduling. An empirical investigation into the application of iterated local search to deterministic scheduling problemns with tardiness penalities. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany (2004)

[58] Schiavinotto, T., Stützle, T.: The linear ordering problem: instances, search space analysis and algorithms. Journal of Mathematical Modelling and Algorithms 3, 367–402 (2004)

[59] Yuan, B., Gallagher, M.: Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tino, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 172–181. Springer, Heidelberg (2004)

[60] Battiti, R., Tecchiolli, G.: The reactive tabu search. ORSA Journal on Computing 6, 126–585 (1994)

[61] Laporte, G., Nobert, Y., Desrochers, M.: Optimal routing under capacity and distance restrictions. Operations Research 33, 1050–1073 (1985)

[62] Pellegrini, P., Birattari, M.: Instances generator for the vehicle routing problem with stochastic demand. Technical Report TR/IRIDIA/2005-10, Iridia, Université Libre de Bruxelles, Brussels, Belgium (2005)

[63] Whitley, D., Starkweather, T., Shaner, D.: The traveling salesman problem and sequence scheduling: quality solutions using genetic edge recombination. In: Davis, L. (ed.) Handbook of Genetic Algorithms, pp. 350–372. Van Nostrand Reinhold, New York (1991)

[64] Friedman, J.H.: Multivariate adaptive regression splines. The Annals of Statistics 19, 1–141 (1991)

[65] Birattari, M.: On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs? Technical Report TR/IRIDIA/2004-01, Iridia, Université Libre de Bruxelles, Brussels, Belgium (2004)

Analogue Circuit Optimization through a Hybrid Approach

Mourad Fakhfakh, Amin Sallem, Mariam Boughariou, Sameh Bennour,
Eya Bradai, Emna Gaddour, and Mourad Loulou

Abstract. Optimal analogue circuit sizing is investigated in this chapter. It is shown that hybridization of a global optimization approach with a local one leads to better results in optimization of such circuits, than using classical approaches. The case of merging Genetic Algorithms with the Simulated Annealing technique is considered. The hybrid algorithm is detailed and is evaluated using test functions. It is shown through three application examples, i.e. optimization of performances of a current conveyor, an operational transconductance amplifier and a low noise amplifier, that such hybrid algorithms yield optimal solutions in a much shorter time, when compared to conventional meta-heuristics.

1 Introduction

Advances in very low scale integration (VLSI) technology nowadays allow the realization of complex integrated electronic circuits and systems [1]. Design automation of digital circuits has already been widely explored and many CAD tools are available [1-3]. However, the automated design of analogue circuits is far to be mature. It still lags behind that of digital circuits. Optimal synthesis, design and sizing of analogue components/circuits are very often a bottleneck in the design flow [1-5].

Many analogue circuit optimization techniques are proposed in the literature with the trend to the use of statistic-based approaches, see for instance [3,6-9]. These approaches generally start with finding a "good" DC operating point, and then a simulation-based tuning procedure takes place. However, these approaches are time consuming and do not guarantee the convergence to the 'global' optimum solution. Actually, analogue circuit optimization problems simultaneously deal with different types of variables, objective functions and constraints. Therefore, classical optimization techniques as well as statistic-based approaches are

Mourad Fakhfakh · Amin Sallem · Mariam Boughariou · Sameh Bennour
Eya Bradai · Emna Gaddour · Mourad Loulou
University of Sfax, Tunisia
email: mourad.fakhfakh@ieee.org

generally not adequate. Meta-heuristics are used to solve such hard constrained problems [10,11]. They offer the advantages to be 'easily' modified and adapted to suit specific problem requirements [10-13]. Even though they don't guarantee to find exactly the optimal solution, a fact due to their stochastic nature, they give, within an acceptable computing time, a good approximation of it. Meta-heuristics can be classified into two categories [13]:

• Population based approaches. This category mainly comprises:

- Evolutionary Algorithms (EA) [14,15]: Genetic Algorithms (GA) [16-18], the Evolutionary Programming (EP) [14,15,19], etc.,
  and

- Swarm Intelligence Techniques (SI) [20]: Particle Swarm Optimization (PSO) [21,22], Ant Colony Optimization (ACO) [23], Bacterial Foraging Optimization (FBO) [24], etc.

and

• Single solution based approaches, such as Simulated Annealing (SA) [25-28], Tabu Search (TS) [29,30]…

Some among the aforementioned optimization techniques were used in optimizing analogue circuits, see for instance [16-18,27,28,31-36]. These approaches are part of the so called global optimization (i.e. a global search approaches (GS)). On the other hand, SA is sometimes referred in the literature as a local search (LS) approach, since it is basically based on a LS one, namely the Metropolis algorithm [37].

Both EA and SI techniques are robust global optimization techniques for solving problems having many local optima; nonetheless, they require long computation time. Further, they suffer from poor convergence performances [38]. On the other hand, local search algorithms can converge in a few iterations but are deficient in a global outlook; they rely on a suitable starting point: It is the major hurdle that can meet LS approaches. Combining a global search procedure and a local one should offer the advantages of both, while offsetting their drawbacks.

Some hybrid algorithms [38,39] are proposed in the literature, such as genetic annealing [40]. The latter was used in many optimisation applications, but in the circuit design field it was applied only to solve problems related to floorplanning and circuit placement [41,44]. According to the knowledge of the authors, such hybrid techniques have not been used to the optimal sizing of analog circuits.

This chapter presents a hybrid optimization technique that merges a genetic algorithm with a local search technique, namely the SA. Fig. 1 depicts the proposed idea. The efficiency of this hybrid approach is first shown through a constrained mathematical test-case. The hybrid algorithm, called hybrid GA/SA, is then used for the optimal sizing of different aspect analogue circuits.

The rest of the chapter is structured as follows. In section 2, the analogue circuit optimization problem is detailed. In sections 3 and 4, GA and SA algorithms are presented. Section 5 highlights the hybrid GA/SA algorithm. In sections 6, 7 and 8 three application examples of optimizing the sizing of analogue circuits are presented: Section 6 deals with the optimal sizing of a positive second generation current conveyor. Section 7 focuses on the optimal design of a fully differential folded cascode transconductance amplifier. Section 8 highlights optimization of an ultra wide band low noise amplifier through its symbolic scattering parameters. Finally, concluding remarks are given in section 9.

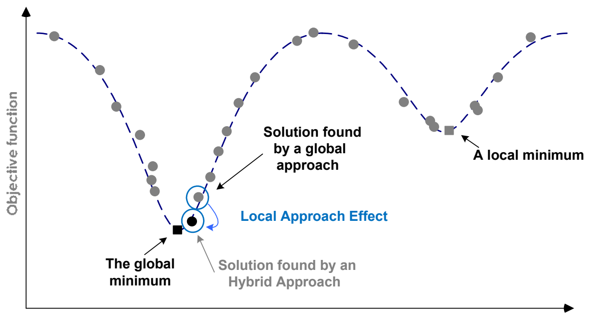


Fig. 1 Effect of combining a LS approach to a GS one.

2 The Analogue Optimization Problem Formulation

A general optimization problem can be defined in the following format:

$$\left| \begin{array}{ll} \textit{Minimize } f(\vec{x}); & f(\vec{x}) \in R \\ \textit{such that}: & \\ \vec{g}(\vec{x}) \leq 0; & \vec{g}(\vec{x}) \in R^{m} \\ \textit{and } \vec{h}(\vec{x}) = 0; & \vec{h}(\vec{x}) \in R^{n} \\ \textit{where } x_{Li} \leq x_i \leq x_{Ui}, & i \in [1, p] \end{array} \right. \tag{1}$$

m, n and p are numbers of inequality constraints to satisfy, equality constraints to assure and parameters to manage, respectively. $\vec{x}_L$ and $\vec{x}_U$ are lower and upper boundary vectors of the parameters.

Classical meta-heuristics cannot handle inequality constraints, so the constrained optimization problem is transformed into an unconstrained one by minimizing the following function:

$$\tilde{f}(x) = f(x) + \gamma(d(x, Fr)) \tag{2}$$

where $d(x, Fr)$ is a distance metric of the infeasible point to the feasible region Fr. It may simply be zero if no constraint violation occurs and is a positive scalar, otherwise [15]. The definition of this distance metric includes the penalty coefficients that are used to stress the importance of a particular constraint violation over others. γ is a weight coefficient to set a global importance of the constraints with respect to the objective function [15].

Optimal design/sizing of analogue circuits consists of finding a variable set $\vec{x} = \{x_1, x_2, \ldots, x_n\}$ that optimizes a performance function(s), such as gain, offset, signal to noise ratio, maximum operating frequency etc., while meeting imposed specifications and/or inherent constraints, for example, saturation conditions of transistors, technology limits, impedance matching, etc. Vector $\vec{x}$ may encompass biases, lengths and widths of MOS transistors, component values etc.

Fig. 2 gives a pictorial view of design optimization approach.

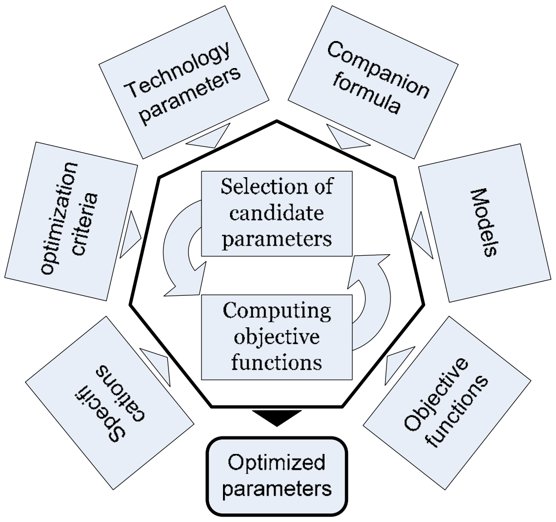


Fig. 2 Pictorial view of a design optimization approach.

Most of analogue optimization problems require different types of variables, objective and constraint functions simultaneously in their formulation. Therefore, classical optimization procedures are generally not adequate, as it was highlighted in section 1. Meta-heuristics allow solving multi-criterion large size problems. They can be adapted to specific problem requirements. Even though they don't guarantee finding the 'exact' optimal solution, they give a 'good' approximation of it within a tolerable CPU computing time.

In this chapter we focus on two meta-heuristics: a Genetic Algorithm and the Simulated Annealing technique, and the hybridization of both.

3 The Genetic Algorithms

Genetic algorithms (GAs) [45,46] are inspired by *Darwin*'s theory of evolution; they mimic natural evolution processes to evolve a solution to a problem:

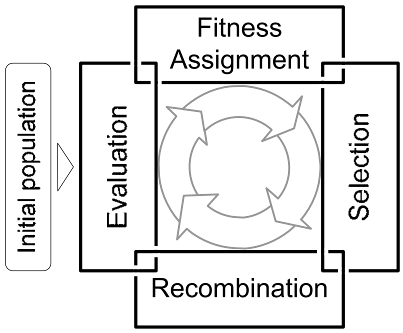


Fig. 3 The basic cycle of evolutionary algorithms.

combination of selection, recombination and mutation. They form a subclass of evolutionary algorithms. Fig. 3 illustrates its basic principle.

where *Evaluation* consists of computing the objective values of the solution candidates. *Fitness Assignment* uses the objective values to determine fitness values. *Selection* chooses the fittest individuals for reproduction, and *Reproduction* creates new individuals from the mating pool by crossover and mutation [15,46].

The pseudo code of a basic GA is given in Fig. 4.

```
Setup the GA.
**Main** ( )
    InitPopulation ( ) // *random initialization of the population*
    max_fitness := 0
    **for each** member chromosome
            fitness := Fitness_Evaluation (chromosome)
            **if** fitness > max_fitness
                    max_fitness := fitness
                    fittest_solution = chromosome
            **end if**
    **end for**
    **while** generation < max_generations
            offspring := Selection&Recombination (parents) // *Fig. 5*
            fitness := Fitness_Evaluation (offspring)
            **if** fitness > max_fitness
                    max_fitness := fitness
                    fittest_solution = offspring
            **end if**
    save fittest_solution
**end**
```

Fig. 4 The pseudo-code of a GA.

```
Selection&Recombination ( )
    while num of programs < max_prog/2 Do
        parent_1 := Roulette-Wheel_Selection ( )
        parent_2 := Roulette-Wheel_Selection ( )
        randomly choose x-over point
        child_1 := parent_1 [head] ^ parent_2 [tail]
        child_2 := parent_2 [head] ^ parent_1 [tail]
        for each child
            mutation_pt := Random_Mutation_Point (child)
            new_instr := Random_New_Instruction ( )
            Instruction (mutation_pt, new_instr)
        end for
end
```

Fig. 5 The pseudo-code Selection & Recombination routine.

4 The Simulated Annealing Technique

The simulated annealing technique [26] is inspired by the natural annealing process used in metallurgy. The annealing technique is used to create a solid state by slowly cooling a melted metal. The gradual decrease of the metal temperature produces the crystalline lattice, which minimizes its energy probability distribution.

In fact, when the temperature of a metal is high, the particles within the metal are able to move around, changing the structure of the metal, freely. As the temperature is lowered, the particles are limited in the movements they can make as many movements have a high energy cost and are increasingly limited to only those configurations with lower energy than the previous state.

From an algorithmic point of view, this can be modeled as a random exploration on a search graph. Each vertex of the substance represents a solution (or a state) with a certain fitness value, and the adjacent vertices represent other similar solutions (in the sense that their fitness value is not expected to be significantly different). At each step, the algorithm selects randomly an adjacent vertex. It transits to the new state if the current solution improves the actual state. Worse solutions are not radically rejected, but they may be considered and the algorithm may use the new 'worse' state with a probability determined by the global 'temperature' parameter. This behaviour allows escaping from local minima. As the temperature drops, this becomes less likely and the search drops into a nearby local minimum [26-28].

The pseudo code of the SA technique is given in Fig. 6.

```
Setup the SA.
Main ( )
Choose a random solution $x$ which takes the minimum solution $x_{min}$
($x=x_{min}$)
Evaluate the fitness function $f(x)$
Initialize the temperature $T$
Repeat
  Repeat
      Generate a neighbor $X'$ perturbing the solution $X$.
          Acceptance with the criterion of 'Metropolis'// Fig. 7
   Until thermodynamic equilibrium reaches
Decrease temperature
Until maximum iterations or minimum error criteria is attained
end
```

Fig. 6 The pseudo-code of a SA.

```
If $\Delta f<0$
        Update the current solution ($x' \leftarrow x$)
If not
        Compute P=exp($-\Delta f/T$)
        Generate a random number $R\in[0,1]$ (using an uniform distribution)
        If $R\leq P$
        Accept the new solution $x'$ and update $x$
        If not reject the new solution $x'$
        end if
end if
```

Fig. 7 The Metropolis pseudo-code.

Table 1 Advantages of GA and SA

GA	SA
Optimizes with continuous or discrete variables	Very simple to be set up
Deals with a large number of variables	Does not require memory (past) in order to find spaces to seek local following (future)
Optimizes variables with extremely complex –cost surfaces	Can deal with arbitrary systems and cost functions

Table 2 Drawbacks of GA and SA

GA	SA
No guaranteed optimal solution in a finished time,	Important choice of the beginning solution: Knowledge of the problem
Initialization of several parameters, important choice of the methods,	It is necessary to determinate the parameters by hand: initial temperature, elementary modification... by testing various values

5 The Hybrid GA/SA

Merging a local search technique and an evolutionary approach is a very fertile area [39]. Indeed, the idea consists of taking maximum benefits from the robustness of the search technique of a LS method, and those of the recombination process of an evolutionary algorithm (such hybrid algorithms are also known as *memetic* algorithms). At each n iterations (n is a prefixed number), the algorithm applies the LS method to the elements of the (current) population, and then recalls the recombination mechanism in order to generate new elements. In other words, the approach consists of injecting each n iterations a potential solution, obtained using SA techniques, in the GA population, in order to refine the whole solution. It is also to be noticed that accordingly, the algorithm is capable to overcome the premature convergence of the GA algorithm and escape from local optimal solutions.

Tables 1 and 2 summarize main advantages and drawbacks of both GA and SA, respectively [47,48], and Fig. 8 presents the GA/SA flowchart.

In order to show the viability of the proposed approach, the hybrid GA/SA was evaluated using the following test functions [47]. Test functions are given by (3), (4) and (5). Their plots are given in Fig. 9, Fig. 11 and Fig. 13. Corresponding algorithm parameters are given in Table 3.

Table 3 The algorithm parameters

Population size	100
Mutation rate	0.01
Crossover rate	0.9
Initial temperature	1
Stopping temperature	10^{-6}
Cooling schedule	0.9

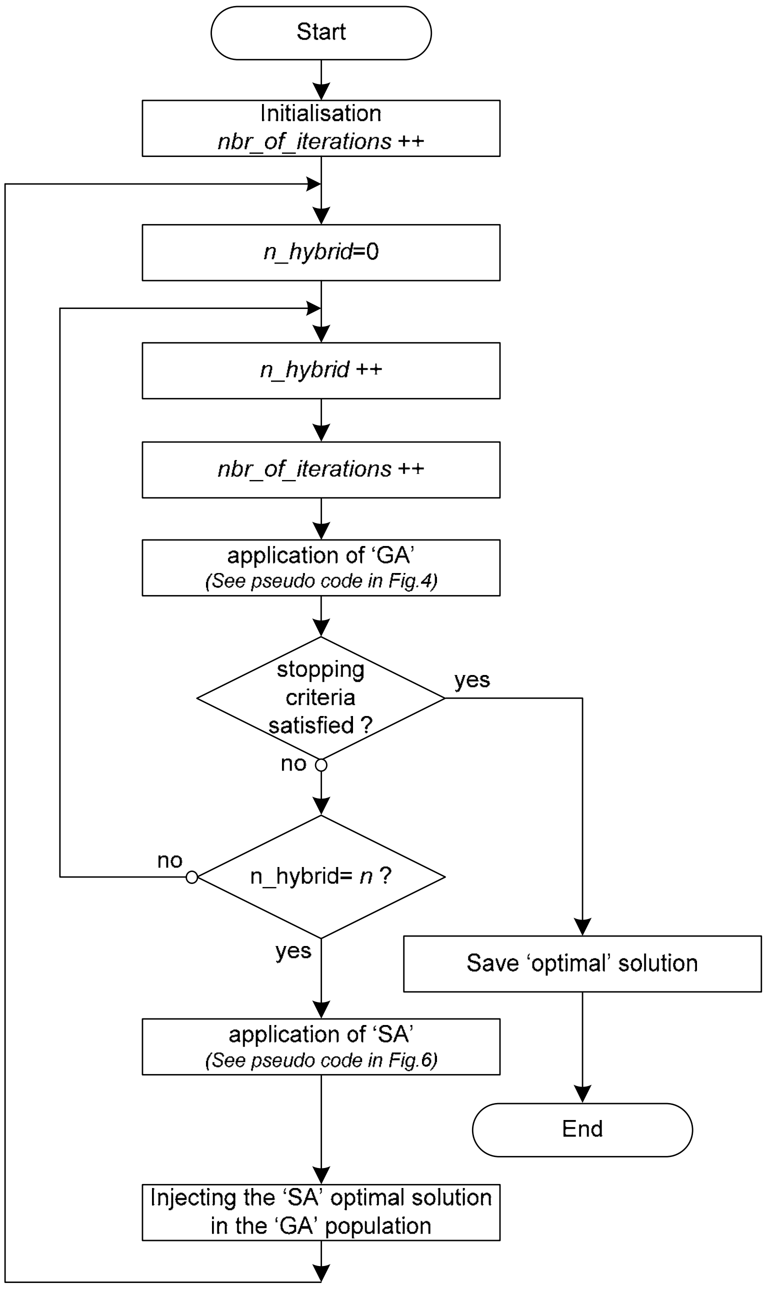


Fig. 8 Flowchart of a hybrid GA/SA approach.

Test function #1:

$$\left| \begin{array}{ll} \textit{Minimize} & f(x,y) = y\sin(4x) + 1.1\, y\sin(2y) \\ \textit{subject to} & x \in [0,10],\ y \in [0,10] \end{array} \right. \tag{3}$$

The goal is to find the global minimum of *f(x,y)* among the large number of local minima.

SA, GA and GA/SA algorithms were applied to *f(x,y)*. Fig. 10 shows a comparison between results obtained using GA/SA and GA, where the rapid convergence of the hybrid algorithm can be noticed. Optimal parameters are (x,y)=(0.9039,0.866), and the optimal fitness is $f(x,y)$=-18.554. Both GA/SA and SA give (x,y)=(0.9040,8.664) and $f(x,y)$=18.559, in 0.9 sec and 14 sec♦, respectively.

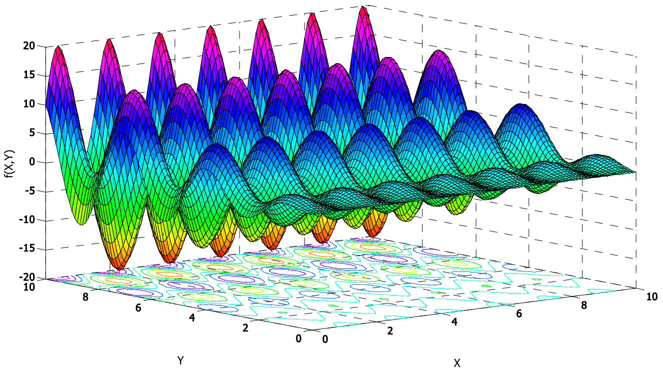


Fig. 9 Plot of the test function #1.

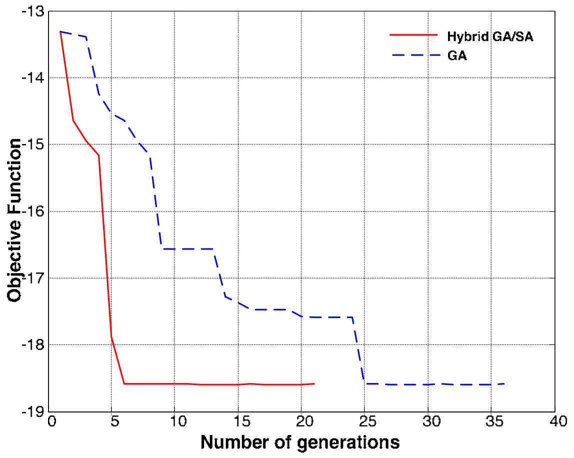


Fig. 10 Optimizing *f(x,y)* (function #1): A comparison between GA and GA/SA results.

♦a (2 GHz, 2 Go RAM) core 2 DUO PC was used for this purpose. The same conditions are respected for the three test functions.

Test function #2:

$$\begin{array}{|ll}\textit{Minimize} & f(x,y) = 0.5 + \dfrac{\sin^2\left(\sqrt{x^2+y^2}\right) - 0.5}{1+0.1(x^2+y^2)} \\ \textit{subject to} & x \in [-10,10],\ y \in [-10,10]\end{array} \qquad (4)$$

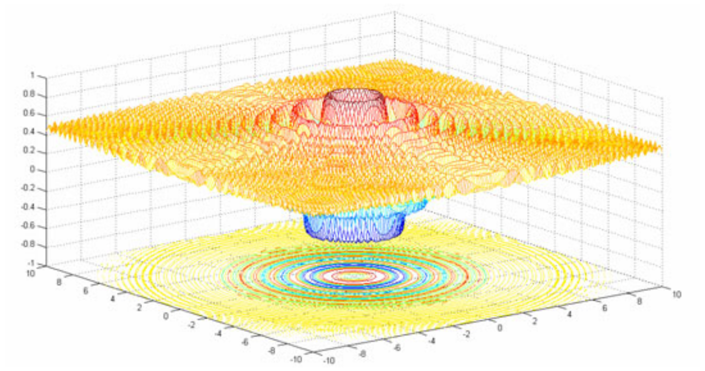


Fig. 11 Plot of the test function #2.

Fig. 12 shows a comparison between results obtained using GA/SA and GA. The rapid convergence of the hybrid algorithm is to be noticed. Optimal parameters are (*x*,*y*)=(1.897,1.006). The optimal fitness is *f*(*x*,*y*)=-0.523. Obtained 'optimal' parameters are (2.107,0.362) and (1.754,1.487) for GA/SA and SA, respectively. Reached fitnesses are respectively -0.522 and -0.374. Convergence times are 1.2 sec (GA/SA) and 20 sec (SA).

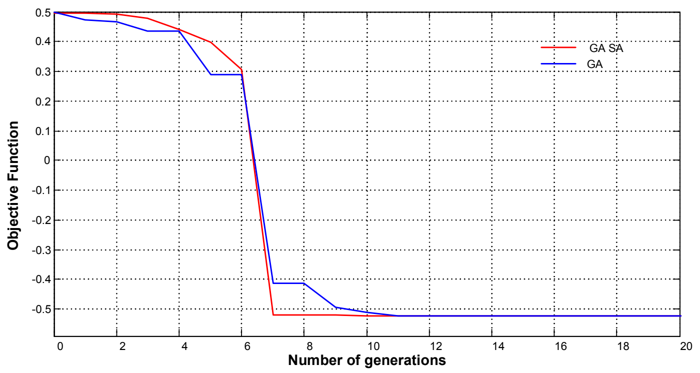


Fig. 12 Optimizing *f(x,y)* (function #2): A comparison between GA and GA/SA results.

Test function #3:

$$\left| \begin{array}{ll} Minimize & f(x,y) = -x.sin(\sqrt{|x-(y+9)|} - (y+9).sin(\sqrt{|y+0.5.x+9|} \\ subject\ to & x \in [-20,20],\ y \in [-20,20] \end{array} \right. \quad (5)$$

Fig. 14 shows a comparison between results obtained using GA/SA and GA, where the rapid convergence of SA/GA algorithm can be easily noticed. For the test function #3, the optimal parameters are (-14.580,-20.000) and the fitness is -23.806. GA/SA gives (-17.007,-20.730) and -25.230 in 0.7 sec, whereas SA gives (-15.356,20.475) and a fitness of -24.646 in 12 sec.

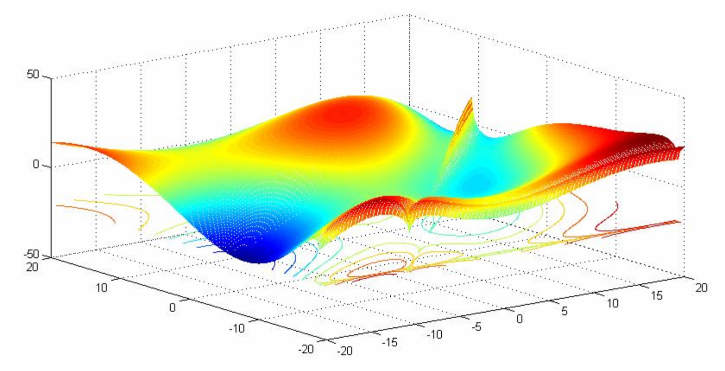


Fig. 13 Plot of the test function #3.

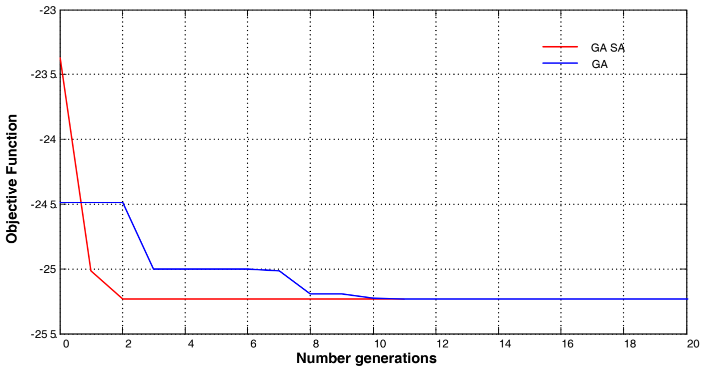


Fig. 14 Optimizing *f(x,y)* (function #3): A comparison between GA and GA/SA results.

6 Application of GA/SA to the Optimization Current Conveyors

Current conveyors are the most well known current mode circuits. They can perform many analogue signal processing functions. Besides, they simplify in many ways the design of analogue circuits in comparison to their voltage mode counterparts, i.e. operational amplifiers [3]. Current conveyors (CC) can be represented as shown in Fig. 15.

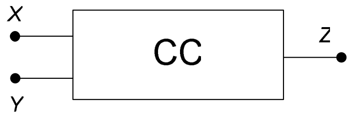


Fig. 15 General representation of current conveyor.

The electric behaviour of a positive second generation current conveyor (CCII+) [3,49] is described as follows:

$$\begin{cases} V_X = V_Y \\ I_Y = 0 \\ I_Z = I_X \end{cases} \quad (6)$$

Fig. 16 presents a translinear loop based CMOS CCII+ [3], where transistors M_1-M_4 instantiate the translinear loop and insure V_X=V_Y. I_0 is the bias current and transistors M_9-M_{12} are current mirrors. Transistors M_5-M_8 reproduce the current applied to pole X, at pole Z.

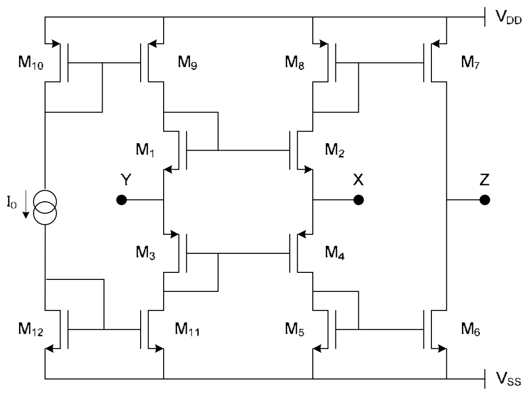


Fig. 16 A CMOS positive second generation current conveyor.

It has been confirmed that current bandwidth limits the frequency application range of a current conveyor, since the voltage frequency range is intrinsically higher than the current one [3]. Thus, in this example we focus on maximizing the current high cut off frequency (f_{ci}) of the CCII+.The symbolic expression of f_{ci} is not given due to their large number of terms.

MOS transistors forming the CCII+ have to operate in the saturation mode. Expressions (7) and (8) give these transistors' saturation conditions for all the MOS transistors:

$$\frac{V_{DD}}{2} - V_{Tn} - \sqrt{\frac{8I_0}{\mu_n C_{ox}\left(\frac{W_i}{L_i}\right)_N}} > \sqrt{\frac{8I_0}{\mu_p C_{ox}\left(\frac{W_i}{L_i}\right)_P}} \tag{7}$$

$$\frac{V_{DD}}{2} - V_{Tp} - \sqrt{\frac{2I_0}{\mu_p C_{ox}\left(\frac{W_i}{L_i}\right)_P}} > \sqrt{\frac{2I_0}{\mu_n C_{ox}\left(\frac{W_i}{L_i}\right)_N}} \tag{8}$$

where μ_n, μ_p and C_{ox} are MOS technology parameters. V_{Tn} and V_{Tp} represent respectively the threshold voltages of NMOS and PMOS transistors. I_0 is the bias current and V_{DD} is the supply voltage. W_i/L_i is the aspect ratio of the corresponding MOS transistor.

GA and GA/SA were applied to maximize f_{ci}. Fig. 17 depicts a comparison between obtained results, where the rapid convergence of the hybrid algorithm can be clearly noticed. Fig. 18 presents SA results (f_{ci} vs stages of temperature). Table 4 gives the algorithm parameters.

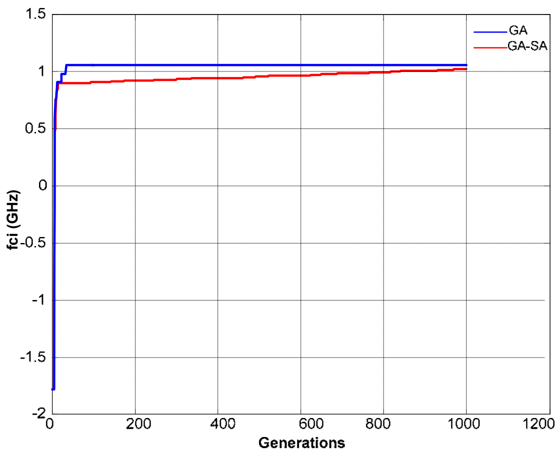


Fig. 17 Optimizing f_{ci}: A comparison between GA and GA-SA results.

Fig. 19 shows SPICE simulation results of the CCII+ current transfer corresponding to the optimal values of the CCII+ parameters.

Table 4 The algorithm parameters of GA/SA

Population size	1000
Numbers of iteration	1000
Mutation rate	0.01
Crossover rate	0.8
Initial temperature	1
Stopping temperature	10^{-6}
Cooling schedule	0.95

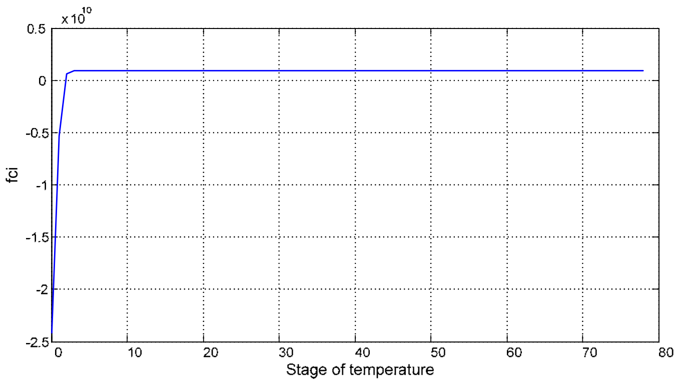


Fig. 18 Optimizing f_{ci} using SA: f_{ci} vs stage of temperature.

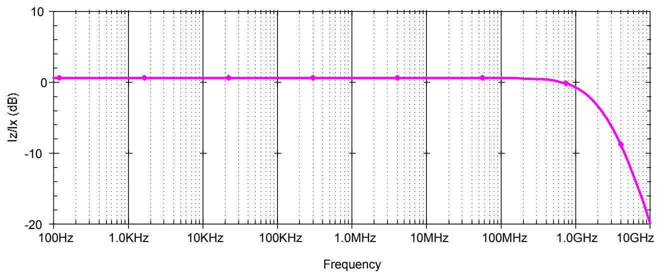


Fig. 19 SPICE simulation results: f_{ci}=931 MHz.

Table 5 presents the optimal device sizing and performances obtained by GA/SA and SA.

Table 5 Optimal device sizing and obtained performances: a comparison.

	GA/SA	**SA**
Technology	0.35 μm AMS	0.35 μm AMS
Voltage supply (V)	±2.5	±2.5
Wn(μm)/Ln(μm)	30.25/0.52	19.30/0.54
Wp(μm)/Lp(μm)	49.95/0.35	33.34/0.35
GBW (GHz)	1.1	0.924
Running Time (min)	0.68	1.25

Finally, and in order to highlight robustness of the proposed hybrid algorithm, Fig. 20 presents results obtained for 100 runs of the hybrid algorithm. The mean value of f_{ci} is 0.956 GHz.

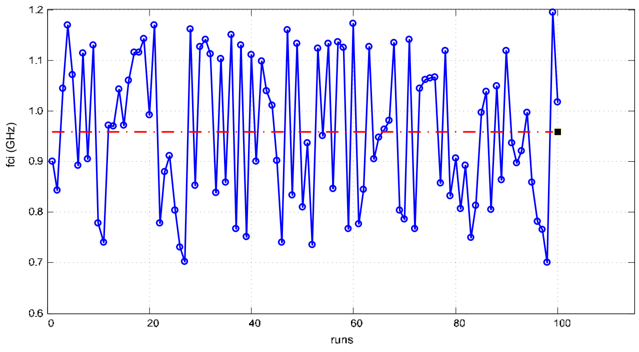


Fig. 20 Values of f_{ci} for 100 runs.

7 Application of GA/SA to the Optimization of Operational Transconductance Amplifiers

The operational transconductance amplifier (OTA), whose schematic symbol is represented in Fig. 21, is an amplifier whose output current is proportional to the differential input voltage. Expression (9) gives the linear function between the differential input voltage and the output current.

$$I_{out} = (V_{in+} - V_{in-})g_m \tag{9}$$

where V_{in+} and V_{in-} are voltages at the non-inverting and the inverting inputs, respectively. g_m is the transconductance of the amplifier.

The amplifier's output voltage is expressed as follows:

$$V_{out} = I_{out}\, R_L \tag{10}$$

R_L is the load of the OTA.

The voltage gain is then the output voltage divided by the differential input voltage:

$$A_V = \frac{V_{out}}{(V_{in+} - V_{in-})} = R_L\ g_m \tag{11}$$

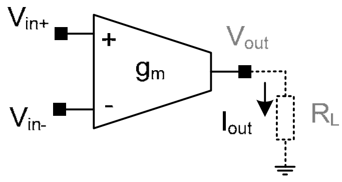


Fig. 21 Schematic symbol for the OTA.

OTAs are generally used to drive small capacitive loads at high frequencies. An OTA is basically an Op-Amp without any output buffer, preventing it from driving resistive or large capacitive loads. They are preferred over op-amps mainly because of their smaller size and their simplicity [50]. Typically, OTAs can be classified into two basic architectures, namely folded-cascode OTAs and telescopic OTAs. The advent of deep submicron technologies enables increasingly high speed circuits, but makes designing high DC gain OTAs more difficult [51]. In this application we deal with maximizing the voltage gain of a fully differential folded cascode OTA (FDFC). Fig. 22 shows the CMOS implementation of a FDFC.

This OTA uses cascoding in the output stage combined with an unusual implementation of the differential amplifier to achieve good input common-mode range. Thus, the folded cascode OTA offers self-compensation, good input common-mode range, and the gain of a two-stage OTA [52].

The open-loop voltage gain (A_v) and the unity-gain frequency (F_t) of the FDFC are given respectively by equations (12) and (13):

$$A_v = g_{m9} R_{out} \tag{12}$$

$$F_t = g_{m9} / 2\pi C_L \tag{13}$$

where $R_{out} = R_2 /\!/ (g_{m3} r_{03} R_1)$, $R_1 = r_{01} /\!/ r_{09}$ and $R_2 = g_{m3} r_{05} r_{07}$. g_{m3} and g_{m9} are respectively the transconductances of transistors M_3 and M_9. r_{01}, r_{03}, r_{05}, r_{07} and r_{09} are

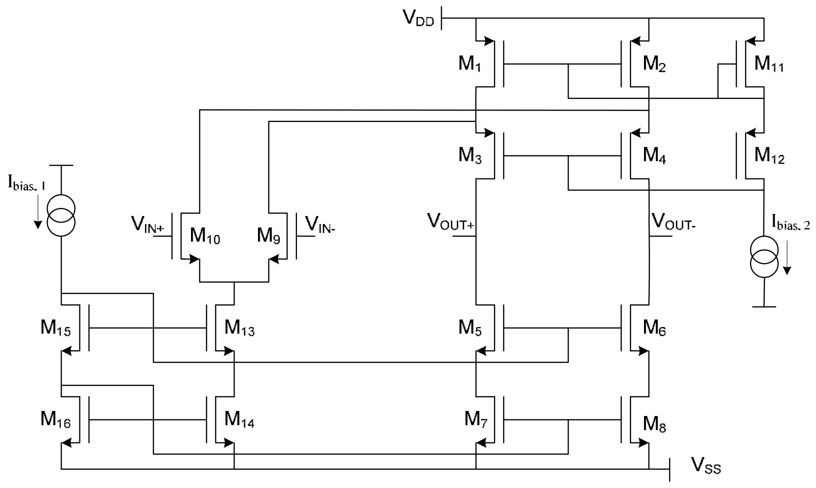


Fig. 22 Folded cascade OTA.

respectively the drain-source resistances of transistors M_1, M_3, M_5, M_7 and M_9. C_L is the load capacitance.

The problem consists in maximizing the voltage gain A_v while satisfying a set of inherent constraints, i.e. MOS saturation conditions, whose expressions are given by Eqns. (14)-(18).

$$V_{DD} - V_{outMAX} \geq \sqrt{\frac{3\,I_{bias1}}{K_P\left(\frac{W_1}{L_1}\right)}} + \sqrt{\frac{2\,I_{bias1}}{K_P\left(\frac{W_3}{L_3}\right)}} \tag{14}$$

$$V_{out\min} - V_{SS} \geq \sqrt{\frac{I_{bias1}}{K_N\left(\frac{W_5}{L_5}\right)}} + \sqrt{\frac{I_{bias1}}{K_N\left(\frac{W_7}{L_7}\right)}} \tag{15}$$

$$V_{DD} - V_{inMAX} + V_{TN} \geq \sqrt{\frac{3\,I_{bias1}}{K_P\left(\frac{W_1}{L_1}\right)}} \tag{16}$$

$$V_{in\min} - V_{SS} - V_{TN} \geq \sqrt{\frac{I_{bias1}}{K_N\left(\frac{W_9}{L_9}\right)}} + \sqrt{\frac{2\,I_{bias1}}{K_N\left(\frac{W_{13}}{L_{13}}\right)}} + \sqrt{\frac{2\,I_{bias1}}{K_N\left(\frac{W_{14}}{L_{14}}\right)}} \tag{17}$$

$$Slew\ Rate = \frac{I_{bias1}}{C_L} \tag{18}$$

where, I_{bias1} is the bias current, W_i and L_i are respectively widths and lengths of the corresponding transistors, V_{DD} and V_{SS} are the supply voltages, K_N and K_P are technology parameters. V_{TN} is the NMOS threshold voltage.

The static and dynamic performances of the OTA are set according to the specifications of high gain, wide band applications given in Table 6 [53]. The optimization problem consists of maximizing the voltage gain A_v while satisfying a set of constraints, namely the saturation conditions of MOS transistors and the frequency bandwidth.

Table 6 OTA circuit specifications

Technology	CMOS AMS 0.35µm
F_t (MHz)	≥ 200MHz
C_L (pF)	0.1
V_{DD}/V_{SS} (V)	-1.8/+1.8
Slew Rate (V/µsec)	≥ 200

Table 7 gives optimal device sizing obtained thanks to GA/SA and GA, the circuit's performance, and comparison with two published works. I_{bias1} equals 60µA. This value gives a slew rate that equals 300V/µsec. I_{bias2} directly depends on the ratio between aspect ratios of the current mirrors.

Table 7 Optimal device sizing of the FDFC OTA

	GA/SA	**SA**	**[54]**	**Gm/Id** [55]
Technology	0.35 µm AMS	0.35 µm AMS	0.35 µm AMS	0.35 µm AMS
Voltage supply (V)	±1.8	±1.8	±1.8	±2
W1(µm)/L1(µm)	50/1	44.03/1	34.85/1	10.8/1
W3(µm)/L3(µm)	33/1	29.06/1	23/1	5.4/1
W5(µm)/L5(µm)	50/1	47.15/1	47.15/1	2/1
W9(µm)/L9(µm)	50/1	49.9/1	49.9/1	14/1
DC Gain (dB)	84.42	83.89	82.89	77.53
Running Time (min)	0.086	0.47	--	--

Fig. 23 shows a comparison between results obtained using GA and hybrid GA/SA when maximizing the open-loop voltage gain (A_v), where the rapid convergence of the hybrid algorithm can be clearly noticed. Fig. 24 presents SA results: Av vs. stages of temperature.

Table 8 gives the GA/SA algorithm parameters.

Table 8 The algorithm parameters of GA/SA

Population size	2000
Numbers of iteration	2000
Mutation rate	0.01
Crossover rate	0.8
Initial temperature	1
Stopping temperature	10^{-6}
Cooling schedule	0.95

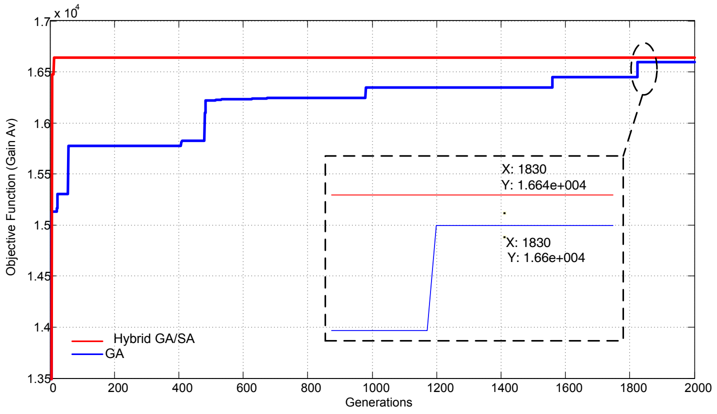


Fig. 23 Optimizing the FDFC OTA Gain: A comparison between GA and GA-SA algorithms.

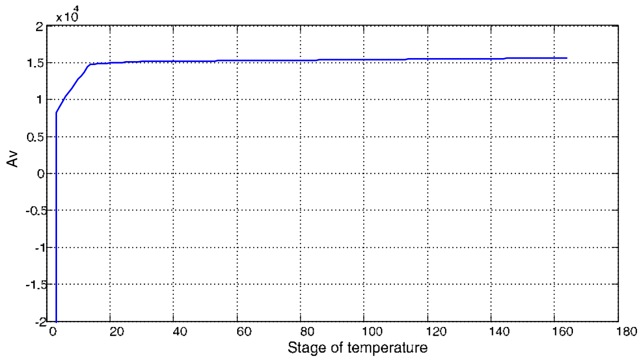


Fig. 24 Optimizing the OTA gain using SA: Av vs stage of temperature.

Table 9 gives a comparison between theoretical (hybrid GA/SA) and SPICE simulation results.

Table 9 Theoretical and simulation results of hybrid GA/SA

Performances	Theoretical values (MATLAB)	Simulation values (SPICE)
A_v(dB)	84.42	84.33
F_t (MHz)	534	517

These values show the good agreement between the SPICE simulation results and the theoretical ones obtained by applying hybrid GA/SA.

Fig. 25 shows SPICE simulation results of the FDFC OTA.

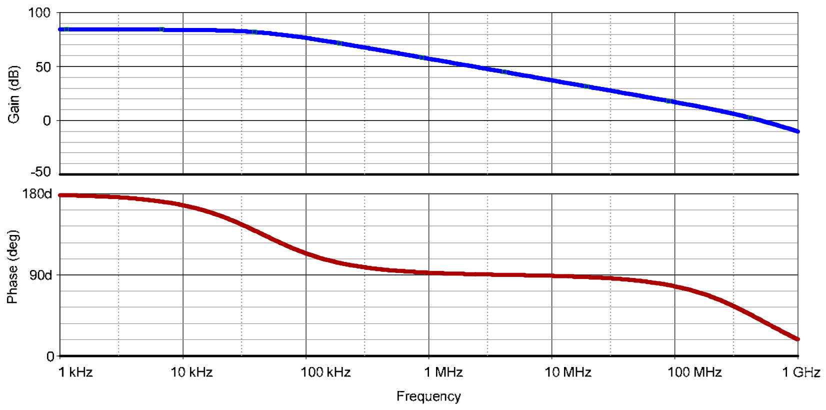


Fig. 25 Spice simulation results for Gain and Phase curve

The folded cascode OTA presents a high gain (84.33 dB), a large unity-gain frequency (517.030 MHz) and a good linearity depicted by its phase margin (51.34°).

Finally, Fig. 26 presents results obtained for 100 runs of the GA/SA algorithm. The mean value of Av is 1503.

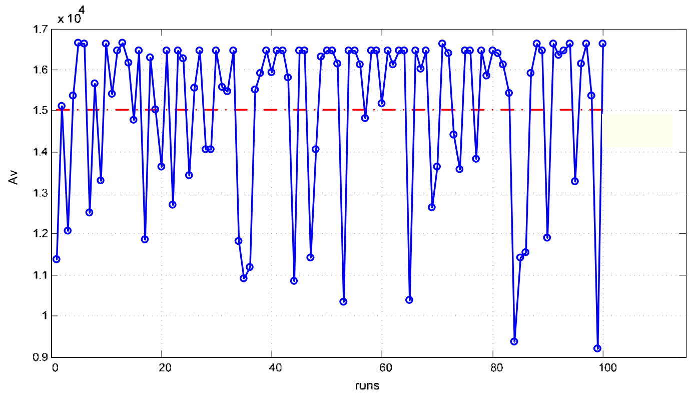


Fig. 26 Values of Av for 100 runs.

8 Application of GA/SA to the Optimization of a RF Circuit

Low Noise Amplifiers (LNA) form an important block in any Radio Frequency chain [56]. The LNA is responsible of amplifying the signal and minimizing the noise [57]. Scattering parameters allow characterizing a LNA circuit [56]. In fact, these parameters reflect the power gain, the input matching and the output matching of the LNA. It has been shown that the symbolic expressions of the scattering parameters can deducted from the symbolic expressions of the Impedance parameters (*Z-parameters*) [59].

The definition of the scattering parameters, noticed *S-Parameters*, is based on the theory of the incident and reflected waves [59,60]. Thus, *S-parameters* describe the relationship between the different waves of a system. Fig. 27 represents a network with two ports including the incident and reflected microwaves [61].

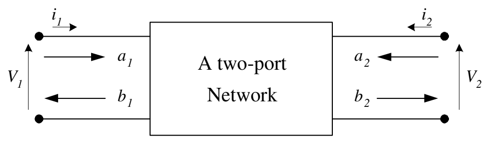


Fig. 27 A two-port network with incident and reflected waves.

a_1 and a_2 represent the electric field of the microwave signal entering the network input and output respectively. b_1 and b_2 represent the electric field of the microwave signal leaving the network input and output respectively.

Consequently, the *S-parameters* are defined by the following expressions [61]:

$$S_{11} = \left.\frac{b_1}{a_1}\right|_{a_2=0},\ S_{21} = \left.\frac{b_2}{a_1}\right|_{a_2=0},\ S_{12} = \left.\frac{b_1}{a_2}\right|_{a_1=0},\ S_{22} = \left.\frac{b_2}{a_2}\right|_{a_1=0} \tag{19}$$

S_{11}, S_{21}, S_{12}, S_{22} are the input reflection coefficient, the forward transmission coefficient, the reverse transmission coefficient, and the output reflection coefficient, respectively.

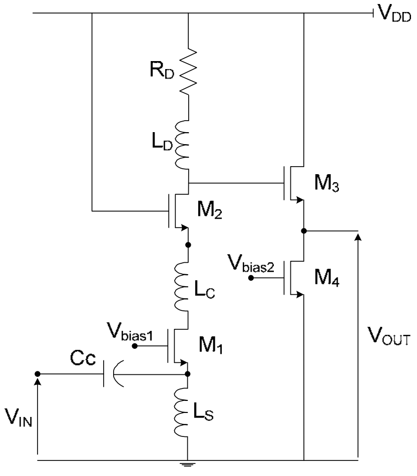


Fig. 28 UWB Common Gate Low Noise Amplifier.

Fig. 28 shows a Common Gate Low Noise Amplifier [62]. This structure is dedicated to the Ultra Wide Band (UWB) standard. The frequency band of this standard is defined from 3 GHz to 10 GHz. The input impedance of this structure is formed essentially by the inductance L_S and the equivalent impedance of the transistor M_1. The transistor M_2 increases low frequency gain and improves isolation. However, the parasitic capacitances of the transistor M_2 decrease the gain at high frequency. Therefore, the role of the inductance L_C is to balance this degradation. Transistors M_3 and M_4 improve the output matching of the structure. The inductance L_D and the resistance R_D permit to obtain a flat gain along the frequency band [3, 10] GHz. Due to the large number of terms of the *S-parameters*, their symbolic expressions are not given; they were generated using the symbolic analyser CASCADES.1 [63,64].

The hybrid GA/SA approach was used to compute the optimal sizing of the transistors forming the common gate LNA and the optimal values of the biases. The objective is to maximize the voltage gain (S_{21}). Design and inherent constraints (maximum acceptable noise figure value, MOS saturation conditions, minimum transition frequency f_T, and impedance matching) are given by expressions (20)-(27).

$$\langle Noise\_Figure \rangle_{dB} < 1dB \tag{20}$$

where

$$Noise\_Figure = 1 + \frac{\gamma}{\alpha g_{m1} R_s} + \frac{\delta\alpha}{5 g_{m1} R_s}\left(\frac{\omega}{\omega_T}\right)^2 + \frac{R_D}{\left(\omega^2 L_D{}^2 + R_D{}^2\right) R_S g_{m1}{}^2 \left(\frac{|Z_{11}|}{|Z_{11}| + R_S}\right)^2}$$

γ, α and δ are process dependant parameters, g_{m1} is the transconductance of transistor M_1, R_S is the terminal (load) resistance which is equal to 50Ω, ω and ω_T are the pulsation and the transit pulsation, respectively. $|Z_{11}|$ is the input impedance of the LNA.

$$(V_{DD} - V_{TN2} - \sqrt{\frac{2 I_{d1} L_2}{\mu_n C_{ox} W_2}} - |S_{21}| V_{inMAX}) > V_{gs1} - V_{TN1} \tag{21}$$

$$(V_{gs2} - R_D I_{d1} - V_{ds1\min} - |S_{21}| V_{inMAX}) > V_{gs1} - V_{TN1} \tag{22}$$

$$(V_{TN3} - \sqrt{\frac{2 I_{d2} L_3}{\mu_n C_{ox} W_3}} - V_{ds4\min}) > V_{gs3} - V_{TN3} \tag{23}$$

$$(V_{DD} - V_{TN3} - \sqrt{\frac{2 I_{d3} L_3}{\mu_n C_{ox} W_3}} - |S_{21}| V_{inMAX}) > V_{gs4} - V_{TN4} \tag{24}$$

$$f_T > 5 f_0 \tag{25}$$

where $f_T = \frac{1}{2\Pi}\frac{g_m}{C_{gs}} = \frac{3}{4\Pi}\frac{\mu_n}{L^2}(V_{gs} - V_{TN})$, f_0 is the central frequency of the considered band, i.e. 6.3 GHz. V_{TNi} is the threshold voltage, V_{inMAX} is the input maximal voltage and V_{gsi} are gate to source voltages of the corresponding transistor (1≤i≤4).

$$|S_{11}|_{dB} < -10dB \tag{26}$$

$$|S_{22}|_{dB} < -10dB \tag{27}$$

Fig. 29 presents a comparison between results obtained using GA and GA-SA. Fig. 30 shows SA results: S_{21} vs. stages of temperature. Table 10 gives the GA/SA algorithm parameters. Table 11 presents values of the optimized parameters and the reached performances.

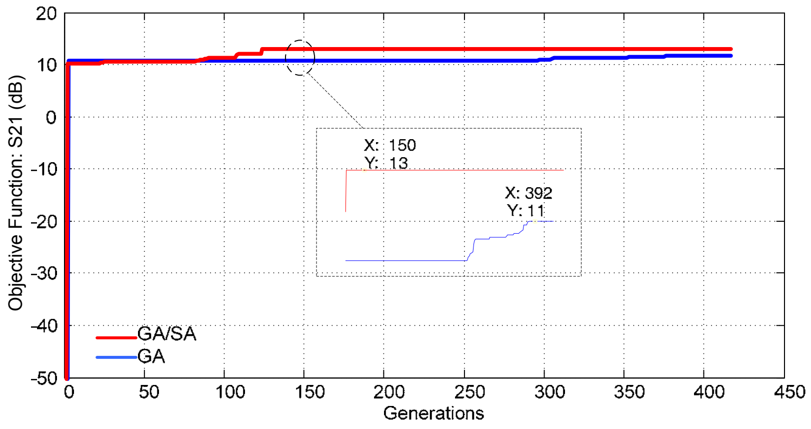


Fig. 29 A comparison between GA and GA/SA results.

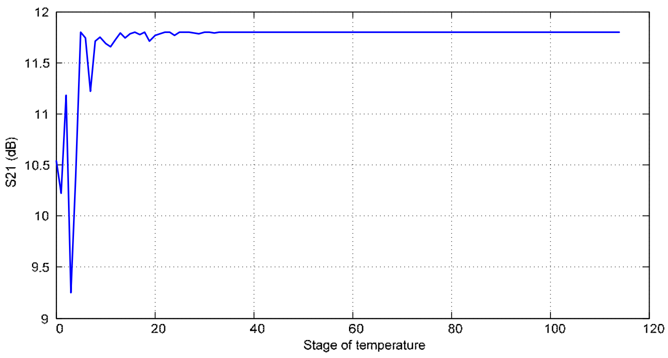


Fig. 30 Optimizing the LNA scattering parameter S_{21} using SA: S_{21} vs stage of temperature.

Table 10 The GA/SA algorithm parameters

Population size	400
Numbers of iteration	400
Mutation rate	0.01
Crossover rate	0.8
Initial temperature	1
Stopping temperature	10^{-6}
Cooling schedule	0.95

Table 11 Optimal device sizing and reached performances for the LNA.

	GA/SA	**SA**
Technology	0.35 μm AMS	0.35 μm AMS
W_1 (μm)/L_1 (μm)	772/0.35	781/0.35
W_2 (μm)/L_2 (μm)	772/0.35	781/0.35
W_3 (μm)/L_3 (μm)	34.07/0.35	31/0.35
W_4 (μm)/L_4 (μm)	34.07/0.35	31/0.35
Id_1 (mA)	5.23	5.18
Id_2 (pA)	6.23	6.23
S_{21}	13.0 dB	11.8 dB
S_{11}	-10.9 dB	-11.6 dB
S_{22}	-15.0 dB	-18.8 dB
Running time (min)	5.2	11.4

Table 12 gives a comparison between theoretical (GA/SA) and simulation (Advanced Design System :ADS) results.

Table 12 Theoretical and simulation results

	Theoretical values (MATLAB)	**Simulation values (ADS)**
S_{21}	13.0 dB	12.5 dB
S_{11}	-10.9 dB	-12.8 dB
S_{22}	-15.0 dB	-12.6 dB

Figs 31-33 show the good agreement between ADS simulation results and the theoretical ones (MATLAB) obtained by applying hybrid GA/SA.

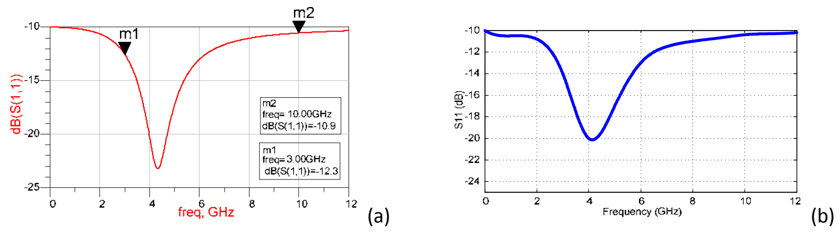


Fig. 31 S_{11}=*f*(frequency): (a) ADS, (b) MATLAB

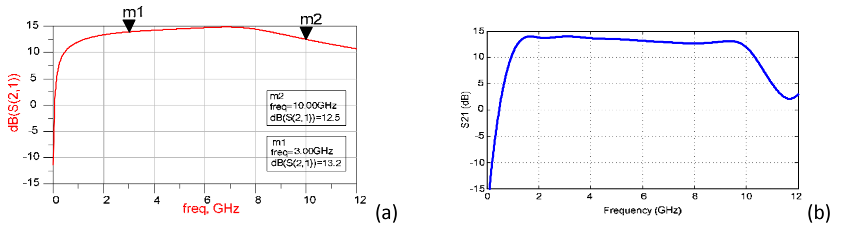


Fig. 32 S_{21}=*f*(frequency): (a) ADS, (b) MATLAB

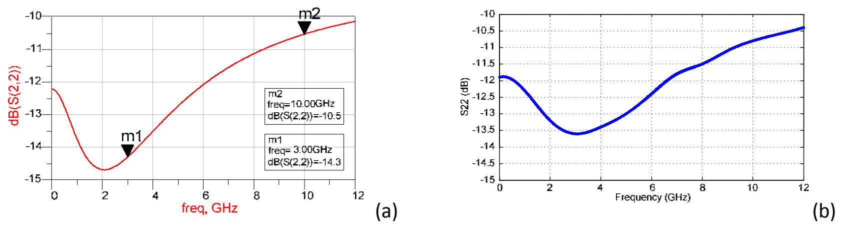


Fig. 33 S_{22}=*f*(frequency): (a) ADS, (b) MATLAB.

Finally, Fig. 34 presents results obtained for 100 runs of the GA/SA algorithm. The mean value of S_{21} is 12.60 dB.

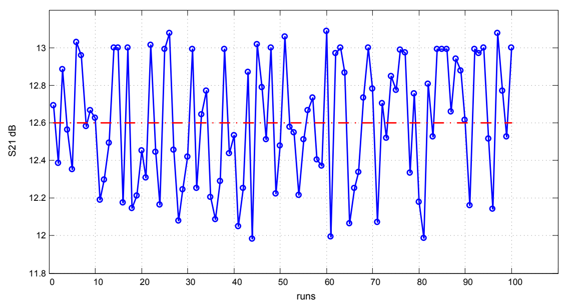


Fig. 34 Values of S_{21} for 100 runs.

9 Conclusion

In this chapter, the hybridization of a genetic algorithm and the simulated annealing technique, and its application to the optimal design of analogue circuits, are proposed. The GA/SA algorithm was implemented in MATLAB. First, it was evaluated using some test functions and advantages of such hybridization, when compared to conventional meta-heuristics, were highlighted. The hybrid GA/SA algorithm was used to optimize the sizing of three typical analogue circuits, namely a positive second generation CMOS current conveyor, a fully differential folded cascode operational transconductance amplifier, and a radio-frequency circuit, i.e. a low noise amplifier. Improvements obtained thanks to the use of GA/SA, in terms of CPU computing time, were highlighted. Further, robustness of the proposed hybrid approach was tested and simulation results (SPICE/ADS) were given to show concordance with theoretical results.

References

[1] Gielen, G., Sansen, W.: Symbolic Analysis for Automated Design of Analog Integrated Circuits. Kluwer Academic Publishers, Dordrecht (1991)
[2] Fakhfakh, M., Tlelo-Cuautle, E., Fernández, F.V.: Design of Analog Circuits through Symbolic Analysis. Bentham Scientific Publisher (2010); eISBN: 978-1-60805-095-6
[3] Toumazou, C., Lidgey, F.J., Haigh, D.G.: Analog Integrated Circuits: The current mode approach. IEEE circuit and systems series 2 (1993)
[4] Roca, E., Fakhfakh, M., Castro-López, R., Fernández, F.V.: Applications of Evolutionary Computation Techniques to Analog, Mixed-Signal and RF Circuit Design – An Overview. In: The IEEE International Conference on Electronics, Circuits, and Systems, ICECS, Tunisia (2009)
[5] Tlelo-Cuautle, E., Duarte-Villasenor, M.A.: Evolutionary electronics: automatic synthesis of analog circuits by GAs. In: Yang Ang, B.L.T., Yin, S. (eds.) Success in Evolutionary Computation. SCI, pp. 165–188. IGI Global (2008)
[6] Liu, B., Fernández, F.V., Gielen, G., Castro-Lopez, R., Roca, E.: A Memetic Approach to the Automatic Design of High-Performance Analog Integrated Circuits. ACM Trans on Design Automation of Electronic Systems 14(3) Article 42 (2009)
[7] Medeiro, F., Rodríguez-Macías, R., Fernández, F.V., Domínguez-Castro, R., Huertas, J.L., Rodríguez-Vázquez, A.: Global Design of Analog Cells Using Statistical Optimization Techniques. Analog integrated circuits and signal processing 6, 179–195 (1994)
[8] Fakhfakh, M., Loulou, M., Masmoudi, N.: A Novel Heuristic for Multi-Objective Optimization of Analog Circuit Performances. Analog Integrated Circuits & Signal Processing 61(1), 47–64 (2009)
[9] Fakhfakh, M.: A Novel Alienor-Based Heuristic for the Optimal Design of Analog Circuits. Microelectronics Journal 40(1), 141–148 (2009)
[10] Talbi, E.G.: A Taxonomy of Hybrid Metaheuristics. Journal of Heuristics 8(5), 541–564 (2002)
[11] Siarry, P., Michalewicz, Z.: Advances in Metaheuristics for Hard Optimization, pp. 1619–7127. Springer, Heidelberg (2007); ISSN: 1619-7127

[12] Abraham, A., Hassanien, A.E., Siarry, P., Engelbrecht, A.: Foundations of Computational Intelligence Volume 3: Global Optimization. Springer, Heidelberg (2009); ISSN:1860949X

[13] Dréo, J., Petrowski, A., Siarry, P., Taillard, E.: Metaheuristics for Hard Optimization: methods and case studies. Springer, Heidelberg (2006); ISBN: 978-3-540-23022-9

[14] Daniel, A.: Evolutionary Computation for Modeling and Optimization. Springer, Heidelberg (2006); ISBN: 978-0-387-22196-0

[15] Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer, Heidelberg (2007); ISBN: 978-3-540-40184-1

[16] Grimbleby, J.B.: Automatic analogue circuit synthesis using genetic algorithms. IEE Proceedings-Circuits, Devices and Systems 147, 319–323 (2000)

[17] Dinger, R.H.: Engineering design optimization with genetic algorithm. In: IEEE Northcon Conference, Seattle WA, USA (1998)

[18] Marseguerra, M., Zio, E.: System design optimization by genetic algorithms. In: The IEEE Annual Reliability and Maintainability Symposium, Los Angeles, California USA (2000)

[19] Fogel, L., Owers, A., Walsh, M.: Artificial intelligence through simulated evolution. Wiley, Chichester (1996); ISBN-13: 978-0471265160

[20] Chan, F.T.S., Tiwari, M.K.: Swarm Intelligence: focus on ant and particle swarm optimization. I-Tech Education and Publishing (2007); ISBN 978-3-902613-09-7

[21] Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference On Neural Networks, pp. 1942–1948.

[22] Clerc, M.: Particle swarm optimization. In: International Scientific and Technical Encyclopedia (2006); ISBN-10: 1905209045

[23] Dorigo, M., Di-Caro, G., Gambardella, L.M.: Ant algorithms for discrete optimization. Artificial life Journal 5, 137–172 (1999)

[24] Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems Magazine, 52–67 (2002)

[25] Siarry, P., Berthiau, G., Durdin, F., Haussy, J.: Enhanced simulated annealing for globally minimizing functions of many-continuous Variables. ACM Transactions on Mathematical Software 23, 209–228 (1997)

[26] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Journal of Science 220, 671-220-680 (1983)

[27] Courat, J.P., Raynaud, G., Mrad, I., Siarry, P.: Electronic component model minimization based on Log Simulated Annealing. IEEE Transactions on Circuits and Systems 41, 790–795 (1994)

[28] Durbin, F., Haussy, J., Berthiau, G., Siarry, P.: Circuit performance optimization and model fitting based on simulated annealing. International Journal of Electronics 73, 1267–1271 (1992)

[29] Glover, F.: Tabu search- part I. ORSA Journal on Computing 1(3) Summer (1989)

[30] Glover, F.: Tabu search- part II. ORSA Journal on Computing 2(1) Winter (1990)

[31] Fakhfakh, M., Cooren, Y., Sallem, A., Loulou, M., Siarry, P.: Analog Circuit Design Optimization through the Particle Swarm Optimization Technique. Analog Integrated Circuits & Signal Processing 63(1), 71–82 (2009)

[32] Tlelo-Cuautle, E., Duarte-Villasenor, M.A., Guerra-Gomez, I.: Automatic synthesis of VFs and VMs by applying genetic algorithms. Circuits, Systems, Signal Processing 27, 391–403 (2008)

[33] Tlelo-Cuautle, E., Guerra-Gomez, I., Reyes-Garcıa, C.A., Duarte-Villasenor, M.A.: Synthesis of Analog Circuits by Genetic Algorithms and their Optimization by Particle Swarm Optimization. In: Chiong, R. (ed.) Intelligent Systems for Automated Learning and Adaptation: Emerging Trends and Applications, pp. 173–192. IGI Global (2010), doi: 10.4018/978-1-60566-798-0. ch008

[34] Chu, M., Allstot, D.J.: Elitist nondominated sorting genetic algorithm based RF IC optimizer. IEEE Transactions on Circuits and Systems – I 52, 535–545 (2005)

[35] Yoshida, H., Kawata, K., Fukuyama, H., Takayama, S., Nakanish, Y.: A particle swarm optimization for reactive power and voltage control considering voltage security assessment. The IEEE Transactions on Power Systems 15, 1232–1239 (2001)

[36] Maitra, M., Chatterjee, A.: A novel technique for multilevel optimal magnetic resonance brain image thresholding using bacterial foraging. Elsevier B.V. 41, 1124–1134 (2008)

[37] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. Journal of Chemical Physics 6(21), 1087–1092 (1953)

[38] Kerlner, V., et al.: A hybrid optimization technique coupling an evolutionary and a local search algorithm. Journal of Computational and Applied Mathematics 215, 448–456 (2008); ISSN:0377-0427

[39] Crina, G., Ajith, A., Hisao, I.: Hybrid Evolutionary Algorithms. Springer, Heidelberg (2007)

[40] El-Hosseini, M.A., Hassanien, A.E., Ajith, A., Al-Qaheri, A.: Genetic Annealing Optimization: Design and Real World Applications. In: The International Conference on Intelligent Systems Design and Applications, Kaohsiung, Taiwan (2008)

[41] Kurbel, K., Schneider, B., Singh, K.: Solving optimization problems by parallel recombinative simulated annealing on a parallel computer-an application to standard cell placement in VLSI design. IEEE Transactions on Systems, Man and Cybernetics, Part B 28(3), 454–461 (1998)

[42] Zhang, L., Raut, R., Jiang, Y., Kleine, U.: Placement algorithm in analog-layout designs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(10), 1889–1903 (2006)

[43] Somani, A., Chakrabarti, P.P., Patra, A.: Mixing Global and Local Competition in Genetic Optimization based Design Space Exploration of Analog Circuits. In: The Design, Automation and Test in Europe Conference, Munich, Germany (2005)

[44] Esbensen, H., Mazumder, P.: SAGA: A unification of the genetic algorithm with simulated annealing and its application to macro-cell placement. In: The International Conference on VLSI Design, Calcutta, India (1994)

[45] Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)

[46] Weise, T.: Global optimization algorithms – theory and applications (2009), `http://www.it-weise.de`

[47] Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms. John Wiley & Sons, Chichester (2004); ISBN 0-471-45565-2

[48] Rutenber, R.A.: Simulated Annealing Algorithms: An Overview. IEEE Circuits and Devices Magazine 5(1), 19–26 (1989)

[49] Sedra, A., Smith, K.C.: A second generation current conveyor and its applications. IEEE Transactions on Circuit Theory 17, 132–134 (1970)

[50] Razavi, B.: Design of Analog CMOS Integrated Circuit. TheMcGraw-Hill Companies, Inc., United States (2001); ISBN:0-07-118815-0

[51] Berntsen, O., Wulff, C., Ytterdal, T.: High Speed, High Gain OTA in a Digital 90 nm CMOS Technology. In: NORCHIP Conference, pp. 129–132 (2005)
[52] Phillip, E., Allen, D., Holberg, R.: CMOS Analog Circuit Design. Oxford University Press, Inc., Oxford (2002)
[53] Zhang, L., Kim, H.J., Nadig, V., Ismail, M.: A 1.8 V tri-mode ΣΔ modulator for GSM/WCDMA/WLAN wireless receiver. Analog Integrated Circuits and Signal Processing 49(3), 323–341 (2006)
[54] Daoud, H., Bennour, S., Ben-Salem, S., Loulou, M.: Low power SC CMFB folded cascode OTA optimization. In: The IEEE International Conference on Electronics, Circuits and Systems, ICECS, Malta (2008)
[55] Daoud, H., Ben-Salem, S., Zouari, S., Loulou, M.: Design of folded cascode OTA in different regions of operation through gm/ID methodology. The world academy of science, engineering and technology 45, 28–33 (2008)
[56] Razavi, B.: RF Microelectronics. Prentice Hall press, Englewood Cliffs (1998)
[57] Andreani, P., Sjoland, H.: Noise optimization of an inductively degenerated CMOS low noise amplifier. IEEE Transactions on Circuits and Systems 48(9), 835–841 (2001)
[58] Ellinger, F.: Radio frequency integrated circuits and technologies. Springer, Heidelberg (2007)
[59] Kurokawa, K.: Power waves and the scattering matrix. IEEE Transactions on Microwave Theory and Techniques 13(2), 194–202 (1965)
[60] Kurokawa, K.: An introduction to the theory of microwave circuits. Academic Press, London (1969)
[61] Scott, A.W.: Understanding Microwaves.TK7876.S36. John Wiley & Sons. Inc., Chichester (1993) ISBN 0-471-57567-4
[62] Heng, Z., Xiaohua, F., Sánchez, S.E.: A Low-Power, Linearized, Ultra-Wideband LNA Design Technique. IEEE Journal of solid-state circuits 44(2), 320–339 (2009)
[63] Fakhfakh, M., Loulou, M.: A Software for the Automated Computing of Symbolic Transfer Functions of Analog Circuits. In: The International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design, Erfurt, Germany (2008)
[64] Fakhfakh, M., Loulou, M.: Live Demonstration: CASCADES.1: a Flow-Graph-Based Symbolic Analyzer. In: The IEEE International Symposium on Circuits and Systems, ISCAS, Paris, France (2010)

Evolutionary Inventory Control for Multi-Echelon Systems

Dilay Çelebi

Abstract. The purpose of this chapter is to present the use of Genetic Algorithm (GA) for solving multi-echelon inventory problems. The literature of GA dealing with inventory control problems is briefly reviewed with particular focus on multi-echelon systems. A novel GA based solution algorithm is introduced for effective management of a stochastic inventory system across a distribution network under centralized control. To demonstrate the performance of proposed GA structure, several test cases with different operational parameters are studied and experimented. The percentage differences between the total cost obtained by GA and the lower bounds and simulation results are used as performance indicators. Findings of the experiments show that the proposed GA approach can be very useful for obtaining feasible and satisfying solutions for the centralized inventory distribution problem.

1 Introduction

Most consumer or industrial products are manufactured in and distributed through multi-echelon systems. Inventory control is critical in multi-echelon systems because of the financial necessity of maintaining a sufficient supply of products to meet both customers' needs and manufacturing requirements. Opportunity cost is the main component of inventory related costs; money tied up in inventories is not available for some other use. Inventories also create additional operational cost by consuming physical space, personnel time, and capital. Holding of inventories can cost anywhere between 20% and 40% of the product value, hence the effective management of inventory is critical in supply chain operations (Ballou, 1999).

The importance of a good inventory management in a supply chain is fully recognized by practitioners and researchers. Besides the traditional inventory management problems, the variability of orders increases in moving up from the

Dilay Çelebi
Istanbul Technical University, 34367 Maçka, Istanbul
e-mail: celebid@itu.edu.tr

downstream members to upstream members in the supply chain. This phenomenon is known as *bullwhip effect* (Lee et al., 1997) which causes excessive inventory, loss of revenue, low customer service levels, and inaccurate production plans throughout supply chain systems. Much of the literature has shown that the bullwhip effect can be minimized through information sharing and synchronized inventory control in the supply chain (Cachon and Fisher, 2000). Hence the supply chain performance might be improved by the lowering of inventory levels and the reduction of the cycle times.

The scope of this chapter is confined to use of Genetic Algorithms (GA)s for handling operational issues of inventory control and management in multi-echelon inventory networks. The chapter is structured in six parts. Section 2 gives some fundamental definitions and briefly explains the complexity of multi-echelon inventory control problems. Section 3 provides a review of literature on use of GAs to solve multi echelon inventory control problems and evaluates the state of GA applications in these areas. The studies are classified into three categories according to the network structures they handle. Section 4 presents a novel GA structure for a stochastic lot sizing problem in a centralized distribution system. Model structure and the steps involved in development of the proposed GA scheme, such as chromosome representation, initialization, fitness function development, and determination of operational parameters are explained in detail in relevant subsections. The performance of GA is demonstrated through experiments conducted on several test cases with different operational parameters. Finally, section 5 discusses the issues and boundaries related to the application of the GA on inventory control problems. The chapter is closed with discussions for further research directions.

2 Inventory Control in Multi-Echelon Systems

A multi-echelon inventory system refers to a multistage production/inventory system in which each stage obtains its supply from its predecessor(s) and supplies its supply to its successor(s). Inventory control in multi-echelon systems deals with the problem of determining the best replenishment sizes of items at each stage, mostly with the purpose of minimizing the total cost of the system which usually covers the costs of carrying inventories, costs of making orders, costs of inter-location transfers, and costs of shortages.

Several multi-echelon inventory/production systems can be modeled as a serial system (Clark and Scarf, 1960). In a serial system, each installation has at most one predecessor and at most one successor. An illustration of a serial system is given in Figure 1. The customer demand only arises at the lowest level. Each installation is replenished from its predecessor and the highest installation replenishes from an outside supplier. It is, in general, considerably easier to deal with serial systems than with other types of multi-echelon systems. The main reason to discuss such systems is to obtain preliminary results to study more complex systems.

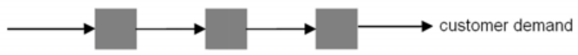


Fig. 1 A Serial Inventory System.

Within a manufacturing context, a final product is sometimes the result of a process which can be decomposed into several levels, broadly corresponding to assembly activities. As illustrated in Figure 2, in an assembly system, each installation has at most one immediate successor. In such systems, the safety stock levels should be positioned wisely and the assembly schedule should be done carefully for effectively managing the component procurement and maintaining service levels on the demand side.

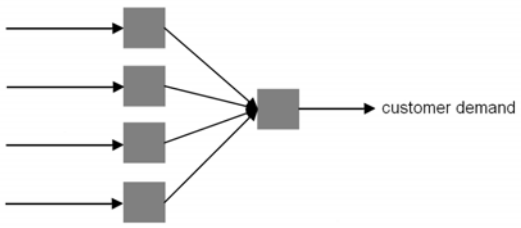


Fig. 2 An Assembly Inventory System.

Meanwhile, inventory distribution systems are generally divergent. A distribution system involves a number of installations at the lowest level which satisfy customer demand and in turn act as customers of higher level installations. Figure 3 shows such a system with two levels: a central warehouse and a number of retailers.

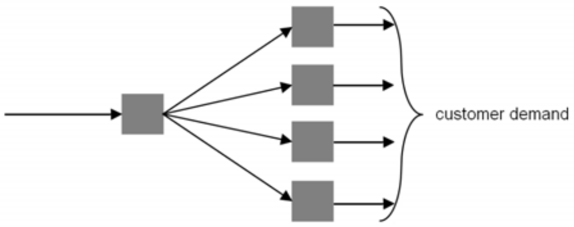


Fig. 3 A Distribution Inventory System.

There exists substantial amount of studies for multi-echelon inventory control, concerned with the analysis and modeling of systems under different operating parameters and modeling assumptions. Extended reviews about the topic might be found in van Houtum et al. (1996) and Gumus and Guneri (2007).

Two distinct configurations for multi-level inventory systems can be considered according to the center of management. First is the *decentralized* systems, where each member of the network takes replenishment decisions on its own and based on only local data. Though it is simple to construct and control such systems, it may not be the most effective. Recently, to increase the competitiveness and effectiveness of their supply chains, companies have begun to set cooperative agreements to manage inventory, which requires sharing demand information and setting mutually agreed upon performance targets for the supply chain. However, most of the time, it's mistakenly assumed that efficiency can be attained simply by sharing information and forming *strategic alliances* within supply chain partners (Silver et al., 1998). In fact, only few companies are able to fully exploit the advantages of collaboration in their supply chains (Holweg et al., 2005), because incorporating customer demand information into inventory control processes to develop sound inventory management is critical to long term survival and competitive advantage. It is important not only to exchange information, but equally, to alter the replenishment and planning decision structure so that a range of additional benefits can be achieved. As a result, a second type of systems become popular as *centralized* systems, in which the stock control activities of the whole system become concentrated within a particular member or group of members. These members take the full control of the inventory replenishment of the chain, and use demand and cost visibility in planning supply operations. The centralization of inventory management might provide cost reductions and improved service levels due to the decreased uncertainty and better utilization of resources for production and transportation (Waller et al., 1999).

Considering centralized solutions for inventory control in supply chains introduces computational difficulty. Schwarz (1973) shows that in a one-warehouse, multi-retailer situation, the form of the optimal policy can be very complex; in particular, it requires that the order quantity at one or more of the locations vary with time, even if all relevant demand and cost factors are time-invariant. Federgruen (1993) notes that algorithms for determining optimal strategies are complicated even for most deterministic demand systems, and complexity dramatically increases in models with stochastic demand.

For centralized control of multi-echelon systems, Clark and Scarf (1960) introduce the concept of *echelon stock*. Echelon stock consists of the stock at any given installation plus stock in transit to or on hand inventory at a lower installation. They have shown that order-up-to policies based on echelon stock inventories are optimal for serial inventory systems with periodic review. Their optimality results for serial systems are later generalized to an infinite time horizon by Federgruen and Zipkin (1984), to assembly systems by Rosling (1989), and to batch ordering by Chen (2000). However, determination of optimal lot sizes by provided models for large scale problems still suffers from computational burden. Moreover, it is not possible to show that echelon stock inventory policy is optimal for distribution systems due to the allocation problem, and for some cases a strategy based on echelon stock inventories might be inferior to a installation stock based strategy (Chen, 2003). The optimization of such systems requires the analysis of a multi-dimensional dynamic programming.

There isn't any known stochastic, multi-period, multi-location model capable of handling complex systems like inseparable cost structures and nonlinear transportation costs (Federgruen, 1993; Chen, 2003). In a practical setting, it is considered too difficult to solve the distribution lot sizing problem by dynamic programming numerically due to the curse of dimensionality. Even for the smallest number of retailers and periods, the exact solution is considered to be impractical (Federgruen and Zipkin, 1984).

3 GAs for Multi-Echelon Inventory Problem

During the last two decades, the opportunities for efficient control of multi-echelon inventory systems have increased substantially (Axsater, 2003). One reason is new information technologies which have created a completely different infrastructure and increased the possibilities for efficient supply chain coordination. Another reason is progress in research, which has resulted in new and efficient techniques for solving hard combinatorial optimization problems. Meta-heuristics such as tabu search, GA and simulated annealing, are examples of such tools which have become popular tools for solving multi-echelon inventory control problems due to computational complexity of such problems.

GA has received considerable attention regarding their potential as an effective optimization technique (Gen and Cheng, 2000). First pioneered by Holland (1975), GA is powerful stochastic search and optimization technique based on principles of natural selection and evolution that has been widely studied, experimented and applied in many fields in engineering (Goldberg, 1989; Holland, 1975). Many of the real world problems, which might be difficult to solve by traditional methods but are ideal for GA.

There exists wide range of studies which implement GA to cope with the multi-echelon inventory management problem. Table 1 provides a review of how GAs have been used to solve multi-echelon inventory problems and following sections give brief summaries of these studies, classified under three main categories according to the network structure they handle.

Table 1 Use of GAs in Multi-echelon Inventory Problems

Study	Objective	Network Structure	Demand Structure
Vergara et al. (2002)	Determination of product scheduling and replenishment sequences of products in a synchronized supply chain to minimize the total costs	Assembly	Deterministic
Syarif et al. (2002)	Minimization of the total transportation, inventory, order and warehouse costs while fully supplying deterministic demand	Distribution	Deterministic
Kimbrough et al. (2002) and O'Donnell et al. (2006)	Minimization of the bullwhip effect by in a serial supply chain model based on the MIT beer game	Serial	Stochastic
Yokoyama (2002)	Determination of the target order-up-to levels for Distribution Center (DC)s and transportation quantities to minimize the expected total costs in a single item distribution system	Distribution	Stochastic
Berretta and Rodrigues (2004)	Determination of the quantity to be produced in different periods in a planning horizon, such that an initially given demand forecast can be attained in a multistage production system with capacity constraints	Assembly	Deterministic
Daniel and Rajendran (2005)	Optimization of the inventory levels by a simulation-based GA to minimize the total supply chain cost	Serial	Stochastic
Han and Damrongwongsiri (2005)	Formulating a model to define stochastic, multi-period, two-echelon inventory with the many-to-many demand-supplier network problem to develop a (R,s) inventory management system	Distribution	Stochastic
Torabi et al. (2006)	Evaluation of optimal lot sizes and delivery schedule that would minimize the average of holding, setup, and transportation costs per unit time	Assembly	Deterministic
Wang and Wang (2008)	Implementation of (Han and Damrongwongsiri, 2005) to solve a real industry case	Distribution	Stochastic
Fakhrzad and Khademi Zare (2009)	Optimization of lot-sizes in a complex multi-stage production scheduling problems with production capacity constraint	Serial	Deterministic
Hnaien et al. (2009)	Optimization of a two-level assembly system under lead time uncertainties, where the finished product demand for a given due date is supposed to be known	Assembly	Stochastic

3.1 Serial Networks

Fakhrzad and Khademi Zare (2009) present a combination of GA with Lagrange multipliers for lot-size determination in a multi-stage, multi-product and multi-period production scheduling problem. First, the original problem is converted to several individual problems using a heuristic approach based on the limited resource Lagrange multiplier. Then, each individual problem has been solved using GA combined with one of the neighborhood search techniques. Each chromosome is represented by a $(T \times 2m)$ matrix where m is the number of elements and T is the number of periods. This representation consists of lot-size (X) and inventory values (I) for each element in each period which is illustrated as follows:

$$\begin{bmatrix} X_{11} & X_{12} & \dots & X_{1\mathrm{T}} & I_{11} & I_{12} & \dots & I_{1\mathrm{T}} \\ X_{21} & X_{22} & \dots & X_{2\mathrm{T}} & I_{21} & I_{22} & \dots & I_{2\mathrm{T}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{\mathrm{m}1} & X_{\mathrm{m}2} & \dots & X_{\mathrm{mT}} & I_{\mathrm{m}1} & I_{\mathrm{m}2} & \dots & I_{\mathrm{mT}} \end{bmatrix}$$

Fitness function is developed to have two modes. One mode represents the cost value for the feasible solution and the other indicates the feasibility of the solution. A solution for the initial population is obtained using WagnerWhitin (WW) algorithm. Two different combinatorial operations, namely a memetic algorithm and WW combination, are used for crossover operation to generate new offsprings. Experiment results over a set of 60 test problems show that proposed hybrid algorithm solves the problem in much less time, with better solutions and lower costs compared to memetic algorithm and CPLEX solution.

Daniel and Rajendran (2005), study the performance of a single-product serial supply chain operating with a base-stock policy. A single period, multi-echelon, single product model is formulated to optimize the inventory (i.e. base stock) levels in the supply chain to minimize the total supply chain cost, comprising holding and shortage costs for all installations in the supply chain. A set of actual values of the base-stock levels are used to code all the genes in a chromosome that represent every member in the chain. The chromosome length is set equal to the number of installations, because every installation in the supply chain is assumed to operate with a particular base-stock level. Every chromosome in the population is evaluated through simulation and fitness value is computed by using the objective function value. Test results illustrate that the proposed GA performs superior to a random search procedures for defined experiment sets.

Both Kimbrough et al. (2002) and O'Donnell et al. (2006) manage to decrease the bullwhip effect by implementing GA on a serial supply chain model based on the MIT beer game. Each player in the game makes their own ordering decisions based on only the orders from the next downstream player. The inventory and backorder cost are calculated for every period in the simulation experiments with each unit cost being constant throughout and the sum of the two types of costs, called total cost, is used as the criteria for GAs to determine the optimal ordering policies.

3.2 Assembly Networks

Torabi et al. (2006) investigate the lot and delivery scheduling problem in a simple supply chain where a single supplier produces multiple components on a flexible flow line and delivers them directly to an assembly facility. The main objective is to find a lot and delivery schedule that would minimize the average of holding, setup, and transportation costs per unit time for the supply chain where the decision variables are production sequence vectors at each stage and machine. The problem is formulated as a mixed integer nonlinear program and a Hybrid Genetic Algorithm (HGA) is developed to solve the problem. The proposed HGA incorporates a neighborhood search into a basic GA that enables the algorithm to perform genetic search over the subspace of local optima.

Two different encoding schemes are considered to represent the discrete part of a solution for the problem. In the first format, each chromosome is composed of m sub-chromosomes, where m gives the number of stages. So that, each sub-chromosome represents the production schedule in that stage. At stages with only one machine, the corresponding sub-chromosomes are permutation vectors of size n where n is the number of components. At stages with multiple parallel machines, the corresponding sub-chromosomes are composed of a component symbol list and a partitioning symbol list, in which integers are used to represent sequence of components and asterisks are used to designate the partition of components to the machines. This first format is good for representing a complete solution for the problem, because it covers the entire solution space and there is a unique string associated with every solution for the problem. However, there are difficulties associated with crossover and mutation operations, so a new encoding scheme is considered. The second format relates to the set of permutation vectors of size n. Each permutation vector represents the order in which the given set of components is processed at each stage. Such a vector by itself does not specify the complete solution, so an appropriate procedure out of two constructive heuristics is used to construct a complete solution for every given permutation vector. The fitness function is set equal to the objective function of the problem and the optimal lot sizes associated to each solution representing a given production schedule is solved by a NLP.

Vergara et al. (2002) develop an evolutionary algorithm that calculates the production sequence at each supplier that would minimize transportation, setup, and inventory holding costs across a multi-component assembly system. The goal of the proposed GA is to determine a common delivery cycle time and production sequence of components for each member of a synchronized supply chain. Integer value representation is used for encoding the solutions. First three places in each chromosome contain the total cost, synchronized delivery time, and the minimal cycle time. The rest of the chromosome is composed of the production sequences of components for each supplier. The last space in the chromosome is used for holding the cost of the assembly center. The performance of the algorithm is tested through the comparisons with an enumeration procedure that identifies the global minimum. On the average, GAs are observed to find the global optimum in 97% of the cases and the error term is less than 0.0038 in the remaining cases.

Berretta and Rodrigues (2004) deal with the multistage capacitated lot-sizing problem with an objective of determining the production lot-sizes of multiple items that minimizes the production, inventory and setups costs subject to demand and capacity limitations. A memetic algorithm approach is used for solution of the problem. Each solution is represented by a matrix of size $2N \times T$ (where N is number of items and T number of periods), with lot-size and inventory of each item in each period. Each solution of the initial population is created using the WW algorithm. Fitness of each solution is composed of two values, one for value of objective function, the other for representing the feasibility of the solution. Each chromosome uses a local search algorithm to improve its current fitness value and transfers the best solution to the population.

Two different crossover schemes are used to stimulate diversity of solutions in the population. First crossover method determines a production lot size of offspring randomly over parent chromosomes and updates the inventory level accordingly. Second method uses WW algorithm, which changes the setup costs of some items randomly and develops a solution for each item by WW algorithm. The performance of the algorithm is tested through comparisons of solutions with lower bounds evaluated by Lagrangean Relaxation in three groups of instances. The average gaps between the lower bound and the heuristic solution are observed to be between than $11-12\%$.

Hnaien et al. (2009) examine supply planning for two-level assembly systems under lead time uncertainties. They handle an optimization problem for a single period system where the finished product demand for a given due date is supposed to be known. The objective is to find the component release dates in order to minimize the sum of the holding costs of the components and the shortage cost for the finished product. Each chromosome has been coded with an array of integer numbers where each gene of a chromosome represents an order release date. Therefore, with a chromosome length equal to number of components, a complete encoding that ensures that all solutions to the problem is represented and considered by the algorithm. The expected total cost of the system is used to evaluate the fitness of each individual of the population. Two selection phases are considered for generation of new generations. First is a reproduction selection to determine the individuals on which the evolutionary operators will be applied. Second is a replacement selection concerning the evolution of the population from a generation to the next. A standard single point crossover and mutation is then applied to the generated offsprings to introduce some diversity in the population. The mutuant gene is selected randomly at each generation by sampling a uniform random number. In addition to the typical genetic operators, also a local search is incorporated in order to speed up the convergence of the algorithm.

3.3 Distribution Networks

Syarif et al. (2002) consider a single period capacitated distribution problem which also consists of the binary decisions of opening plants and Distribution Center (DC).

The number of the DC's to be opened is assumed to be given as a constant. The objective is to minimize the total transportation, inventory, ordering and warehouse costs under capacity constraints, while fully supplying deterministic customer demand. A spanning tree based GA is used to solve the model. Vertex encoding is used with Prüfer number representation which establishes a unique sequence of length $n-2$ associated with the tree with n vertices. The chromosome consists of five sub-strings as illustrated in Figure 4. The first and the second substrings are binary digits representing opened/closed plants and DCs, respectively. The last three numbers are the Prüfer numbers to represent the production and distribution pattern for each echelon.

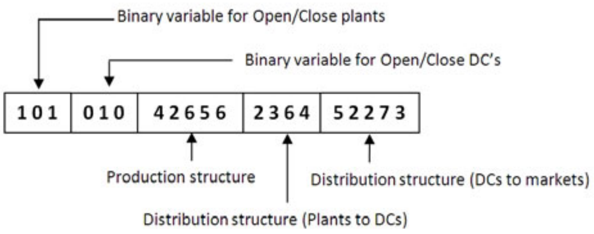

Binary variable for Open/Close plants
Binary variable for Open/Close DC's
101
010
42656
2364
52273
Production structure
Distribution structure (DCs to markets)
Distribution structure (Plants to DCs)

Fig. 4 Chromosome Structure Used by Syarif et al. (2002).

The infeasibility that may result from capacities and distribution structures are eliminated with a repair strategy, simply by replacing the digits in Prüfer numbers until the number of connections in the supplier set is equal to the number of connections in the supplier plants set for each node. A single point crossover operation is used. For mutation, an inversion-displacement operation is employed. Inversion selects two positions within a chromosome at random and inverts the substring between these two positions where displacement selects a substring at random and inserts it in a random position. The results of various experiments show that proposed algorithm provides near optimal solutions both for small and large scaled problems.

Yokoyama (2002) present a model and a solution procedure based on GA for single item distribution system with stationary and probabilistic demand. The objective is to determine the target order-up-to levels for DCs and transportation quantities to minimize the expected total inventory related costs and transportation costs. Simulation and linear programming are used for calculating the estimates of expected costs. Each solution is represented by a integer array of size equal to the number of the DC where each gene holds the order-up-to level for the given DC. Total expected cost for a given chromosome is estimated by simulation where optimal transportation quantities for given inventory levels are determined by linear programming. Two-point crossover operation, roulette wheel selection, and random replacement mutation operations are implemented for generating offsprings. The results of the algorithm are compared to the results of random local search algorithm. GA is observed to produce slightly better results than random search within same computation times.

Han and Damrongwongsiri (2005) develop a model to define stochastic, multi-period, two-echelon inventory with the many-to-many demand-supplier network problem to develop a (R,S) inventory management system where R refers to the replenishment period and S is the order-up-to level. GA is applied to derive optimal solutions through a two stage optimization problem. First stage covers the optimization of inventory order-up-to levels of the warehouses based on historical demand. Binary representation of the order levels is used for encoding where the length of the bitstream assigned to each warehouse is determined by the capacity of the warehouse. The fitness function is calculated as the sum of inventory carrying and shortage costs. The optimal inventory order-up-to levels determined in the first stage are used as inputs of the second stage, distribution planning. The goal of this stage is to determine the optimal transportation quantities of the retailers that minimizes inventory related costs. Binary encoding is preferred for chromosome representation, where the bit length assigned for each retailer is determined by both the capacity of the retailer and the total warehouses' maximum inventory level. First population is randomly generated and roulette wheel approach is used for selection. Crossover is done by one-cut-point method and random point mutation is used. Results of numerical experiments do not contain any performance comparisons with other methods but various experiment sets are presented to illustrate the flexibility of the method to handle many uncertainty factors.

The approach developed by Han and Damrongwongsiri (2005) is implemented to solve a real industry case by Wang and Wang (2008). Both the mathematical model and the GA structure are adopted for optimizing the distribution operations of a medical products manufacturer which supplies to four Nordic region markets from three geographically distinct warehouses. The demands at each market are assumed to be normally distributed with parameters forecasted through past data and independent of each other. Again no data is provided for performance comparisons but it's noted that GAs are able to compute the trade-off of all parameters and derive a good inventory and distribution plan, which might lead to a reduction up to 80% on the total cost of the system.

4 A GA Approach for Stochastic Lot-Sizing in a Centralized Distribution Network

This section presents a novel GA structure and the implementation issues for solution of a stochastic lot sizing problem in a centralized distribution system. To author's knowledge, this is the first study that investigates the use of GA for the capacitated lot sizing problem of One Warehouse - Multi Retailer (OWMR) system under stochastic and time varying demand. A similar study is given by Celebi and Bayraktar (2008) for deterministic demand case. The main contribution of the proposed GA structure is the domain specific encoding scheme and the fitness value calculation technique that uses dynamic programming for evaluating best order structure of the warehouse. This structure can be utilized as a collaborative supply chain planning tool to effectively manage the distribution process.

This section also presents an extensive numerical study over simulations which identifies the parameter settings where the proposed GA based method performs better or poorer than a widely used approximation technique. These results also implicitly shows the impact of cost parameters on the performance of the studied distribution network.

4.1 Model Description

A two-echelon distribution inventory system with a single central warehouse and multiple retailers is considered. The network is controlled by a central distributor which is occupied with all relevant information. That means, the distributor monitors the end customer demand and retailers' inventory levels in order to decide on order quantities, shipping and timing of replenishment orders.

Retailers directly replenish their stocks from the warehouse where warehouse orders from an outside supplier. It is assumed that the outside supplier have infinite source of supply or work at very high service levels so delays from the supplier side are negligible. All facilities follow a periodic inventory order policy where the lengths of planning periods are the same for all retailers and the warehouse. Customer demands are probabilistic and only placed in retail locations. It is assumed that the demand rates might change from one period to another, but remain constant within a period. This is not a restrictive assumption when the period length is kept small enough compared to the planning horizon. Moreover, this assumption is a good representative of the practical situation when demand quantities are forecasted by a time series method. In such a case there exists a demand forecast for given period, and the variations of demand within the period are estimated by a probability distribution.

An inventory problem for T periods is considered. There is a fixed ordering cost incurred with each replenishment with a cost function $\delta(q)$;

$$\delta(q_t^n) = \begin{cases} cq_t^n + K_t^n, & \text{if } q_t^n > 0 \\ 0, & \text{if } q_t^n = 0, \end{cases} \tag{1}$$

where K_t^n is the setup cost, c is the unit purchasing cost, and q_t^n is the replenishment quantity at period t. During each period, the stock on hand is decreased by an amount equal to the demand.

In addition to ordering cost, for all locations, carrying inventories incurs holding costs at a rate of h which is charged on the inventory level at the end of each period. Unfilled customer demands are fully back-ordered at retailer level and the retailer shortages are penalized a rate of π, the back-order cost per unit.

$$L(x_t^n) = \begin{cases} h\sum_{u=0}^{x_t^n}(x_t^n - u)P(u) + \pi\sum_{u=x_t^n}^{\infty}(u - x_t^n)P(u), & \text{if } x_t^n > 0 \\ \pi\sum_{u=0}^{\infty}(u - x_t^n)P(u), & \text{if } x_t^n \leq 0. \end{cases} \tag{2}$$

In equation (2), x_t^n is used to express the inventory level and $P(u)$ is the probability of observing u units of demand.

The allocation decision made for shipping to a total number of N retailers has direct impact on warehouse's costs. At the beginning of each period, warehouse allocates a total of $\sum_N q_t^n$ units to the retailers. Delivery of a replenishment arrives ℓ periods after the allocation decision, when the stock on hand is increased by the amount of the replenishment. Since customer transactions only occur at retailer points and the warehouse directly replenishes from an infinite supply source, there aren't any costs associated with inventory shortages on warehouse side. Inventory carrying costs for the warehouse for holding y units of inventory, is denoted by $H(y, q_t)$ and given as:

$$H(y, q_t) = h_0(y - \sum_N q_t^n) \tag{3}$$

In a centralized system the optimal policy is not necessarily the aggregation of individual optimal policies because of the dependencies among members and costs associated with those dependencies. The purpose of our model is to obtain the minimum total cost for the overall system which is formulated as follows:

$$\text{Minimize} \quad \sum_{t=1}^{T} \Big(H(y_t, q_t) + \delta(p_t) + \sum_{n=1}^{N} \big(L(x_t^n) + \delta(q_t^n) \big) \Big). \tag{4}$$

subject to

$$x_{t+1}^n = x_t^n - D_t^n + q_{t-\ell}^n \qquad n = 1, \ldots, N \tag{5}$$

$$\sum_{n=1}^{N} q_t^n \leq y_t \qquad t = 1, \ldots, T \tag{6}$$

$$x_t^n \leq C_n \qquad t = 1, \ldots, T, n = 1, \ldots, N \tag{7}$$

$$y_t \leq C_0 \qquad t = 1, \ldots, T \tag{8}$$

The solution of model presented above for the lot sizing problem of the centralized system gives the minimum value of objective function (4) under given constraints. The first two terms in the objective function refer to total expected warehouse inventory costs. Last two terms are the sum of expected inventory related costs of all retailers. First constraint is a balance equation which adjusts the inventory levels between two consecutive periods. This is not a simple linear equation due to the stochastic variable D_t^n. Second constraint limits the number of shipped products from warehouse to all retailers with warehouse's on-hand stock in period t. Constraints (7) and (8) ensures that retailers' and warehouse's inventory holding capacities are not exceeded.

Due to the stock allocation problem of distribution systems, the optimality formulations are functions of distributor's and N retailers' inventory levels and can not be broken down in the form of independent formulations (Clark and Scarf, 1960). The state of the system at the beginning of period t can be described by the vector $(x_t, q_{t-\ell_n+1}, \ldots, q_{t-1})$. For any period $t \in 1 \ldots T$, the minimum total expected cost function is defined as $f_t(x_t, q_{t-\ell+1}, \ldots, q_{t-1}, y_t, p_{t-\ell_0+1}, \ldots, p_{t-1})$. Here x_t and q_t refers to the vector of the inventory levels and replenishment quantities of all retailers, y_t and p_t refers to the inventory level and replenishment quantity of the warehouse for period t. The optimal lot sizing policy in centralized system is given by one combined formulation due to the dependencies between retailers' and distributor's orders. The recursive formulation for period t then becomes:

$$\begin{aligned} & f_t(x_t, q_{t-\ell}, \ldots, q_{t-1}, y_t, p_{t-\ell_0}, \ldots, p_{t-1}) = \qquad (9) \\ & \min_{0 \le y_t \le C_0} \Big\{ \min_{0 \le \sum q_t \le min(y_t, C_n)} \big\{ \delta(q_t) + \delta(p_t) + H(y_t, p_t) + L(x_t) \\ & + f_{t+1}(x_t + q_{t-\ell} - u, q_{t-\ell+1}, \ldots, q_t, y_t + p_{t-\ell_0} - \sum q_t, p_{t-\ell_0+1}, \ldots, p_t) P(u) \big\} \Big\}. \end{aligned}$$

Solving (9) recursively for T periods gives us optimal policy for allocation and inventory replenishment strategy of the overall system.

4.2 Motivation for Using Genetic Algorithms

For finding the global optimum convexity plays a crucial role in minimization problems. Finding a local optimum solution is an important step of solving the global problem, however it is not sufficient most of the time. Traditional optimization technique works by obtaining the zeros of a function's derivative, and testing for optimality. Such derivative tests obtain local information, and hence yield solutions that are locally optimal. Removing the convexity assumption on the function to be optimized, this method may prove severely inefficient, as it cannot provide anything more than local information. In such a case, the characteristics of the solution space should be investigated for ensuring the global optimum.

The problem presented in section 4.1 is proven to be NP-complete (Florian et al., 1980) and there is no known method to decompose the model into smaller ones. Besides the non convex behavior of total cost function, (4), distribution network lot sizing problem consists of variables that are diverse in their behavior, boundaries, and the probability distribution type. Moreover, the individual objective functions of the parties are not linear and multiple objectives can not be combined into a single metric. The combinatorial and sequential behavior of the two-echelon lot sizing problem can not be easily handled by traditional optimization techniques. Then it is reasonable to investigate a search algorithm to approximate the global minimum of the inventory distribution problem.

The known methods developed so far need considerable computational effort to obtain an optimal solution and so are only able to solve relatively small problems within a reasonable time. The problem requires a considerable computational burden when the problem instance is large. Even for a simple example which represents a very small instance of a problem, the total cost function might have multiple local minima (Celebi, 2008). A search algorithm that only uses the gradient ascent will be trapped in a local optimum, but any search strategy that analyzes a wider area will be able to cross the local optimum and achieve better results.

GAs are capable of handling non-linear functions and can also deal with multiple objectives. They do not have any restrictions on the nature of data or mathematical requirements about the problem structure, unlike most traditional approaches. Due to their evolutionary nature, they can handle any kind of objective functions and constraints (linear or nonlinear) defined on discrete or continuous, or mixed search spaces (Gen and Cheng, 1997).

One of the difficulties when dealing with non convex optimization problems by search algorithms is that one often falls into local optima. When this happens, often the global optimum is then impossible to reach. The probabilistic evolution of operators makes GA very effective at performing global search and reaching global optima. The GA based solution methods have the advantage of being able to generate both convex and non convex points of the optimization curve, accommodate nonlinearities in the objective functions, and not be restricted by the peculiarities of a weighted objective functions (Scott et al., 1995).

One should keep in mind that as it is common with all heuristic methods, GAs cannot guarantee to locate the global optimum in a problem space in a finite time. But still, for some engineering problems such as many design and simulation tasks, the most desirable solution may not be the conventional global optimum but instead a solution representing a robust answer to the problem in hand is sought.

4.3 *The Proposed GA Structure*

The construction of a GA for any problem can be separated into four distinct and yet related tasks (Hou et al., 1994):

1. The choice of the representation of the solutions,
2. The determination of the fitness function,
3. The design of the genetic operators to be used for creation of new generations, and
4. The determination of the probabilities controlling the genetic operators.

Each of the above four components greatly affects the solution obtained as well as the performance of the GA. The summary of these steps involved in the proposed GA structure are described in detail in the following sections.

4.3.1 Encoding and Initialization

The most critical problem in applying a GA is in finding a suitable encoding of the examples in the problem domain to a chromosome. A good choice of representation will make the search easy by limiting the search space, a poor choice will result in a large search space. Our candidate solutions are combinations of all possible order quantities of each retailer and the distributor, for a number of T periods, hence the phenotype space P is the set of all such combinations. To design a GA defined by a representation of phenotypes from P, integer value representation is used where each chromosome represents retailers order up to levels for each period. Each chromosome takes the following sequence in the proposed encoding scheme:

$$\mathscr{C} = q_1^1 q_2^1 \dots q_T^1 q_1^2 \dots q_T^2 \dots q_1^N \dots q_T^N$$

Here, q_t^n is the direct value representation of the replenishment quantity for retailer n in period t. Each chromosome is a string of $N \times T$ genes, where N is the total number of retailers and T is the number of periods. That means, i^{th} gene in the sequence is the replenishment quantity for retailer $\lceil i/T \rceil$ in period $i(\mathrm{mod}\ T)$. Each gene can take values between 0 and C_n which corresponds to the inventory carrying capacity of retailer n.

This design guarantees the completeness and the correctness requirements of encoding. Completeness is simply a consequence of using allocation quantities for *all retailers* in *all periods*. The correctness condition is provided by a simple check before fitness calculation. If the inventory on hand exceeds the capacity level for the given allocation quantity in any period, allocation quantity is updated to provide a feasible inventory level. Hence, feasibility of the chromosomes are kept in the legal domain without use of any additional constraint. First population is created with randomly generated individuals.

4.3.2 Fitness Evaluation and Selection

The role of the fitness function is to represent the requirements for improvement (Eiben and Smith, 2007) for a given individual. The quality of the given solution f, represented by a chromosome $\mathscr{C}$, is determined by the minimum expected total cost of the system given by equation (4). The total cost represented by any chromosome is evaluated in three steps:

1. Each retailer's expected costs for the replenishment scheme proposed by the chromosome are calculated by using equation (10) for total of T periods.

$$g_t^n(x_t^n, q_{t-\ell}^n, \dots, q_{t-1}^n) = \big\{\delta q_t^n + L(x_t^n) + \sum_{u=0}^{\infty} g_{t+1}(x_t^n + q_{t-\ell}^n - u, q_{t-\ell+1}^n, \dots, q_t^n)P(u)\big\}. \quad (10)$$

2. The replenishment quantities are summed up to develop the aggregate allocation quantities ($\sum q_t$) and the ordering policy of the distributor is evaluated by dynamic programming formulation as given in equation (11).

$$f_t(y_t, p_{t-\ell_0}, \dots, p_{t-1}) = \min_{p_t \geq 0} \big\{\delta p_t + H(y_t, \sum q_t) + f_{t+1}(y_t + p_{t-\ell_0} - \sum q_t, p_{t-\ell_0+1}, \dots, p_t)\big\}. \quad (11)$$

3. The objective function value is taken as the sum of the cost of warehouse (Eqn. (10)) and total cost of retailers (Eqn. (11)).

Resulting raw fitness scores are converted to values in a range that is suitable for the selection function. To avoid the effect of the spread of the raw scores, *Rank* fitness scaling method is used which scales the raw scores based on the rank of each individual instead of its score (Gen and Cheng, 1997). The selection of mating parents is done through *roulette wheel algorithm*. The pseudocode of this algorithm can be found in (Eiben and Smith, 2007).

4.3.3 Creation of New Generations

At each iteration, the current population is used to create the offsprings that make up the next generation through a *general replacement scheme*, so that, the chromosomes in the current population are completely replaced by the offspring. That means, population size is kept constant in its initial level through generations. The creation of next generation is conducted by three types of children. Figure 5 presents the schematic illustration of the three types of children.

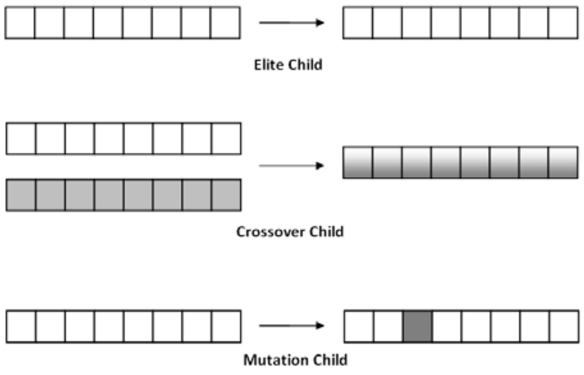


Fig. 5 Schematic Illustration of Three Types of Children.

Elite children are the individuals in the current generation with the best fitness values. These individuals are automatically passed to the next generation without any modification. Such an elitist algorithm is recorded to be able to speed up the performance of the GA significantly by preventing loss of good solutions once they are found (Zitzler et al., 1998). Experimenting on varying numbers (from 0 to 10) of elite chromosomes, the number of elite chromosomes is set to 2 which has given highest scoring individuals and provided best results in means of fitness value and time of convergence.

Crossover children are created by paring up the chromosomes and combining the vectors of a pair of parents. *Intermediate recombination*, which creates a new value for each gene of the offspring that lies between those parents. The function creates the child, c from parent$_1$, and parent$_2$ using the following formula:

$$\text{offspring} = \alpha\text{parent}_1 + (1-\alpha)\text{parent}_2,$$

where α is a random number generated from the range [0,1].

This method protects the feasibility of the chromosomes but might assign fractional numbers to the offspring genes. Such genes are repaired by rounding the number to the nearest integer for representing a valid order quantity.

Mutation children are created by introducing random changes, or mutations, to a single parent. To introduce variations into the chromosomes, *Random Resetting* in multiple points is implemented. Genes are selected according to a probability of being mutated, P_{mut}, which is defined by the *mutation rate*. Selected genes are then replaced with a value which is a realization of uniformly distributed random variable within the capacity range.

4.3.4 Setting Operational Parameters for GA Cycles

The selection of the best genetic parameters such as population size, number of generations, probability of crossover, and probability of mutation, is one of the important issues for the successful application of GA. Identifying the best parameters for a specific task is an open and challenging problem. Larger population sizes reduce the chance that GA will return a local minimum by searching the solution space more thoroughly but it also causes the algorithm to run more slowly. Experimenting on different population sizes, for the given problem instance it is observed that a population size of 50 gives satisfactory results both in sense of convergence speed and fitness values for our problem.

Two genetic operators, crossover and mutation, competes over the field of convergence. High crossover rate decreases the level of variation in the population so forces the convergence, while mutation forces diversity in the population. As a result of this fact, an optimum setting for the operator probabilities should have been determined. Optimal rates of these operators are problem specific and there are no defined rules on selecting the best GA operator fractions. To overcome this, the crossover and mutation rates are determined through several GA experiments for different rates of crossover and mutation by linear variations as suggested by Davis (1991). The experiment sets are composed by all combinations of 11 different crossover fractions over $[0.5, 1.0]$ and 11 different mutation rates over $[0, 0.2]$. The performance of each configuration is calculated by the value of the objective function, which is total system cost.

First one problem instance with 10 periods which represents the maximum period is used in our test cases. GA is run $121(11 \times 11)$ times to observe each configuration of crossover and mutation rate combinations. Two terminating conditions are set: First on the maximum number of generations, as 500, and second, on the number of generations without any improvement on fitness function, as 100. The minimum, average fitness function values of these three runs are recorded and crossover and mutation parameters are set to ones which give the minimum fitness values. The experiment results are presented in Figure 6.

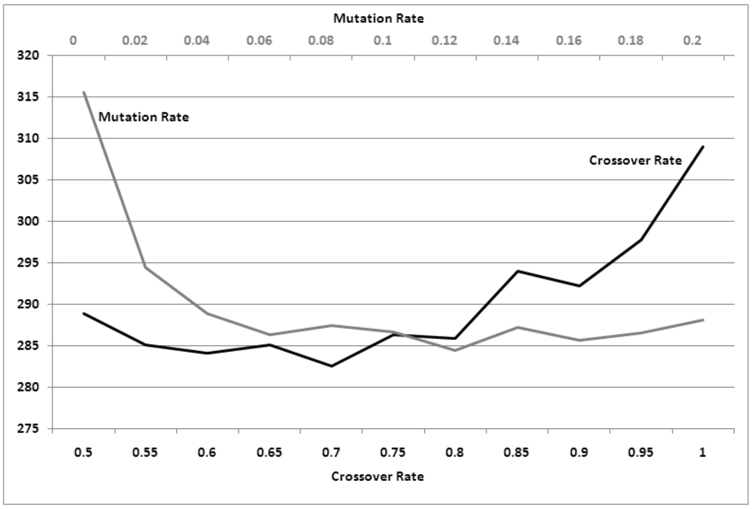


Fig. 6 Determination of Best Crossover and Mutation Rates.

Experiment results show that one of the crossover or mutation rates perform superior than others. Generally, high crossover rates are observed to give better results when mutation rate is also increased, however best results are obtained by a moderate crossover fraction, 0.7. On the average, mutation rate of 0.12 is observed to perform slightly better than others.

4.4 Numerical Study and Discussions

To demonstrate the performance of proposed GA approach, several test cases with different operational parameters are experimented. All algorithms are implemented in Matlab due to its efficiency for numerical computations, advanced data analysis capabilities, visualisation tools, and special purpose application domain toolboxes. The built-in functions of population creation, crossover, mutation, and fitness evaluation of Matlab-GA toolbox are modified according to the structure of the proposed GA design. Cost function and dynamic programming algorithms are coded as a common set of interdependent functions which are both used by GA algorithm and Balance Assumption (BA).

Since the optimal policy and the associated cost are unknown, instead of comparing the cost obtained by GA to the optimal cost, the cost of the system evaluated under BA and the cost realizations of system simulations(sim) are taken as the benchmark values. BA implies that in each period the downstream stock levels are balanced in such a way that a cost minimizing allocation without restrictions on the allocation variables will never result in negative allocation quantities. Thus, the system-wide cost calculated analytically under BA provides a lower bound for the true optimal cost. Detailed explanation of the balance assumption may be found in Eppen and Schrage (1981).

Assumption of zero lateral transshipment among retailers leads to a relaxation of the original optimization problem and is infeasible in real life. To provide a realistic benchmark value, an estimate for the real cost of the given policy (by BA) is obtained by simulation, so the cost of a feasible policy can be achieved. Each simulation is run for at least 100 times and terminated as soon as the width of a 99% confidence interval about the average cost function was within 1% of the average cost. The relative gaps between the results of the GA runs and the lower bound and the simulation runs ($\%\varepsilon_{lb} = 100 \times \frac{GA-LB}{LB}$ and $\%\varepsilon_{sim} = 100 \times \frac{SIM-GEN}{LB}$ respectively) are used as measures to assess the performance of the proposed GA method. Since the optimal cost of the original problem is between LB and GA, a small relative gap (ε_{lb}) implies that GA value is close to the optimal cost of the original problem, meaning that GA leads to an accurate approximation of the true optimal cost. On the other hand, even though the ”balance assumption” might seem somewhat unrealistic, it has since been used extensively in the inventory literature and has been shown to produce solutions of very good quality in many different situations, (see for example Eppen and Schrage (1981); Federgruen and Zipkin (1984); van Houtum et al. (1996)). Policies that can provide considerable improvements over the BA in less or equal computation times, might be considered as well performing. Hence, a large relative gap between the simulation of ”balance” policy and GA policy ($\%\varepsilon_{sim}$) is an indicator of the success of proposed GA structure for solving the given inventory distribution problem.

Due to the curse of dimensionality, only the case with two retailers with demands distributed over integers in $[0,4]$ with probabilities $[0.1, 0.2, 0.4, 0.3]$ is considered. This approximately corresponds to a moderate level of coefficient of variation. Both lead times for the retailers and the distributor are taken as 1. A limited number of test cases are structured by varying following cost parameters:

Fixed Costs: A variety of cases is considered for fixed replenishment costs defined by three different values for the retailers, $(K_n = 0, 5, 10)$ and three for the distributor:$(K_0 = 20, 10, 0)$).

Inventory Carrying Costs: Inventory carrying cost of retailers is taken constant, $h_n = 1$, and the variation is provided by changing the added value of the distributor: $(h_0 = 0.1, 0.5, 0.9)$.

Shortage Costs: The values of shortage costs are chosen as 4, 9, 19 and 99 which approximately correspond to no-stockout probabilities of 80%, 90%, 95% and 99% respectively.

A full factorial design is used to generate experimental cases that corresponds to 108 problem instances. All test cases are set for 10 periods.

First the lower bound and relevant simulation values for each test case are calculated. Then GA is run three times for each test and the run is stopped when either a pre-specified number of searches reaches to 1000, or there is no improvement in the best fitness value for 100 generations. The performance of each configuration is calculated by the value of the objective function, which is the expected total system cost. The best of these three runs which provides the minimum cost is used. The relative gap measures ($\%\varepsilon_{lb}$ and $\%\varepsilon_{sim}$) for scenarios $1-108$ are graphically depicted in Figure 7.

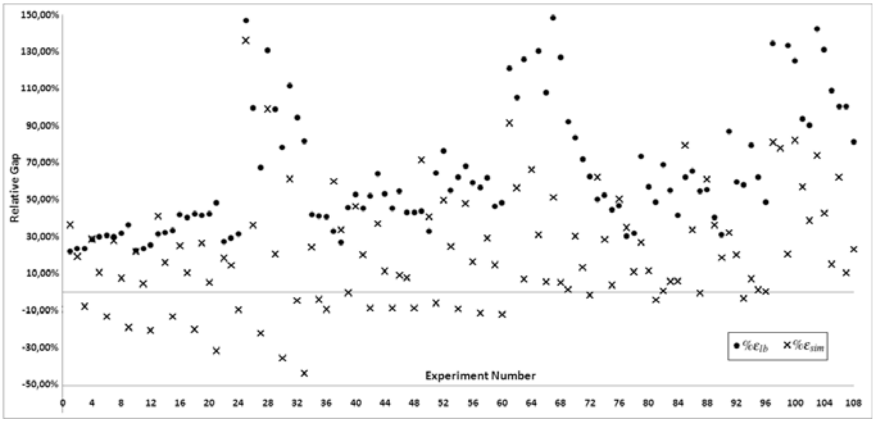


Fig. 7 Experiment Results.

When these 108 test instances are ranked with respect to $\%\varepsilon_{lb}$, 61 of them have $\%\varepsilon_{lb} > 50\%$. Among these cases only 11 of them have a $\%\varepsilon_{sim} < 0$, that means for remaining 50 cases, balance assumption fails to provide a good lower bound and an effective replenishment policy. In general, GA produced better results than balance assumption in 82 out of 108 cases with an $\%\varepsilon_{sim} > 0$. Savings over the total cost that are achieved by use of GA is 21.82% on the average, while savings up to 136.23% are recorded.

In order to see the influence of the cost parameters on the results, test results are also given in Tables 2 and 3, where data is summarized with respect to one parameter at a time. For example, the left part of Table 2 is dedicated to display the effect of the fixed replenishment costs of retailers. The first column gives the values of various measures for a set of 36 test instances in which $K_n = 0$. The measures used in the analysis are minimum, maximum and average $\%\varepsilon_{lb}$ (denoted by $\%\varepsilon_{lb}^{-}$, $\%\varepsilon_{lb}^{+}$ and $\%\bar{\varepsilon}_{lb}$, respectively), the minimum, maximum and percentage of improvement (denoted by $\%\varepsilon_{sim}^{-}$, $\%\varepsilon_{sim}^{+}$ and $\%\bar{\varepsilon}_{sim}$, respectively), and the number of the cases that GA produced better and worse results than the simulation of balance assumption (denoted by $\oplus$ and $\ominus$, respectively).

The findings can be summarized as follows:

1. In the test bed of 36 problem instances with $K_n = 0$, there are 14 scenarios with $\%\varepsilon_{sim} < 0$. This number decreases with increasing value of K_n. Similarly, the average improvement is 13.31% when retailer fixed replenishment costs are zero, and the improvement increases up to 30.79% with increasing value of K_n. This is in line with expectations. Balance assumption implies zero transshipment costs

Table 2 The summary of the results - Fixed Replenishment Costs

	Retailers' Fixed Costs			Distributor's Fixed Costs		
	$K_n = 0$	$K_n = 5$	$K_n = 10$	$K_0 = 0$	$K_0 = 10$	$K_0 = 20$
$\%\varepsilon_{lb}^{-}$	22.19	26.98	30.37	40.69	27.22	22.19
$\%\varepsilon_{lb}^{+}$	147.06	173.80	165.65	173.80	86.96	73.57
$\%\bar{\varepsilon}_{lb}$	48.83	71.81	75.37	105.35	50.56	41.32
$\%\varepsilon_{sim}^{-}$	−22.98	−12.13	−4.13	−44.08	−31.86	−20.79
$\%\varepsilon_{sim}^{+}$	136.23	91.69	82.28	136.23	79.45	62.23
$\%\bar{\varepsilon}_{sim}$	13.31	22.59	30.79	33.34	17.35	14.78
$\oplus$	22	27	33	29	26	27
$\ominus$	14	9	3	7	10	9

Table 3 The summary of the results - Inventory Carrying and Penalty Costs

	Inventory Carrying Costs			Penalty Costs			
	$h_0 = 0.9$	$h_0 = 0.5$	$h_0 = 0.1$	$\pi = 4$	$\pi = 9$	$\pi = 19$	$\pi = 99$
$\%\varepsilon_{lb}^{-}$	22.19	23.40	23.68	22.19	28.22	29.85	22.49
$\%\varepsilon_{lb}^{+}$	173.80	165.65	133.42	165.65	173.80	148.52	100.47
$\%\bar{\varepsilon}_{lb}$	74.57	63.82	58.84	68.26	67.50	74.02	53.19
$\%\varepsilon_{sim}^{-}$	0.58	−4.73	−44.08	−22.19	−35.84	−44.08	−20.79
$\%\varepsilon_{sim}^{+}$	136.23	78.01	38.72	136.23	99.09	74.05	62.17
$\%\bar{\varepsilon}_{sim}$	48.80	21.42	−4.76	36.43	27.44	14.60	8.82
$\oplus$	36	34	12	21	22	19	20
$\ominus$	0	2	24	6	5	8	7

between retailers thus increasing transshipment costs decreases the effectiveness of the method. That means when retailer replenishment costs are high, GA may provide better solutions than the policies based on balance assumption.

2. Similarly, when the distributor's replenishment costs are high, GA tend to perform better on the basis of comparisons to LB. This is not parallel to comparisons to SIM. Though increasing K_0 doesn't have a visible impact on the number of cases that GA performs better, the level of average improvement decreases dramatically.
3. Highest impact on performance of GA is observed on added value of the transportation to retailers. The performance of GA is top when the costs of carrying inventories in the warehouse is high. When $h_0 = 0.9$, GA never performed poorer than the policy based on balance assumption even though there is not significant change on the value of $\%\bar{\varepsilon}_{lb}$. The average improvement on the cost value is almost 50% where it reaches up to 136.23%. On the other hand, GA perform poorest when the carrying costs of inventories in the warehouse is comparably lower than carrying costs of retailers. When $h_0 = 0.1$, policy based on balance assumption produces better results than GA in 67% of the cases.

4. The value of the penalty costs has the lowest impact of GA performance. No trend can be observed on the number of the cases GA performs better and the value of the $\%\bar{\varepsilon}_{lb}$ with increasing value of the penalty costs, but the average rate of improvement $\%\bar{\varepsilon}_{sim}$ decreases visibly with increasing value of the penalty costs. The reason is not that GA performs poorer in such cases but balance assumption based policy performs better. For example, when π increases from 4 to 99, the average improvement of GA drops from 36.4% to 8.82% but the average gap between the GA and the lower bound, $\%\bar{\varepsilon}_{lb}$, also decreases from 68.26% to 53.19%.

The experiment results show that GA generally produce better results than the policy based on balance assumption. However in some cases, the solution provided by GA give higher total system costs than the benchmark. This shows that as in common with all heuristic methods, GAs cannot guarantee to locate the global optimum in a problem space in a bounded time. This is mainly through stochastic search behavior of the GAs. The results can be improved by increasing the number of trials and the computation time of the algorithm, and also experimenting on different GA operators such as population size, mutation and crossover. Besides, in practice the most desirable solution may not be the conventional global optimum but instead a solution representing a robust answer to the problem in hand is sought. Hence, for a large system with a high number of periods and retailers, GAs can be used as an effective algorithm for solving the multi-echelon inventory distribution problem under stochastic demand.

This study only presents the comparison of the proposed GA method with most known and used heuristic, for a system under a limited number of parameters. The experiment results can be strengthened by a more comprehensive numerical study, specifically targeting to assess the performance of the proposed heuristic. Another extension can be comparison of these results with the results obtained by other heuristic solutions, as well as the real optimal solution of the system.

5 Conclusion

GAs have been applied to a wide variety of multi-echelon inventory control problems in various studies. Tests on artificial data sets show that GA are pretty successful for determining a good solution even for the most complicated problems. However, some barriers might exist for the successful implementation of the proposed methods to real life. The complexity of today's business world means that it is often not possible to link external sources of information into the vendor's production and inventory control processes (Stank et al., 2001), as in many cases the same level of detailed information cannot be obtained from all of the distribution channels. For some environments, centralization may be expensive, very complex, or the coordination may be too much of a burden. This is especially true for large systems, which would require substantial computational power to store and process large amounts of information for centralized decision-making. A practical contribution can be made if GAs are applied to industrial inventory problems as integrated with interactive decision support systems where application data and test results on the algorithms performance are collected from real life applications.

Main assumption of multi-stage inventory control is the share of information among the supply chain members, however in practice, this assumption might be restrictive. The companies involved in strategic or incidental supply-network partnerships might be not willing or prepared to share information needed required for coordination of supply chain. Hence, the participants might agree to share only partial information due to the unwillingness of the participants to share private information such as cost structure. A distributed algorithm based on evolutionary algorithms that allows the distributed system to perform just as well as a centralized one may be designed for such cases. For example, Shin (2007) propose a framework for such a collaborative coordination mechanism: The coordinator solves the aggregate problem and delivers the solution results to all members. Each member evaluates the performance of the delivered solution from the coordinator using its own cost structure, solves its own problem in terms of its own objectives and measures the performance, calculates its penalty, and returns the penalty with its solution to the coordinator. Then, the coordinator selects the facility with the largest penalty value, modifies and solves the problem again, and redistributes the solution results to the all members. Not only the local optimization procedures but the collaboration mechanism can easily be optimized by GA to provide a global optimal solution.

For most of the multi-echelon systems, uncertainty is an unavoidable factor of inventory control and recognized to have a major impact on the manufacturing and service functions (Wilding, 1998). Uncertainties such as high variability in demand, manufacturing processes or supply create problems in planning, scheduling and control that jeopardize delivery performance (Fisher et al., 1997). Incorporating uncertainty might pose severe problems for the current GA structures developed for multi-stage inventory control. Explicitly, incorporating uncertainty will undoubtedly result in very complex models. However, the power of GAs to deal with such complex models proposes a promising topic of investigation and new research opportunities.

Nomenclature

ℓ Transportation lead time, page 13
π Unit Shortage Cost, page 13
c Unit purchasing cost, page 12
C_0 Maximum replenishment quantity for the distributor, page 14
C_n Inventory holding capacity of retailer n, page 14
D_t^n Demand faced by retailer n in period t, page 13
h Unit Inventory Holding Cost, page 12
K_t^n Fixed Cost per Order of retailer n at period t, page 12
$L(x)$ One period expected inventory carrying and shortage penalty costs, page 13
N The number of retailers, page 13
$P(u)$ Probability of observing u units of demand, page 13
P_{mut} Mutation Rate, page 18
q_t^n Replenishment quantity of retailer n at period t, page 12
T Number of periods in planning horizon, page 12
x_t^n Inventory level of retailer n in the beginning of period t, page 13

Acronyms

BA Balance Assumption
DC Distribution Center
GA Genetic Algorithm
HGA Hybrid Genetic Algorithm
OWMR One Warehouse - Multi Retailer
WW WagnerWhitin

References

Axsater, S.: Serial and distribution inventory systems. In: Kok, Graves (eds.) Handbooks in Operations Research and Management Science:V11 Supply Chain Management: Design, Coordination and Operation, pp. 525–559. Elsevier Ltd., Oxford (2003)

Ballou, R.H.: Business logistics management: planning, organizing, and controlling the supply chain. Prentice-Hall, Englewood Cliffs (1999) ISBN 0137956592

Berretta, R., Rodrigues, L.F.: A memetic algorithm for a multistage capacitated lot-sizing problem. International Journal of Production Economics 87, 67–81 (2004)

Cachon, G.P., Fisher, M.: Supply chain inventory management and the value of shared information. Management Science 46(8), 1032–1048 (2000)

Çelebi, D.: Stochastic Lot Sizing in a Centralized Distribution Network. PhD thesis, Istanbul Teknik Üniversitesi (2008)

Celebi, D., Bayraktar, D.: A genetic algorithm for multi location inventory problem. In: Proceedings of Fifteenth International Working Seminar on Production Economics (2008)

Chen, F.: Optimal policies for multi-echelon inventory problems with batch ordering. Operations Research 48(3), 375–389 (2000)

Chen, F.: Information sharing and supply chain coordination. In: Kok, Graves (eds.) Handbooks in Operations Research and Management Science:V11 Supply Chain Management: Design, Coordination and Operation, pp. 341–451. Elsevier Ltd., Oxford (2003)

Clark, A.J., Scarf, H.: Optimal policies for a multi-echelon inventory problem. Management Science 6(4), 475 (1960)

Daniel, J.S.R., Rajendran, C.: A simulation-based genetic algorithm for invertory optimization in a serial supply chain. International Transactions in Operational Research 12, 101–127 (2005)

Davis, L.: Handbook of Genetic Algorithms. van Nostrand Reinhold, New York (1991) ISBN 0-442-00173-8

Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series. Springer, Heidelberg (2007) ISBN 978354040184-1

Eppen, G., Schrage, L.: Centralized ordering policies in a multi-warehouse system with leadtimes and random demand. In: Schwarz, L. (ed.) Multi-Level Production/Inventory Control Systems: Theory and Practice, pp. 51–69, North Holland, Amsterdam (1981)

Fakhrzad, M.B., Khademi Zare, H.: Combination of genetic algorithm with lagrange multipliers for lot-size determination in multi-stage production scheduling problems. Expert Systems with Applications 36, 10180–10187 (2009)

Federgruen, A.: Centralized planning models. In: Graves, Kan, R., Zipkin, P. (eds.) Handbooks in Operations Research and Management Science:V4 Logistics of Production and Inventory, pp. 133–173. Elsevier Ltd., Oxford (1993)

Federgruen, A., Zipkin, P.: Approximations of dynamic, multilocation production and inventory problems. Management Science 30(1), 69 (1984)

Fisher, M.L., Hammond, J., Obermeyer, W., Raman, A.: Configuring a supply chain to reduce the cost of demand uncertainty. Production and Operations Management 6(3), 211–225 (1997)

Florian, M., Lenstra, J.K., Rinnooy Kan, A.H.G.: Deterministic production planning: Algorithms and complexity. Management Science 26, 669–679 (1980)

Gen, M., Cheng, R.: Genetic Algorithms and Engineering Design. John Wiley and Sons, New York (1997) ISBN 0-471-12741-8

Gen, M., Cheng, R.: Genetic Algorithms and Engineering Optimization. John Wiley and Sons, New York (2000 ISBN 0-471-31531-1

Goldberg, D.E.: Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, Harlow (1989)

Gumus, A.T., Guneri, A.F.: Multi-echelon inventory management in supply chains with uncertain demand and lead times: literature review from an operational research perspective. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 221(10), 1553–1570 (2007)

Han, C., Damrongwongsiri, M.: Stochastic modeling of a two-echelon multiple sourcing supply chain system with genetic algorithm. Journal of Manufacturing Technology Management 16(1), 87–107 (2005)

Hnaien, F., Delorme, X., Dolgui, A.: Genetic algorithm for supply planning in two-level assembly systems with random lead times. Engineering Applications of Artificial Intelligence 22, 906–915 (2009)

Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)

Holweg, M., Disney, S., Holmstrom, J., Smaros, J.: Supply chain collaboration: Making sense of the strategy continuum. European Management Journal 23, 170–181 (2005)

Hou, E.S.H., Ansari, N., Ren, H.: A genetic algorithm for multiprocessor scheduling. IEEE Transactions on Parallel and Distributed Systems 5(2), 113–120 (1994)

Kimbrough, S.O., Wu, D.J., Zhong, F.: Computers play the beer game: Can artificial agents manage supply chains? Decision Support Systems 33, 323–333 (2002)

Lee, H.L., Padmanabhan, V., Whang, S.: Information distortion in a supply chain: The bullwhip effect. Management Science 43(4), 546–558 (1997)

O'Donnell, T., Maguire, L., McIvor, R., Humphreys, P.: Minimizing the bullwhip effect in a supply chain using genetic algorithms. International Journal of Production Research 44, 1523–1543 (2006)

Rosling, K.: Optimal inventory policies for assembly systems under random demands. Operations Research 37, 565–579 (1989)

Schwarz, L.B.: A simple continuous review deterministic one-warehouse n-retailer inventory problem. Management Science 19(5), 555–566 (1973)

Scott, E., Eheart, J.W., Ranjithan, S.: Using genetic algorithms to solve a multiobjective groundwater monitoring problem. Water Resources Research 31, 399–410 (1995)

Shin, H.J.: Collaborative production planning in a supply-chain network with partial information sharing. International Journal of Advances Manufacturing Technology 34, 981–987 (2007)

Silver, E.A., Pyke, D.F., Peterson, R.: Inventory Management and Production Planning and Scheduling, 3rd edn. John Wiley and Sons, West Sussex (1998) ISBN: 0471119474

Stank, P., Keller, S., Daugherty, P.: Supply chain collaboration and logistical service performance. Journal of Business Logistics 22(1), 29–49 (2001)

Syarif, A., Yun, Y., Gen, M.: Study on multi-stage logistic chain network: a spanning tre-based genetic algorithm approach. Computers & Industrial Engineering 43, 299–314 (2002)

Torabi, S.A., Fatemi Ghomi, S.M.T., Karimi, B.: A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chains. European Journal of Operational Research 173, 173–189 (2006)

van Houtum, G.J., Inderfurth, K., Zijm, W.H.M.: Materials coordination in stochastic multi-echelon systems. European Journal of Operational Research 95(1), 1–23 (1996)

Vergara, F.E., Khouja, M., Michalewicz, Z.: An evolutionary algorithm for optimizing material flow in supplpy chains. Computers & Industrial Engineering 43, 407–421 (2002)

Waller, M., Johnson, M.E., Davis, T.: Vendor managed inventory in the retail supply chain. Journal of Business Logistics 20(1), 183–203 (1999)

Wang, K., Wang, Y.: Applying genetic algorithms to optimize the cost of multiple sourcing supply chain systems an industry case study. Studies in Computational Intelligence 92, 355–372 (2008)

Wilding, R.: The supply chain complexity triangle: uncertainty generation in the supply chain. International Journal of Physical Distribution & Logistics Management 28(8), 599–616 (1998)

Yokoyama, M.: Integrated optimization of inventory distribution systems by random local seach and a genetic algorithm. Computers & Industrial Engineering 42, 172–188 (2002)

Zitzler, E., Deb, K., Thiele, L.: Comparison of multi-objective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (1998)

Fuzzy Skyhook Surface Control Using Micro-Genetic Algorithm for Vehicle Suspension Ride Comfort

Yi Chen

Abstract. A polynomial function supervised fuzzy sliding mode control (PSFαSMC), collaborated with a skyhook surface method, is presented for the ride comfort of a vehicle semi-active suspension. The multi-objective micro-genetic algorithm (MOμGA) has been utilised to the PSFαSMC controller's parameter alignment in a training process with three ride comfort objectives for the vehicle semi-active suspension, which is called the 'offline' step. Then, the optimised parameters are applied to the real-time control process by the polynomial function supervised controller, which is named 'online' step. A two degree of freedom dynamic model of a vehicle semi-active suspension system is given for passenger's ride comfort enhancement studies and a simulation with the given initial conditions has been devised in $MATLAB/SIMULINK$. The numerical results have shown that this hybrid control method is able to provide a real-time enhanced level of ride comfort performance for the semi-active suspension system.

1 Introduction

Ride comfort is a measure of the sensations felt by a vehicles' passengers whilst it is in motion. It is mainly dependent on the magnitude and type of the vibrations experienced by the vehicle body and is one of the most important characteristics to consider when designing a vehicle suspension system. Usually, a major component of vehicle body vibration is due to the road surface irregularities that are transmitted through the suspension system. The magnitude and characteristics of road surface irregularities vary from that of standard test roads to the random variations of the road surface elevation profile. An example of a vehicle suspension is shown in Fig. 1 [1], which includes tyres, springs, dampers and some other accessory

Yi Chen
School of Mechatronics Engineering,
University of Electronic Science and Technology of China, Chengdu, China, 611731
e-mail: leo.chen.yi@gmail.com

components. The multibody vehicle suspension dynamic model of a certain commercial vehicle has been built for the studies on the ride comfort and driveline torsional vibration, A, B, C and D are the rear and front tyres with the disturbances of the road roughness; E and F are the springs and the dampers; G is the five-gear manual transmission; H is the engine; I is the vehicle body, I_0 is the centre of mass (COM) of the rigid vehicle body.

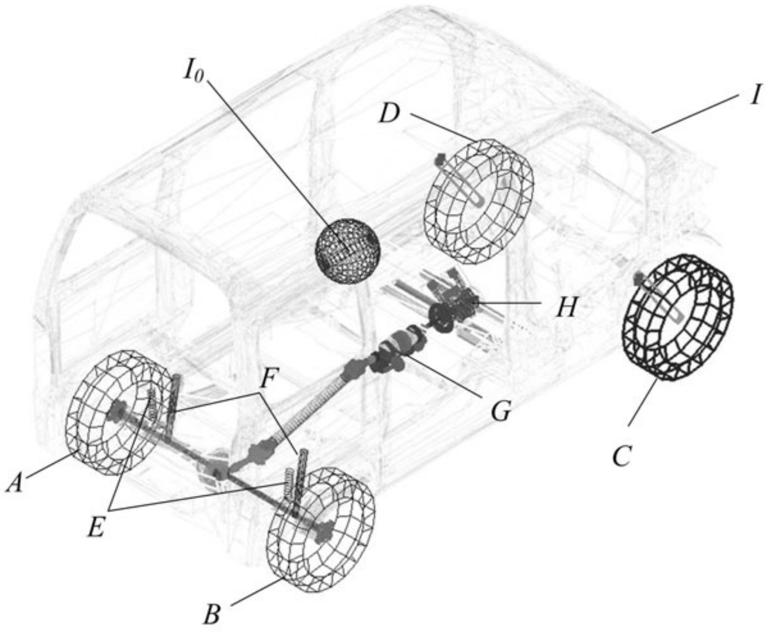


Fig. 1 A vehicle suspension system prototype [1].

1.1 *Passive, Active and Semi-active Suspensions*

As shown in Fig. 2, the main components include vehicle *body*, *springs*, *dampers* and *tyres*, in which

◇ (1) is a passive suspension system, and the spring stiffness, in which damping coefficient values are fixed and cannot be adjusted during operation;

◇ (2) is a semi-active suspension system, in which only the viscous damping coefficient can be changed and which does not invoke any energy inputs to the vehicle suspension system;

◇ (3) is an active suspension system, which uses the actuator to exert an independent force on the suspension system so as to improve the ride characteristics, and requires extra energy inputs.

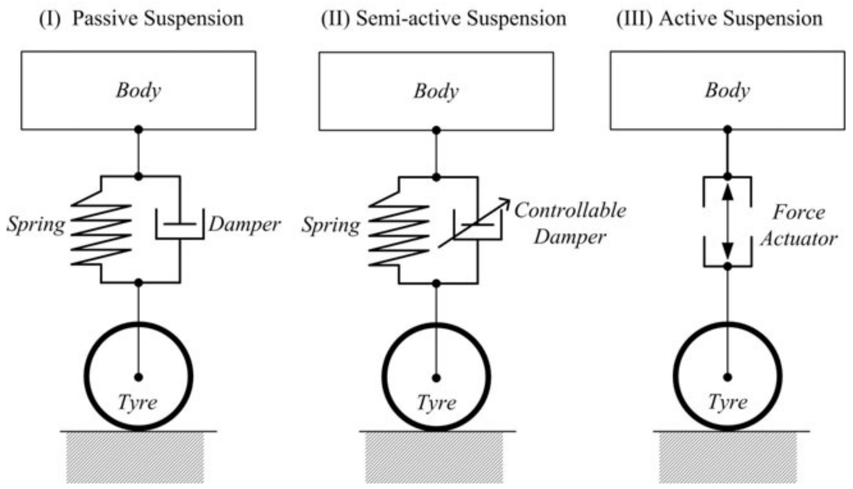


Fig. 2 Passive, semi-active and active suspension systems.

The active suspension system has been investigated since the 1930s, but due to the complexity and high cost of its hardware, it has not had widespread usage and is typically only implemented on sports vehicles, military vehicles or premium luxury vehicles [2]. The active suspension system is designed to use independent electromagnetic actuators to improve the suspension system's ride comfort performance.

By reducing the vibration transmission and keeping proper tyre contact, the active and semi-active suspension systems are designed and developed to achieve a better ride comfort performance than the passive suspension system.

The semi-active (SA) suspension system was introduced in the early 1970s [3], and it has been considered to be a good compromise between the active and the passive suspension systems. The conceptual idea of SA suspension involves replacing active force actuators with continually adjustable elements, which can vary or shift the rate of the energy dissipation in response to the instantaneous condition of motion. The SA suspension system only changes the viscous damping coefficient of the shock absorber, it does not add additional energy to the suspension system. The SA suspension system also has less energy costs than the active suspension system during average working conditions [4].

Over recent years, the research on the SA suspension system has continued to advance with respect to its capabilities, narrowing the gap between the SA and the active suspension system. The SA suspension system can achieve the majority of the performance characteristics of the active suspension system, which produces a wide class of practical applications. The magnetorheological (MR) and the electrorheological (ER) [5, 6, 7, 8] dampers are the most widely studied and tested components of the SA suspension system. MR and ER fluids are materials that respond to applied magnetic or electrical fields with a change in rheological behaviour.

In this chapter, a mathematical model for the SA suspension system will be discussed as the plant for the ride comfort control, in which the viscous damping coefficient is adjustable on the MR or ER damper. The micro-GA method will be utilised as the optimiser for the parameters of the newly proposed polynomial function supervised fuzzy sliding mode controller, which will then be applied to the ride comfort control for a semi-active suspension system.

1.2 Fuzzy Sliding Mode Control

The variable structure control (VSC) with sliding mode was introduced in the early 1950s by Emelyanov and was published in the 1960s [9], and further work was developed by several researchers [10, 11, 12]. Sliding mode control (SMC) has been recognized as a robust and efficient control method for complex high order nonlinear dynamical systems. Sliding mode control is one of the most popular methods, and has been applied to the MR/ER damper control for SA suspension systems. The major advantage of sliding mode control is its low sensitivity to changes of a system's parameters under various uncertain conditions, and that it can decouple system motion into independent partial components of lower dimension, which reduces the complexity of the system control and feedback design. A major drawback of traditional SMC is chattering, which is the high frequency oscillation of the system outputs, excited by the actuators (or sensors) ignored in the system's modelling process.

In order to deal with the chattering phenomenon, one of the widest known methods is the fuzzy logic theory. The fuzzy logic theory was first proposed by Zadeh in 1965 [13] and was based on the concept of fuzzy sets. Fuzzy logic control (FLC) has been used in a wide variety of applications in engineering, such as in aircraft/spacecraft, automated highway systems, autonomous vehicles, washing machines, process control, robotics control, decision-support systems, portfolio selection, etc. Practically speaking, it is not always possible to obtain a precise mathematical model for nonlinear, complex or ill-defined systems. FLC is a practical alternative for a variety of challenging control applications since it can provide a convenient method for constructing nonlinear controllers via the use of heuristic information (or knowledge). The heuristic information may come from an operator that acts as a 'human-in-the-loop' controller and from whom experimental data is obtained.

In recent years, a lot of literature were published in the area of fuzzy sliding mode control (FSMC) [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29], which covered the chattering phenomenon of the traditional SMC designing. The smooth control feature of fuzzy logic can be helpful in overcoming the disadvantages of chattering and this is why it can be useful to combine the FLC method with the SMC method, and thus to create the FSMC method. The involvement of FLC in the design of the FSMC based controller can be harnessed so as to help avoid the chattering problem.

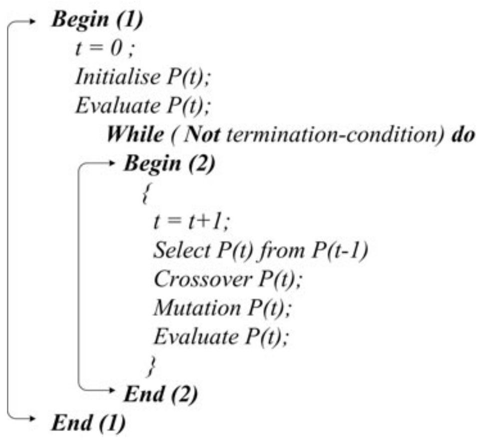
A fuzzy slide mode control with skyhook surface scheme will be discussed in section 5, and then an improved control method supervised by a polynomial function will be presented in section 6.

1.3 Genetic Algorithms

Many real world problems involve finding optimal parameters, which might prove difficult for traditional methods but are ideal for genetic algorithms (GA). The GA method was introduced in the 1970s by John Holland [30] at the University of Michigan. It is inspired by Darwin's theory of natural evolution and involves applying genetic operators, namely: selection, crossover and mutation, to a population of individuals.

As expressed in Fig. 3, a population $P(t)$ is a group of individuals created randomly and t is the timing argument. The individuals in the population are evaluated by a fitness function and then operated on by the three genetic operators.

The choice of fitness function is based on the criteria of the given task. The individuals are then selected based on their fitness. The higher the fitness, the higher the chance of being selected. These individuals then 'reproduce' to create one or more offspring, after which the offspring are mutated randomly. This loop continues until it reaches the termination condition, which could be when a suitable solution has been found or a certain number of generations have passed.



```
Begin (1)
  t = 0 ;
  Initialise P(t);
  Evaluate P(t);
     While ( Not termination-condition) do
     Begin (2)
        {
         t = t+1;
         Select P(t) from P(t-1)
         Crossover P(t);
         Mutation P(t);
         Evaluate P(t);
        }
     End (2)
End (1)
```

Fig. 3 The pseudo-code of the simple genetic algorithm.

The term micro-GA refers to a small population genetic algorithm with reinitialisation, which was first introduced by Krishnakumar [31]. The idea of micro-GA was supported by some theoretical results obtained by Goldberg [32], according to which a population size of three was sufficient to converge, regardless of the chromosomal length.

The population size used in a GA is usually of the order of tens to hundreds, or sometimes thousands. With such a large number of individuals, the time needed to perform calculations of the fitness function can become formidable. It is therefore important to design a high efficient GA for multi-objective optimisation problems. One of the popular methods in this direction is the micro-GA which has a very small internal population (3 to 6 individuals) [45].

2 Two Degree of Freedom Semi-active Suspension System

The role of the vehicle suspension system is to support and isolate the vehicle body and payload from road disturbances, and to maintain the traction force between the tyres and the road surface. The SA suspension system can offer both reliability and acceptable passenger ride comfort over a range of operating conditions with less power consumption than an active system.

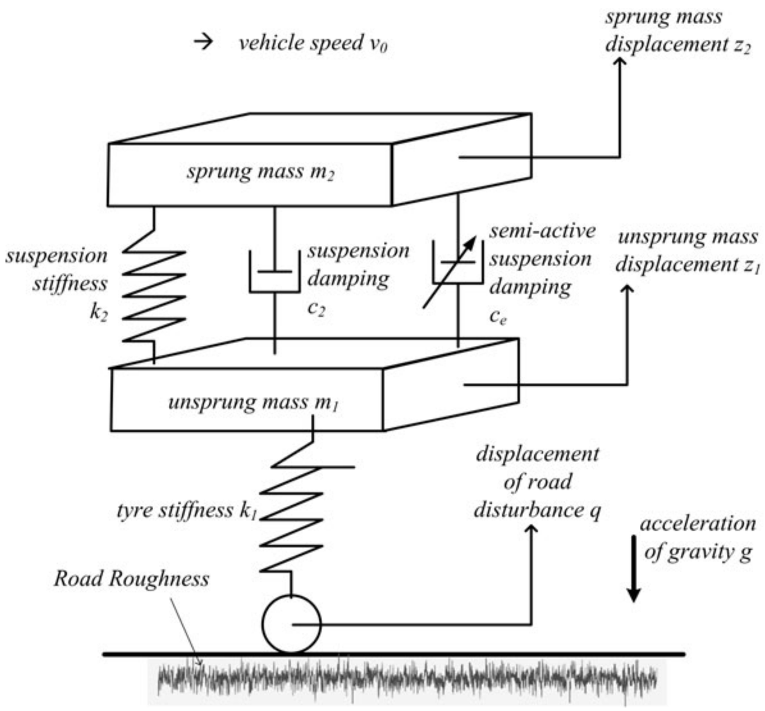

vehicle speed v_0
sprung mass displacement z_2
sprung mass m_2
suspension stiffness k_2
suspension damping c_2
semi-active suspension damping c_e
unsprung mass displacement z_1
unsprung mass m_1
displacement of road disturbance q
tyre stiffness k_1
acceleration of gravity g
Road Roughness

Fig. 4 Two degree of freedom semi-active suspension system.

To achieve a basic understanding of the passengers' response to the vehicle's vibrational behaviour, as shown in Fig. 4, a two-degree-of-freedom (2-DOF) vehicle ride model which focuses on the passenger ride comfort performance is represented for a SA suspension system, in which:

◇ m_1 and m_2 are the unsprung mass and the sprung mass respectively;
◇ k_1 is the tyre stiffness coefficient;
◇ k_2 and c_2 are the suspension stiffness and the suspension damping coefficient, respectively;
◇ c_e is the semi-active suspension damping coefficient, which can generate the semi-active damping force f_d by MR/ER absorber;
◇ z_1, z_2 and q are the displacements of the unsprung mass, the sprung mass and the road disturbance respectively;
◇ v_0 is the vehicle speed, which is the one of the input parameters for the road disturbance;
◇ g is the acceleration of gravity.

Using Newton's second law, the 2-DOF SA suspension model can be stated by the system's equations (1), where f_d is the damping force as stated by equation (2).

$$\begin{cases} m_1\ddot{z}_1 + k_2(z_2 - z_1) + (c_2 + c_e)(\dot{z}_2 - \dot{z}_1) - k_1(z_1 - q) + m_1 g = 0 \\ m_2\ddot{z}_2 - k_2(z_2 - z_1) - (c_2 + c_e)(\dot{z}_2 - \dot{z}_1) + m_2 g = 0 \end{cases} \tag{1}$$

$$f_d = c_e(\dot{z}_2 - \dot{z}_1) \tag{2}$$

$$\begin{cases} \dot{X} = AX + BQ + EU \\ Y = CX + DQ + FU \end{cases} \tag{3}$$

$$X = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} z_1 - q \\ z_2 - z_1 \\ \dot{z}_1 \\ \dot{z}_2 \end{Bmatrix} \tag{4}$$

$$Y = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix} = \begin{Bmatrix} \ddot{z}_2 \\ z_1 - q \\ z_2 - z_1 \end{Bmatrix} \tag{5}$$

$$f_{tyre} = k_1(z_1 - q) \tag{6}$$

$$U = \{\, f_d \,\} \tag{7}$$

$$Q = \begin{Bmatrix} \dot{q} \\ g \\ 0 \end{Bmatrix} \tag{8}$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ \frac{k_1}{m_1} & -\frac{k_2}{m_1} & \frac{c_2}{m_1} & -\frac{c_2}{m_1} \\ 0 & \frac{k_2}{m_2} & -\frac{c_2}{m_2} & \frac{c_2}{m_2} \end{bmatrix} \tag{9}$$

$$B = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & -\frac{1}{m_1} \\ 0 & -1 & \frac{1}{m_2} \end{bmatrix} \tag{10}$$

$$C = \begin{bmatrix} 0 & \frac{k_2}{m_2} & -\frac{c_0}{m_2} & \frac{c_0}{m_2} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{11}$$

$$D = \begin{bmatrix} 0 & -1 & \frac{1}{m_2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{12}$$

$$E = \begin{Bmatrix} 0 \\ 0 \\ -\frac{1}{m_1} \\ \frac{1}{m_2} \end{Bmatrix} \tag{13}$$

$$F = \begin{Bmatrix} \frac{1}{m_2} \\ 0 \\ 0 \end{Bmatrix} \tag{14}$$

In order to observe the state of the 2-DOF SA suspension system, Newton's second law equations, as given by equations (1), can be re-written as the state-space equations in equations (3). The state-space analysis concerns three types of variables (input variables, output variables and state variables) [20, 33], as shown in equations (4), (5) and (7) [34] in the form of matrixes (state matrices), where

◇ X is the state matrix for 2-DOF SA suspension system, which includes the tyre deformation ($x_1 = z_1 - q$), the suspension deformation ($x_2 = z_2 - z_1$), the unsprung mass velocity ($x_3 = \dot{z}_1$) and the sprung mass velocity ($x_4 = \dot{z}_2$), as given in equation (4). $\dot{X}$ is the derivative of the X, as can be seen in Fig. 5;

◇ Y is the output matrix with three state variables for the 2-DOF SA suspension system, which includes the vehicle body acceleration ($y_1 = \ddot{z}_2$), the tyre deformation ($y_2 = z_1 - q$), and the suspension deformation ($y_3 = z_2 - z_1$), as given in equation (5). According to the tyre deformation(y_2), the tyre load f_{tyre} is stated in equation (6);

◇ U is the input matrix (control force matrix) in equation (7);

◇ Q is the external road disturbance matrix in equation (8), which contains two external disturbance signals of road velocity profile and acceleration of gravity;

◇ A, B, C, D, E, F are the coefficient matrices in equations (9) to (14).

Practically, it is convenient to select the measurable quantities as the state variables by a block diagram, because the control law requires the feedback of all state variables with suitable weighting. The block diagram for the state-space equations is given in Fig. 5, which represents the 2-DOF SA suspension system for further controller design.

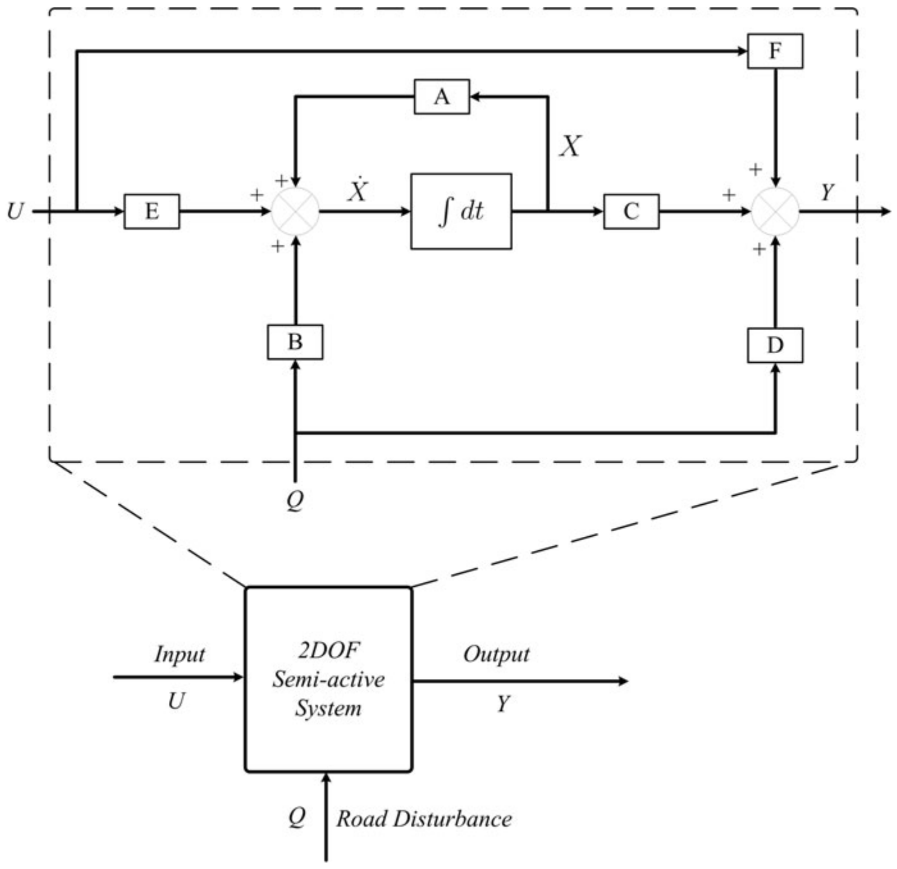


Fig. 5 Block diagram for the two degree of freedom semi-active suspension system.

3 Sliding Mode Control with Skyhook Surface Scheme

The skyhook control strategy was introduced in 1974 by Karnopp et al. [3]. As expressed in Fig. 6, the ideal skyhook damper scheme was introduced, which was one of the most effective methods in terms of the simplicity of the control algorithm. The basic idea is to link the vehicle body sprung mass to the stationary sky by a controllable ‘skyhook’ damper, which could reduce the vertical vibrations caused

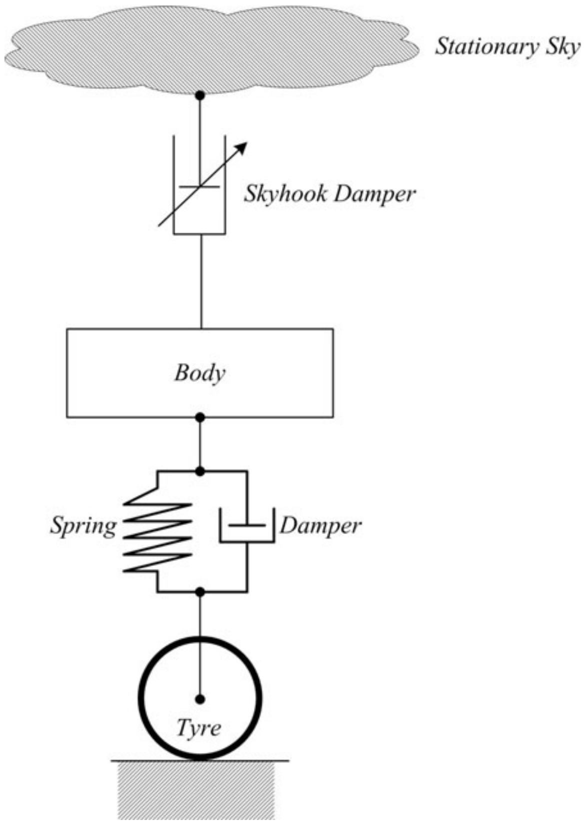


Fig. 6 Ideal damper of skyhook law, adopted from [3].

by road disturbance of all kinds. The skyhook control can reduce the resonant peak of the sprung mass quite significantly and thus produces a good ride quality by adjusting the skyhook damping coefficient when vehicle body velocity and other conditions are changing.

By borrowing the idea of skyhook damping, a soft switching control law is introduced [34, 35, 36] for the major sliding surface switching activity - the 'SkyhookSMC', which is utilised so as to reduce the sliding chattering phenomenon [34, 35, 36]. As shown in Fig. 7, $s = 0$ is the stationary sky location, $u_{SkyhookSMC}$ is the SkyhookSMC control force, the smooth control time-function generated by the SkyhookSMC is expressed in equation (15), where c_0 is an assumed positive damping ratio of the control law.

$$u_{SkyhookSMC} = \begin{cases} -c_0 \tanh\left(\frac{s}{\delta}\right) & s\dot{s} > 0 \\ 0 & s\dot{s} \leq 0 \end{cases} \tag{15}$$

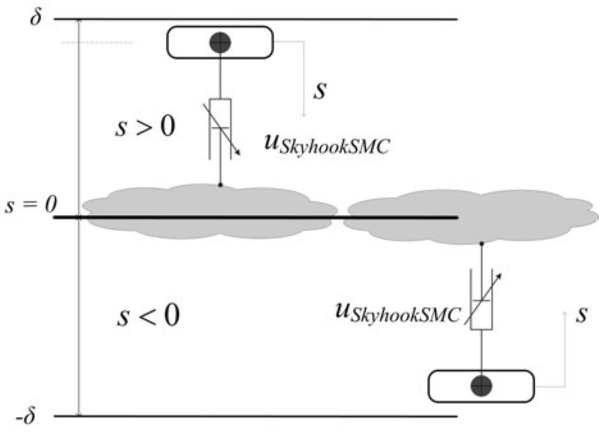


Fig. 7 Sliding mode surface with skyhook scheme [34, 35, 36].

The SkyhookSMC law needs to be chosen in such a way that the existence and the reachability of the sliding mode $s = 0$ are both guaranteed. It is noted that δ is an assumed positive constant, which defines the thickness of the sliding mode boundary layer [37].

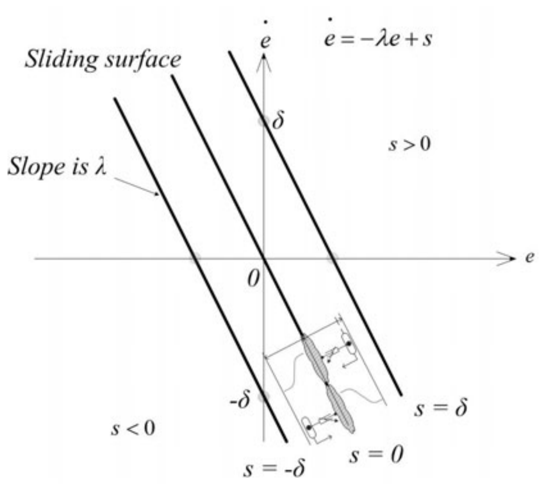


Fig. 8 Sliding surface generation with skyhook scheme [20, 35, 36, 37].

A plant is a set of functional objects or sub-systems to be controlled, such as a mechanical device, a chemical reactor or a spacecraft. In this chapter, the plant is the 2-DOF SA suspension system for its ride comfort control.

The SkyhookSMC technique defines a surface, along which the process slides to its desired value and a reaching function. As shown in Fig. 8, when designing a SkyhookSMC, the objective is to consider the 2-DOF suspension system as the control plant, which is defined by the state-space equations, as stated in equation (3), where

- ○ As given in equation (16), the sliding surface $s(e,t)$ is a function of the order of the process model by Slotine [20];
- ○ λ is a positive constant that defines the slope of the sliding surface;

$$s(e,t) = \left(\frac{d}{dt} + \lambda\right)^{n-1} e \tag{16}$$

As the 2-DOF SA suspension system is a second-order system, it can be given $n = 2$, in which s defines the position and velocity errors, and equation (16) can be re-written as equation (17).

$$s = \dot{e} + \lambda e \tag{17}$$

According to equation (17), the second-order tracking problem of 2-DOF SA suspension system is now being represented by a first-order stabilisation problem, in which s is kept at zero by means of a governing condition defined by the SkyhookSMC control law [20].

The SkyhookSMC is obtained from the use of the Lyapunov stability theorem, given in equation (18), and it states that the origin is a globally asymptotically stable equilibrium point for the control system. The Lyapunov function V, as given in Equation (18), is positive definite and its time derivative is given in inequality (19). To satisfy the negative definite condition, the control system should satisfy the inequality in (19).

$$V(s) = \frac{1}{2}s^2 \tag{18}$$

$$\dot{V}(s) = s\dot{s} = \lambda^2 e(t)\dot{e}(t) + \lambda\left(\dot{e}^2(t) + e(t)\ddot{e}(t)\right) + \dot{e}(t)\ddot{e}(t) < 0 \tag{19}$$

4 Fuzzy Logic Control

Generally, in the FLC design methodology, the human operator needs to write down a set of rules, which shows how to control the process and is called the ‘rule-base’. A fuzzy controller can emulate the decision-making process of the human under the guidance of the rule-base, in which the heuristic information (knowledge) may come from a control engineer who has performed extensive mathematical modelling, analysis, and development of control algorithms for a particular process.

Again, such expertise is loaded into the fuzzy controller to automate the reasoning processes and control actions. Regardless of where the heuristic information

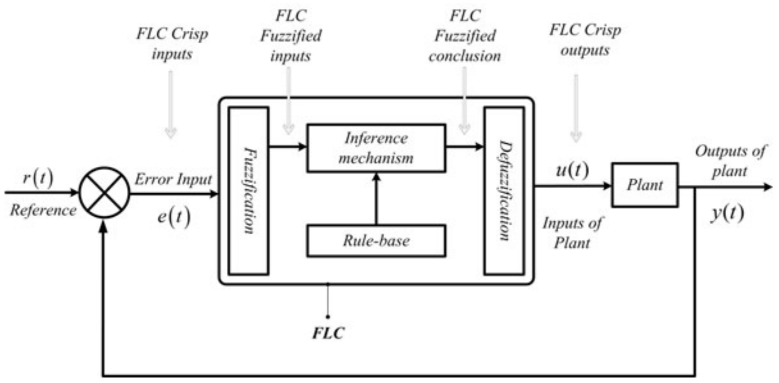


Fig. 9 Fuzzy logic controller architecture [35].

comes from, fuzzy control provides a user-friendly formalism for representing and implementing the ideas which can help to achieve high-performance control.

As shown in Fig. 9, the fuzzy controller has four main components:

- A ‘rule-base’ (a set of ‘IF-THEN’ rules) contains a fuzzy logic quantification of the expert’s linguistic description of how to achieve good control;
- An ‘inference mechanism’, which emulates the expert’s decision making in interpreting and applying knowledge about how efficiently to control the plant. A set of the ‘IF-THEN’ rules are loaded into the rule-base, and an inference strategy is chosen, then the system is ready to be tested, and the closed-loop specifications are needed;
- A ‘fuzzification’ interface converts ‘crisp’ inputs into ‘fuzzy’ information, in which the inference mechanism can be interpreted and compared to the rules in the rule-base;
- Conversely, a ‘defuzzification’ interface converts the conclusions by the inference mechanism into the FLC crisp (actual) outputs as the control inputs for the plant.

Briefly, fuzzy control system can be designed using the following steps:

⟨1⟩ Choosing the fuzzy controller inputs and outputs;
⟨2⟩ Choosing the preprocessing that is needed for the controller inputs and possibly post-processing that is needed for the outputs;
⟨3⟩ Designing each of the four components of the fuzzy controller, as shown in Fig. 9, which includes fuzzification, inference mechanism, rule-base and defuzzification;

Fuzzification is the process of decomposing the system inputs into fuzzy sets. That is, it is to map variables from the crisp space to the fuzzy space. The process of fuzzification allows the system inputs and outputs to be expressed in linguistic terms so that rules can be applied in a simple manner to express a complex system. In the

FLC for the 2-DOF SA suspension system, the velocity and acceleration of the vehicle body are selected as the crisp error (e) and the crisp change-in-error (ec) feedback signals for the 2-DOF SA suspension system control.

There are 7 linguistic terms in the fuzzy sets for two inputs of the fuzzified error (E) and the fuzzified change-in-error (EC), and one output of fuzzified force (U), which are: $\langle NL, NM, NS, ZE, PS, PM, PL \rangle$, as stated in Table 1, and their linguistic values are also listed in it for further numerical simulation with proper ranges of [-5,5] and [-2,2]. Defuzzification is the opposite process of fuzzification, it is to map variables from fuzzy space to crisp space.

Table 1 Fuzzy Linguistic Values

Fuzzy Linguistic Value	Description	E	EC	U
NL	Negative Large	-5	-5	-2
NM	Negative Middle	-4	-4	-1.5
NS	Negative Small	-3	-3	-1
ZE	Zero	0	0	0
PS	Positive Small	3	3	1
PM	Positive Middle	4	4	1.5
PL	Positive Large	5	5	2

A membership function (MF) is a manner that defines how each point in the input space is mapped to a membership value between 0 and 1. The MF for the 2-DOF SA suspension system is a triangular-shaped membership function and is defined by equation (20), where the parameters a and c locate the 'feet' of the triangle and the parameter b locates the peak.

The a, b and c are the parameters of fuzzy membership functions for the linguistic variables (E, EC and U), as listed in Table 1. In the case of 2-DOF SA suspension control, as shown in Fig. 10, the MF of the input E is stated, the MFs of EC and U are of similar shape to the range [-5,5] and [-2,2].

$$f(x,a,b,c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases} \tag{20}$$

The inputs of E and EC are interpreted from the fuzzy set, and the degree of membership is interpreted. The structure of the FLC for the 2-DOF SA suspension system

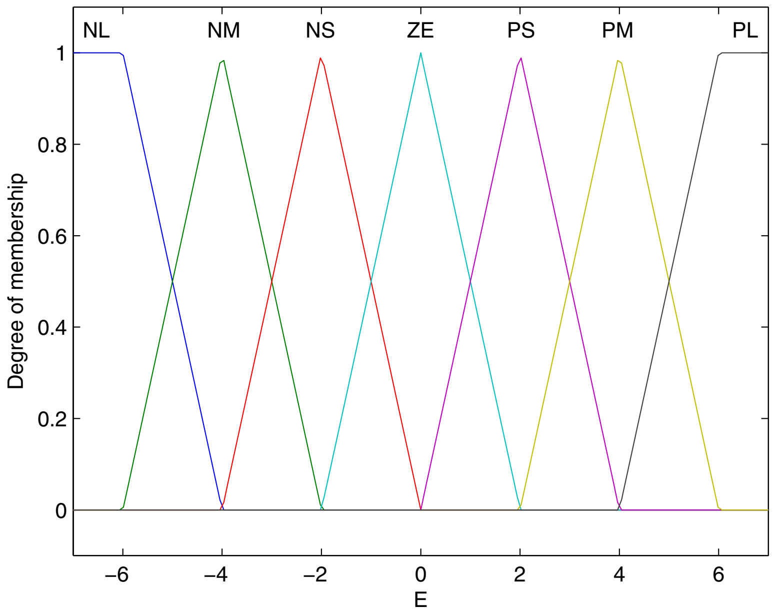


Fig. 10 The triangular-shaped membership function.

is a '2-in-1-out' FLC with two inputs and 1 output, and the 'If-Then' rule-base is then applied to describe the expert's knowledge. The FLC rule-base is characterised by a set of linguistic description rules based on conceptual expertise which arises from typical human situational experience. In particular, the 2-in-1-out FLC rule-base for the ride comfort of the 2-DOF SA suspension system is given in Table 2 [34], with two inputs and 7 linguistic values for each of the two inputs, there are at most $7^2 = 49$ possible rules as shown in the following list:

⟨1⟩ IF **E** = ***NL***, AND **EC** = ***NL***, THEN **U** = ***PL***;
⟨2⟩ IF **E** = ***NL***, AND **EC** = ***NM***, THEN **U** = ***PL***;
⟨3⟩ IF **E** = ***NL***, AND **EC** = ***NS***, THEN **U** = ***PM***;

⋮
⟨49⟩ IF **E** = ***PL***, AND **EC** = ***PL***, THEN **U** = ***NL***;

Table 2 came from the previous experience gained for the semi-active damping force control over the changing of the body acceleration for ride comfort control, which defines the relationship between 2 inputs of the fuzzified error (E) and the fuzzified change-in-error (EC) with 1 output of the fuzzified control force (U). Briefly, the main linguistic control rules are:

Table 2 2-in-1-out FLC rule table for 2-DOF SA suspension system [34]

U		EC						
		NL	**NM**	**NS**	**ZE**	**PS**	**PM**	**PL**
E	**NL**	PL	PL	PM	PS	PS	PS	ZE
	NM	PL	PM	PS	PS	PS	ZE	NS
	NS	PM	PS	ZE	ZE	ZE	NS	NM
	ZE	PM	PS	ZE	ZE	ZE	NS	NM
	PS	PM	PS	ZE	ZE	ZE	NS	NM
	PM	PS	ZE	ZE	ZE	NS	NM	NL
	PL	ZE	NS	NS	NS	NM	NL	NL

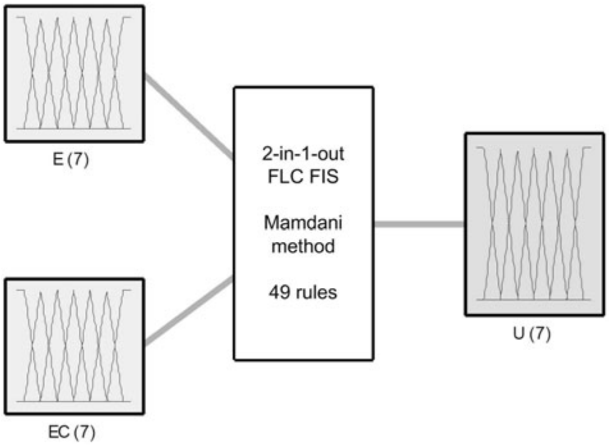


Fig. 11 The fuzzy inference system for 2-DOF SA suspension system.

⟨1⟩ when the body acceleration and velocity increase, the SA damping force decreases;

⟨2⟩ when the body acceleration and velocity decrease, the SA damping force increases.

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. The process of fuzzy inference involves all of the pieces that are described in the previous sections: Membership Functions, Logical Operations, and 'IF-THEN' Rules. Mamdani's fuzzy inference method [40, 41, 42] is the most commonly seen fuzzy methodology, it was among the first control systems built using fuzzy set theory.

It was proposed in 1975 by Mamdani as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani's effort was based on Zadeh's research [43] on fuzzy algorithms for complex systems and decision processes in 1973. As shown in Fig. 11, the Fuzzy Inference System (FIS) of Mamdani-type inference for the 2-in-1-out FLC is a type of fuzzy inference in which the fuzzy sets from the consequent of each rule are combined through the aggregation operator and the resulting fuzzy set is defuzzified to yield the output of the system.

5 Fuzzy Sliding Mode Control with Switching Factor α

A fuzzy sliding mode control with switching factor α (FαSMC) [35, 36] was introduced to combine the FLC with the SkyhookSMC to deal with the chattering phenomenon, which has been harnessed to reduce the 2-DOF SA suspension system ride comfort control with proper parameter selection.

A flow diagram for the FαSMC, applying the SkyhookSMC approach, is given in Fig. 12. The control effects of the FLC and the SkyhookSMC are combined by equation (21).

In equation (21), α is a switching factor which balances the weight of the FLC to that of the SkyhookSMC, $\alpha \in [0,1]$. Clearly,

$\alpha = 0$ represents SkyhookSMC, as discussed in section 3;

$\alpha = 1$ represents FLC, which has been discussed in section 4.

$$u_{F\alpha SMC} = \alpha u_{FLC} + (1-\alpha)\, u_{SkyhookSMC} \tag{21}$$

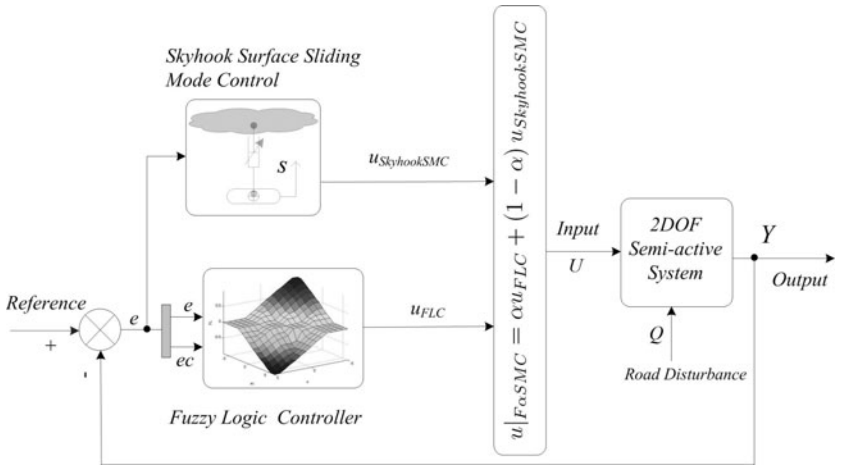


Fig. 12 FαSMC control flow diagram.

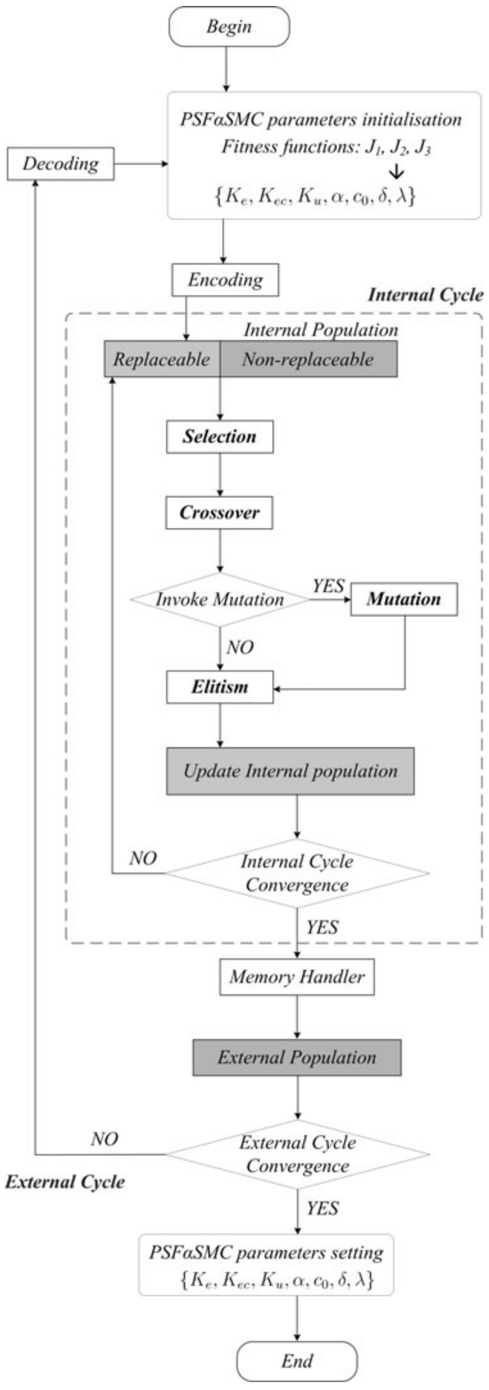

Begin
PSFαSMC parameters initialisation
Fitness functions: J_1, J_2, J_3
$\{K_e, K_{ec}, K_u, \alpha, c_0, \delta, \lambda\}$
Decoding
Encoding
Internal Cycle
Internal Population
Replaceable
Non-replaceable
Selection
Crossover
Invoke Mutation
YES
Mutation
NO
Elitism
Update Internal population
NO
Internal Cycle Convergence
YES
Memory Handler
External Population
NO
External Cycle Convergence
External Cycle
YES
PSFαSMC parameters setting
$\{K_e, K_{ec}, K_u, \alpha, c_0, \delta, \lambda\}$
End

Fig. 13 Micro-genetic algorithm for PSFαSMC work flow diagram.

6 Polynomial Function Supervising FαSMC - An Improvement

To make the proper improvement to the FαSMC method, a hybrid real-time polynomial function [44] supervised FαSMC with skyhook surface (PSFαSMC) is proposed and then will be applied to the ride comfort control for the 2-DOF SA vehicle suspension system.

The basic idea of PSFαSMC is to generate a series of supervised functions (SF) in polynomial function (PF) form. The SFs are used to generate parameters for FαSMC in the online real-time control step.

The concepts of SF and PF were proposed by Chen, et al. in 2004 [44], which are optimised and produced by micro-GA in the offline step for the PSFαSMC in this case of application. The micro-GA is a training process for the PSFαSMC parameter selection.

Briefly, there are two steps in PSFαSMC controller design: the offline step and the online step:

$\langle 1 \rangle$ the offline step is to take the micro-GA as the optimiser to generate polynomial functions for each parameter in FαSMC, including K_e, K_{ec}, K_u, α, c_0, δ, λ. In the micro-GA optimisation process, each loop will take more time than is practical for real-time control, so polynomial functions are used as practical real-time control parameter generators for the online step;

$\langle 2 \rangle$ the online step is to generate proper parameters using the polynomial functions, which came from offline step. The polynomial functions are the real-time parameter generators in the online step and will supervise the actual FαSMC control on SA semi-active suspension system by setting the proper parameters.

The parameters' selection for FαSMC needs a lot of manual testing which is time consuming. In order to reduce the working time for parameter selection in this hybrid control method PSFαSMC, the micro-genetic algorithm (micro-GA) is to be applied as the optimiser to generate proper results for parameter selection. This method has been widely applied in industrial applications [51] and in this case is utilised for the SA suspension system ride comfort control application.

6.1 Multi-objective Micro-GA for the Offline Step

Originally, Pareto optimality is a concept in economics with applications in engineering which named after an Italian economist Vilfredo Pareto. Given a set of alternative allocations of solutions for a set of individuals, a change from one allocation to another that makes at least one individual better off without making the rest worse off is called a Pareto improvement. A Pareto optimum is a maximal element for the partial order Pareto improvement: it is an allocation such that no other allocation is "better" in the sense of the order relation. An allocation is defined as "Pareto optimal" where no further Pareto improvements can be accessed [52, 53].

As shown in Fig. 13, generally there are two loops in the multi-objective micro-GA (MOμGA) process: the internal cycle and the external cycle. Meanwhile, there are also two groups of population memories: the internal population memory, which is used as the source of diversity of the micro-GA internal loop, and the external population memory, which is used to archive individuals of the Pareto optimal set [54]. The Pareto ranking is a kind of ranking method interested in the distribution of solutions, which asks how many individuals have a greater value than the given individual [55]. In the internal loop, the internal population memory is divided into two parts: a replaceable part and a non-replaceable part, and the percentages of each part can be adjusted by the user. In the elitism block, the Pareto ranking methods are taken into the process inside the internal cycle, which could include Goldberg's method [46] or Fonseca and Fleming's method [57].

For the small internal population, mutation is an optional operator for micro-GA. Practically, there are three parts in micro-GA optimisation: (1) fitness functions definition, (2) encoding and decoding definition and (3) genetic operators definition, which includes selection, crossover and mutation. Once the three parts have been well defined, the micro-GA can then create a population of solutions and apply genetic operators such as mutation and crossover to evolve the solutions in order to find better results.

As defined in equation (5), the output matrix Y contains three state variables for the 2-DOF SA suspension system, which are related to the ride comfort performance, including vehicle body acceleration (y_1), tyre deformation (y_2) and suspension deformation (y_3). $H(Y)$ is the error state function, as defined in equation (22), which can generate the error state variables (e_1, e_2 and e_3) for three output state variables (y_1, y_2 and y_3). $y_1|_{ref}$, $y_2|_{ref}$ and $y_3|_{ref}$ are the reference state variables for the PSFαSMC.

$$H(Y) == \begin{Bmatrix} e_1 \\ e_2 \\ e_3 \end{Bmatrix} = \begin{Bmatrix} y_1 - y_1|_{ref} \\ y_2 - y_2|_{ref} \\ y_3 - y_3|_{ref} \end{Bmatrix} = \begin{Bmatrix} \ddot{z}_2 - y_1|_{ref} \\ z_1 - q - y_2|_{ref} \\ z_2 - y_3|_{ref} \end{Bmatrix} \tag{22}$$

As shown in Fig. 14, in the PSFαSMC offline step, the micro-GA is used to optimise and generate polynomial functions for each parameter (K_e, K_{ec}, K_u, α, c_0, δ, λ), and there are three fitness values for three objectives as J_i, as stated in equation (23), where:

∘ $F(*)$ is the function for the fitness function;

$$J_i = F(e_i) = MIN\{RMS[ITAE(e_i)]\}, i = 1,2,3 \tag{23}$$

∘ $RMS[*]$ is the function for the values of root mean square, as defined in equation (24), t is the timer, n is the time series data of the error state variables e_i.

$$RMS[x_t] = \sqrt{\frac{x_1^2 + x_2^2 + \ldots + x_t^2}{n}}, t = 1, \ldots, n \tag{24}$$

According to the International Standard Organisation (ISO) 2631-1:1997 "Mechanical vibration and shock – Evaluation of human exposure to whole-body vibration – Part 1: General requirements" [58], the ride comfort is specified in terms of *RMS* acceleration over a frequency range, then the fitness functions (J_i) for the 2-DOF SA suspension system are the functions of the error state variables (e_i), which are

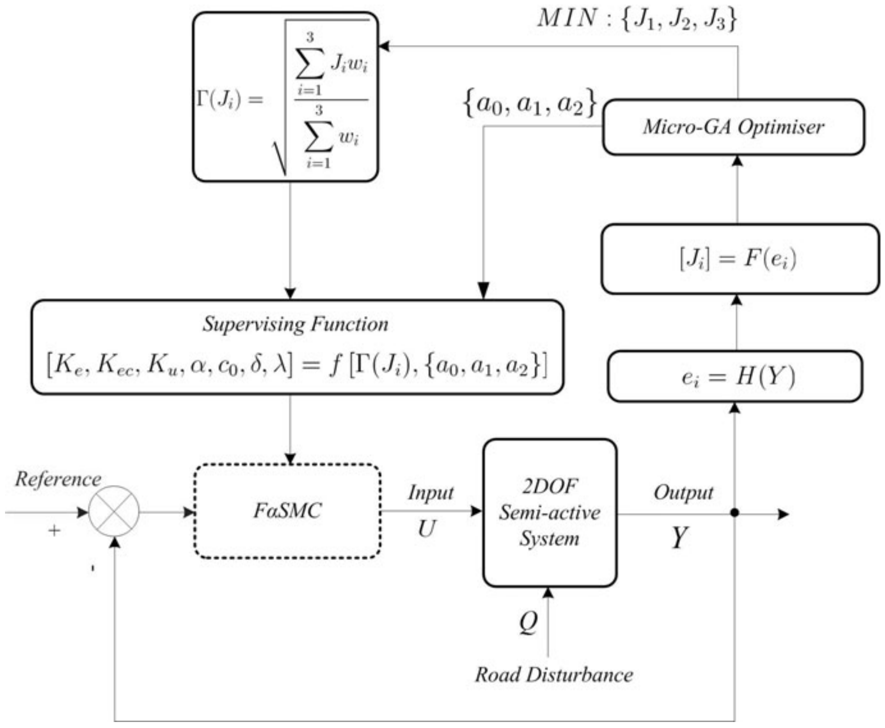

$MIN : \{J_1, J_2, J_3\}$
$\Gamma(J_i) = \sqrt{\frac{\sum_{i=1}^{3} J_i w_i}{\sum_{i=1}^{3} w_i}}$
$\{a_0, a_1, a_2\}$
Micro-GA Optimiser
$[J_i] = F(e_i)$
Supervising Function
$[K_e, K_{ec}, K_u, \alpha, c_0, \delta, \lambda] = f\,[\Gamma(J_i), \{a_0, a_1, a_2\}]$
$e_i = H(Y)$
Reference
+
FαSMC
Input
U
2DOF Semi-active System
Output
Y
Q
Road Disturbance

Fig. 14 PSFαSMC offline step - training.

the values of the root mean square ($RMS[*]$) of the integral of time, times the absolute error ($ITAE$) performance indexes, that is, J_1, J_2 and J_3 are the RMS of $ITAE$ indexes for the error state variables: body acceleration (e_1), suspension deformation (e_2) and tyre loads (e_3), respectively. The $ITAE$ of the error state variables (e_i) is expressed in equation (25).

$$ITAE\,(e_i) = \int_0^\infty t\,|e_i(t)|\,dt \tag{25}$$

As stated in equation(23), the micro-GA's optimising criteria is to minimise the fitness functions (J_i) and to generate a set of solutions for the better supervised function parameters (K_e, K_{ec}, K_u, α, c_0, δ, λ) via the interval arguments a_1, a_2, a_3 and Γ, which will be discussed in section 6.2.

Then, the micro-GA can provide the optimality of a set of solutions for the multi-objective applications of the ride comfort control in the online step, and the engineers can try each of the solution or select the solution by proper policy, e.g. outer range, inner range or average of the Pareto set as an engineering solution.

In the optimisation process by micro-GA, binary encoding/decoding, roulette-wheel selection, and single point crossover are taken in micro-GA evolutional process. In the micro-GA optimisation process, the initial conditions need to be given to the seven design variables, as given in Table 4.

6.2 Offline Step

As shown in Fig. 14, the offline step is a training process used to optimise and generate a series of polynomial functions for further use in the online step, and the micro-GA is the optimiser as discussed in section 6.1.

$$f\left[\Gamma(J_i),\{a_0,a_1,...,a_N\}\right] = a_N\Gamma(J_i)^N + a_{N-1}\Gamma(J_i)^{N-1} + ... + a_2\Gamma(J_i)^2 + a_1\Gamma(J_i) + a_0 \tag{26}$$

As stated in equation (26), $f\left[\Gamma(J_i),\{a_0,a_1,...,a_N\}\right]$ is the supervised function in polynomial form, which is fitted by the least squares principle based on output data from the 'micro-GA optimiser' block, where N is a positive integer; a_0, a_1, ..., a_N are constant coefficients.

$$\Gamma(J_i) = \sqrt{\frac{\sum_{i=1}^{3} J_i w_i}{\sum_{i=1}^{3} w_i}} \tag{27}$$

$\Gamma(J_i)$ is a component of the polynomial supervised function $f\left[\Gamma(J_i),\{a_0,a_1,...,a_N\}\right]$, as defined in equation (27), which is a weighted index by the optimised 2-DOF SA suspension ride comfort indexes J_i.

$$f\left[\Gamma(J_i),\{a_0,a_1,a_2\}\right]_{K_e} = a_2J_i^2 + a_1J_i + a_0 \tag{28}$$

Basically, a smooth supervised function curve is required in the SA suspension system ride comfort control, $N = 2$ is chosen as the highest degree of the supervised polynomial functions. Equation (28) is for the parameter K_e generation, and the similar processes will go with other parameters.

There are seven supervised functions for the seven PSFαSMC parameters (K_e, K_{ec}, K_u, α, c_0, δ, λ), as given in equation (29). As shown in Fig. 14, the supervised functions will be applied to the FαSMC block, where i = 1,2,3.

$$\begin{cases} K_e &= f\left[\Gamma(J_i),\{a_0,a_1,a_2\}\right]_{K_e} \\ K_{ec} &= f\left[\Gamma(J_i),\{a_0,a_1,a_2\}\right]_{K_{ec}} \\ K_u &= f\left[\Gamma(J_i),\{a_0,a_1,a_2\}\right]_{K_u} \\ \alpha &= f\left[\Gamma(J_i),\{a_0,a_1,a_2\}\right]_{\alpha} \\ c_0 &= f\left[\Gamma(J_i),\{a_0,a_1,a_2\}\right]_{c_0} \\ \delta &= f\left[\Gamma(J_i),\{a_0,a_1,a_2\}\right]_{\delta} \\ \lambda &= f\left[\Gamma(J_i),\{a_0,a_1,a_2\}\right]_{\lambda} \end{cases} \tag{29}$$

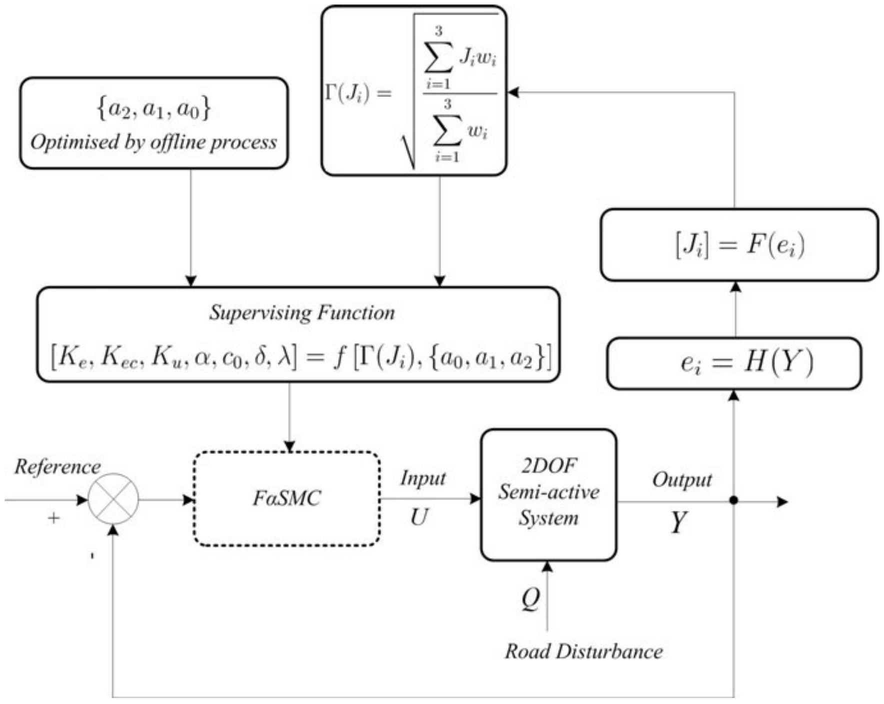

$\{a_2, a_1, a_0\}$
Optimised by offline process
$\Gamma(J_i) = \sqrt{\frac{\sum_{i=1}^{3} J_i w_i}{\sum_{i=1}^{3} w_i}}$
$[J_i] = F(e_i)$
Supervising Function
$[K_e, K_{ec}, K_u, \alpha, c_0, \delta, \lambda] = f[\Gamma(J_i), \{a_0, a_1, a_2\}]$
$e_i = H(Y)$
Reference
+
FαSMC
Input
U
2DOF Semi-active System
Output
Y
Q
Road Disturbance

Fig. 15 PSFαSMC online step - control for SA suspension system.

6.3 Online Step

As shown in Fig. 15, the online step is to apply the polynomial functions to supervise the FαSMC for the 2-DOF SA suspension system. There are seven design variables (K_e, K_{ec}, K_u, c_0, δ, λ, α) in PSFαSMC in the online step which are given in equation (29).

As stated in equation (29), the supervised functions are the functions of $\Gamma(J_i)$,a_0, a_1, a_2, which can generate the parameters for the FαSMC during the real-time simulation.

In Fig. 15, the polynomial functions are the optimised data source for FαSMC parameters (K_e, K_{ec}, K_u, α, c_0, δ, λ), which have been produced in the offline step, which include,

◇ K_e is FLC scaling gains for e;
◇ K_{ec} is FLC scaling gains for ec;
◇ K_u is FLC scaling gains for u;
◇ c_0 is SkyhookSMC damping coefficient;
◇ δ is the thickness of the sliding mode boundary layer;
◇ λ is the slope of the sliding surface;
◇ α is the switching factor of FαSMC in PSFαSMC.

Table 3 2-DOF SA vehicle suspension system parameters

m_1	unsprung mass	$36\ kg$
m_2	sprung mass	$240\ kg$
c_2	suspension damping coefficient	$1400\ Ns/m$
k_1	tyre stiffness coefficient	$160000\ N/m$
k_2	suspension stiffness coefficient	$16000\ N/m$
g	gravity acceleration	$9.81\ m/s^2$
Ω_0	reference spatial frequency	$0.1\ m^{-1}$
$S_g(\Omega_0)$	degree of roughness	$128 \times 10^{-6}\ m^2/cycles/m$
v_0	vehicle speed	$72\ km/h$
w_1	body acceleration weight factor	0.9
w_2	suspension deformation weight factor	0.05
w_3	tyre load weight factor	0.05

Table 4 Micro-GA parameters

external cycle	100
internal cycle	4
external population	50
internal population	6
replaceable population	2
crossover probability	0.9
a_0 initial range	$[-100, 100]$
a_1 initial range	$[-100, 100]$
a_2 initial range	$[-100, 100]$

Table 5 A set of polynomial function coefficients for a_i by Micro-GA

$\{a_2, a_1, a_0\}\|_{K_e}$	a_i coefficients of K_e	$\{-3.3, 2.11, 0.31\}$
$\{a_2, a_1, a_0\}\|_{K_{ec}}$	a_i coefficients of K_{ec}	$\{0.08, -0.19, -10.12\}$
$\{a_2, a_1, a_0\}\|_{K_u}$	a_i coefficients of K_u	$\{5.34, 0.61, 15.42\}$
$\{a_2, a_1, a_0\}\|_{\alpha}$	a_i coefficients of α	$\{-0.09, -0.22, 0.92\}$
$\{a_2, a_1, a_0\}\|_{c_0}$	a_i coefficients of c_0	$\{0.04, 1.56, 4999.04\}$
$\{a_2, a_1, a_0\}\|_{\delta}$	a_i coefficients of δ	$\{4.34, 1.86, 25.15\}$
$\{a_2, a_1, a_0\}\|_{\lambda}$	a_i coefficients of λ	$\{0.46, 0.26, 9.64\}$

6.4 Road Surface Profile

To simulate the road excitation for the vehicle suspension system, a road profile is defined as the cross sectional shape of a road surface under the given conditions, which can be expressed by statistical procedures [59, 60].

There are a few types of excitations for the road surface profile, such as sine waves, step functions and triangular waves, which can provide a basis for

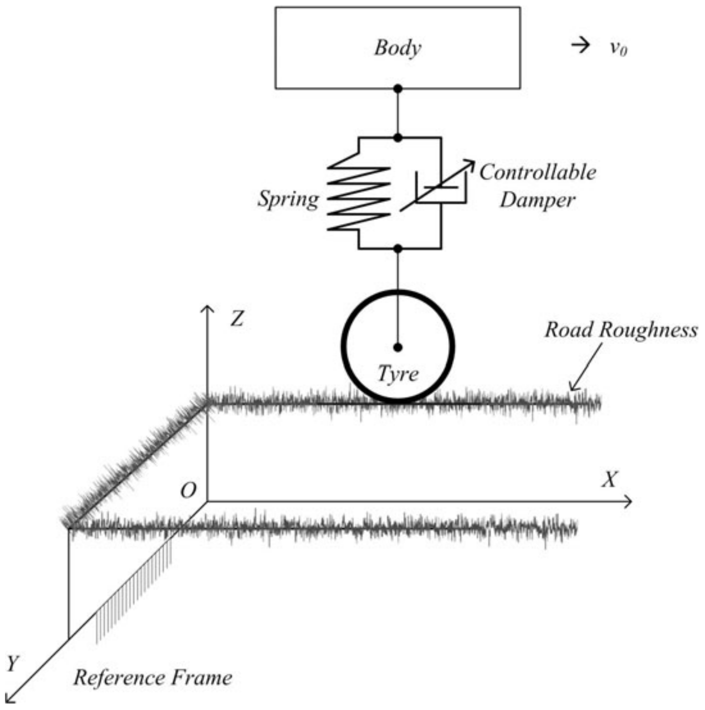


Fig. 16 Road profile as a random function.

comparative studies under some simple road surfaces, but these can hardly serve as a valid and general basis for a practical road roughness of ride behaviour.

As shown in Fig. 16, it is more realistic to describe a road surface profile as a random function or data sequence. There are some existing methods such as the International Roughness Index values and the Fourier transform-based sequence, described in the ISO 8608:1995 ‘Mechanical vibration-Road Surface Profiles-Reporting of Measured Data’[61], however, both only give an average condition for a relatively long section of the pavement.

◇ International Roughness Index (IRI);
◇ Mean Panel Rating (MPR);
◇ Profile Index (PI);
◇ Ride Number (RN);
◇ Slope Variance (SV);
◇ Root Mean Square Vertical Acceleration (RMSVA);
◇ Waveband Indices (WI);
◇ Wavelet Based Power Spectra (WPS);

One statistical way to generate road excitation is to describe the road roughness using power spectral density (PSD). When the road surface profile is regarded as a random function, it can be characterised by a PSD function [62].

To classify the roughness (irregularities) of road surfaces, the International Standards Organisation has proposed a road roughness classification, roughness-A (very good) to roughness-H (very poor) based on the PSD, in which the relationships between the PSD function $S_g(\Omega)$ and the spatial frequency Ω for different classes of road roughness can be approximated by two straight lines with different slopes on a $log-log$ scale, which can be expressed as equation (30) [63], and the values of N_1 and N_2 are 2.0 and 1.5 respectively.

$$\begin{cases} S_g(\Omega) = S_g(\Omega_0)\left(\dfrac{\Omega}{\Omega_0}\right)^{-N_1}, \Omega \leq \Omega_0 = \dfrac{1}{2\pi} cycles/m \\ \\ S_g(\Omega) = S_g(\Omega_0)\left(\dfrac{\Omega}{\Omega_0}\right)^{-N_2}, \Omega > \Omega_0 = \dfrac{1}{2\pi} cycles/m \end{cases} \tag{30}$$

In this case, to generate the road profile of a random base excitation for the 2-DOF SA suspension simulation, a spectrum of a geometrical road profile with road class 'roughness-C' is considered and Ω_0 is the reference spatial frequency. The vehicle is travelling at a constant speed v_0, and the historical road irregularity is given by the PSD method [64, 65, 66].

7 Simulations

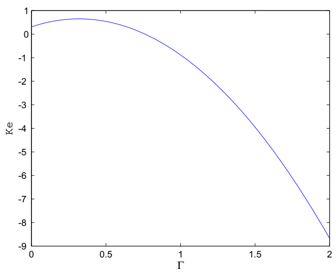
All the results for the ride comfort are obtained by using the parameters for the 2-DOF SA vehicle suspension system and PSFαSMC in Table 3. The numerical results are obtained using a specially devised simulation toolkit of Micro-GA for *MATLAB*, known henceforth here as *SGALAB* [67]. Unless stated otherwise all the results are generated using the parameters of the genetic algorithms as listed in Table 4, in which binary encoding/decoding, tournament selection, single point crossover and mutation are utilised by the Micro-GA evolutionary process, and a set of polynomial function coefficients a_i by Micro-GA are given in Table 5.

As discussed in section 6.1, there are three performance indexes for the vehicle suspension system, which includes body acceleration y_1, tyre deformation y_2 and suspension deformation y_3. In this context, the results for the three indexes are applied to evaluate the performance for the ride comfort of the 2-DOF SA vehicle suspension system.

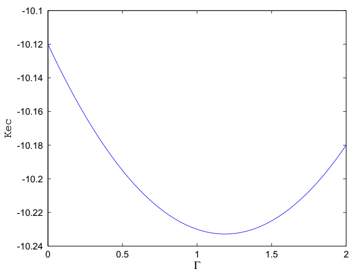
The PSFαSMC parameters require a judicious choice as follows:

$\diamond$ the FLC scaling gains of Ke and Kec for fuzzification of e and ec, Ku is the defuzzification gain factor;

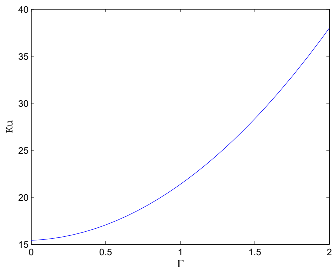
$\diamond$ the SkyhookSMC damping coefficient c_0, as stated in equation (15), is required to expand the normalised controller output force into a practical range. The thickness of the sliding mode boundary layer is given by δ, and the slope of



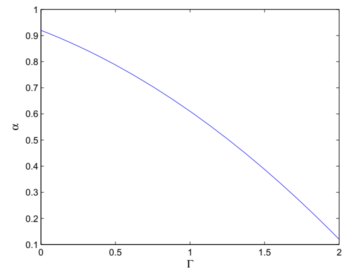
(a) K_e



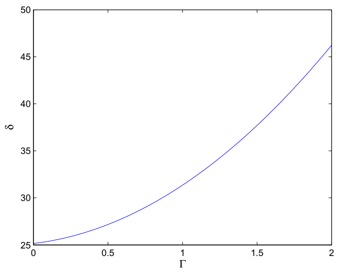
(b) K_{ec}



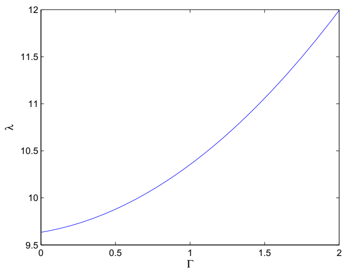
(c) K_u



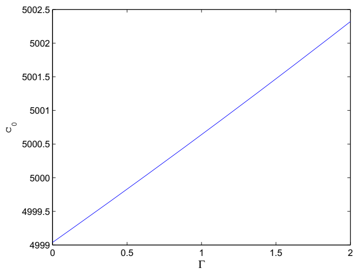
(d) α



(e) δ

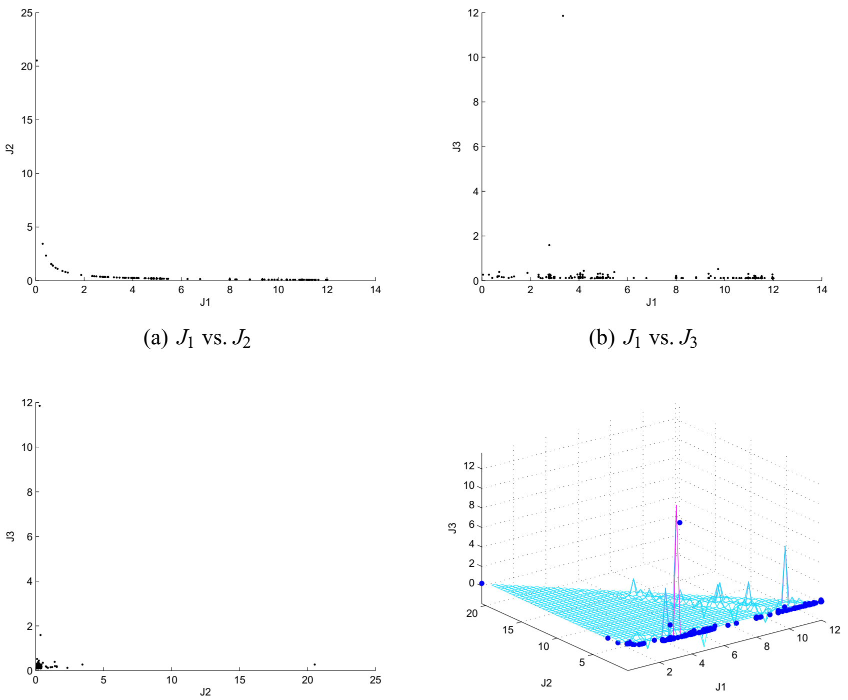


(f) λ



(g) c_0

Fig. 17 Polynomial supervised functions of PSFαSMC parameters for ride comfort control.



(a) J_1 vs. J_2 (b) J_1 vs. J_3

(c) J_2 vs. J_3 (d) J_1, J_2 and J_3

Fig. 18 Fitness functions J_1, J_2 and J_3 for the GA evaluation.

the sliding surface λ, both of δ and λ data come from design step by micro-GA in the offline step;

◇ in the PSFαSMC, α is required to balance the control weight between the FLC and SkyhookSMC. It is easy to switch the controller between the SkyhookSMC and FLC with a proper value of α.

◇ the coefficients $\{a_2, a_1, a_0\}$ for the polynomial supervised functions are given in Table 4, which are fitted by the least mean squares (LMS) algorithm based on data from offline step by micro-GA optimisation.

The polynomial functions for K_e, K_{ec}, K_u, α, δ, λ and c_0 are shown in Fig. 17(a) to Fig. 17(g) with a set of coefficients $\{a_2, a_1, a_0\}$, which are listed Table 5. The coefficients $\{a_2, a_1, a_0\}$ for the polynomial functions are optimised by the micro-GA in the offline step, and then applied to the supervised functions for the ride comfort control in the online step. As can be seen from Fig. 17(a) to Fig.17(g), the coefficients $\{a_2, a_1, a_0\}$ have direct effects on the shapes of the polynomial functions.

With the initial conditions for the micro-GA listed in Table 4, the evolutionary process for each fitness function is listed in Fig. 18, in which Fig. 18(a), Fig. 18(b), and Fig. 18(c) are the Pareto optimal set of J_1 vs. J_2, J_1 vs. J_3 and J_2 vs. J_3.

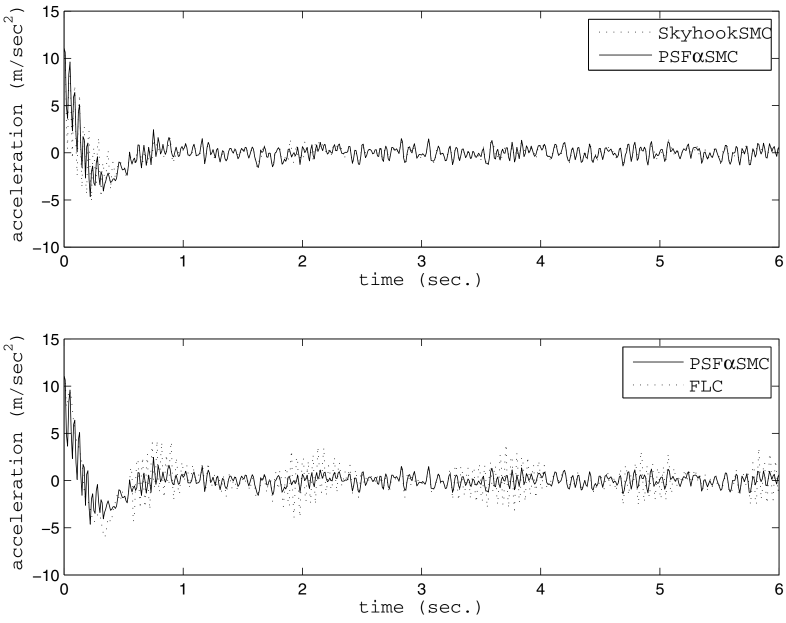


Fig. 19 Vehicle body acceleration response in time domain, y_1.

Fig.18(d) is a 3-D surface for the relationship among J_1, J_2 and J_3, in which the position with higher scatter data density means the Pareto optimal of 'trade-off' solutions. A set of polynomial function coefficients a_i for one of the selected Pareto front is given in Table 5.

Fig. 19 gives the suspension vertical behaviour of the body accelerations, PSFαSMC has a better control effect than the FLC and SkyhookSMC on the vehicle body acceleration, and both the PSFαSMC and SkyhookSMC methods can provide better ride comfort control effects than the FLC method on the 2-DOF SA suspension system.

Fig. 20 shows the tyre load response, PSFαSMC and SkyhookSMC control methods have the similar tyre load level, which is smaller (better) than the FLC control method for the 2-DOF SA suspension system.

Fig. 21 shows the relative displacement (suspension deformation) between vehicle sprung mass and unsprung mass. Compared with passive suspension deformation, PSFαSMC, SkyhookSMC and FLC control methods can reduce the 2-DOF SA suspension deformation, and the PSFαSMC and SkyhookSMC control methods have a similar suspension deformation level, and all of their suspension deformations are smaller than the FLC and passive suspension system's suspension deformation. That is, PSFαSMC can provide better ride comfort performance for the 2-DOF SA suspension system.

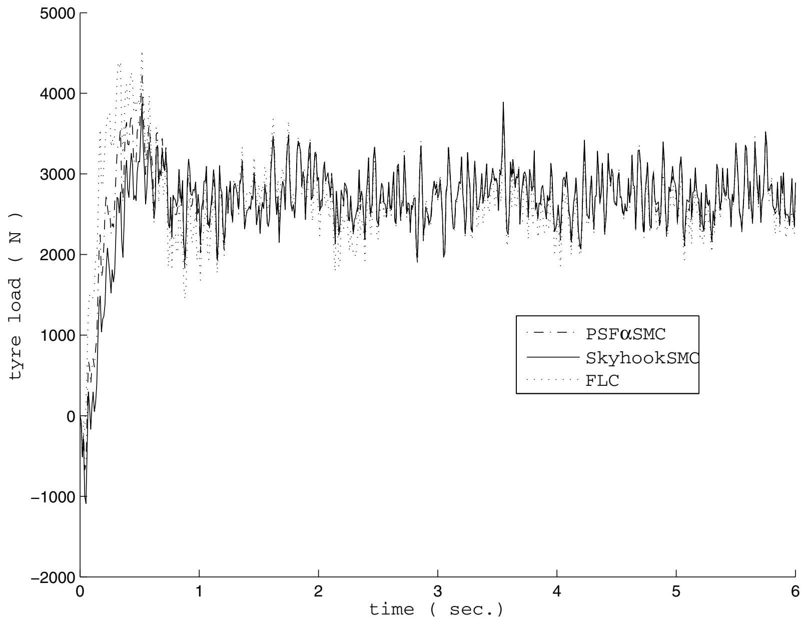


Fig. 20 Tyre load response in time domain, y_2.

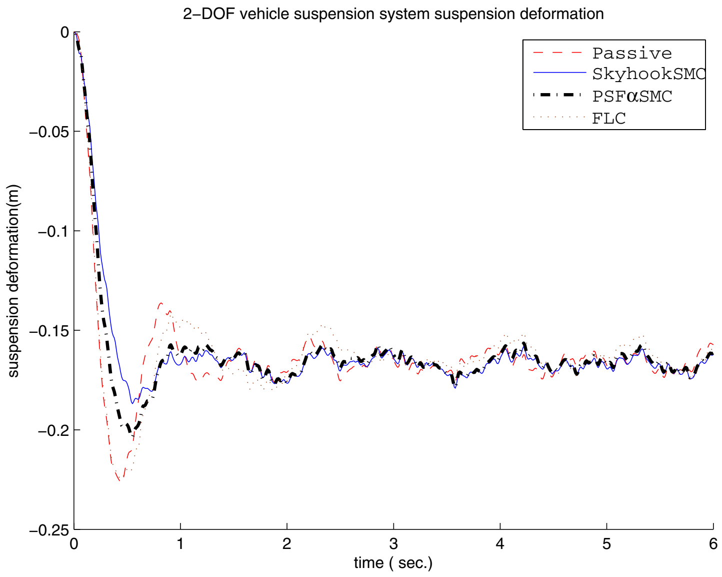


Fig. 21 Suspension deformation response in time domain, y_3.

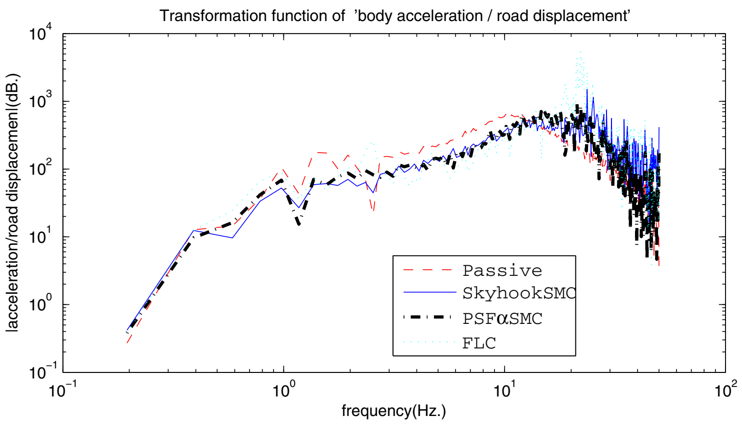


Fig. 22 Vehicle body acceleration response in frequency domain.

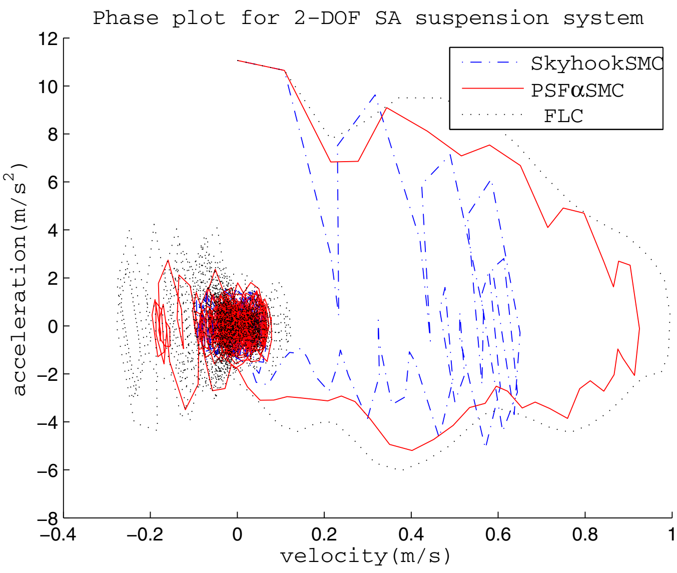


Fig. 23 Vehicle body response phase plot.

Fig. 22 is the body acceleration in frequency domain, which shows that the control methods of PSFαSMC, SkyhookSMC and FLC control methods can reduce the amplitudes at two of the key resonance points (1 Hz and 10^1 Hz). It also shows the PSFαSMC can have better control effects on the 2-DOF SA suspension system ride comfort than the FLC and the SkyhookSMC control methods for the 2-DOF SA

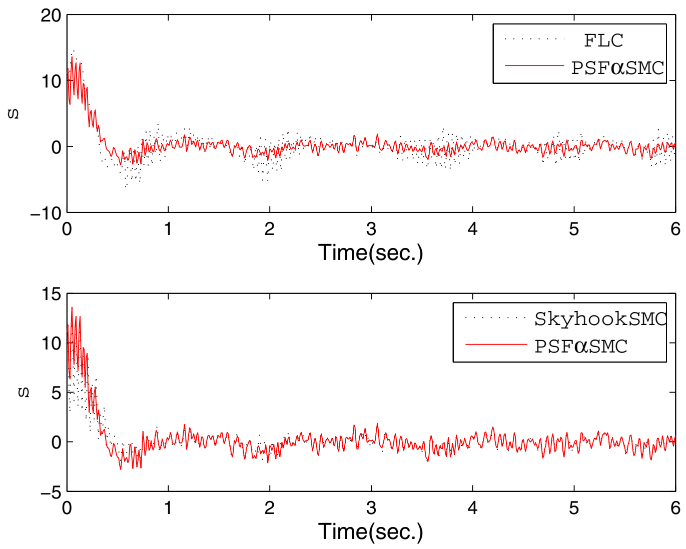


Fig. 24 Sliding surface switching plot.

suspension system, and in higher frequency range ($>$ 10 Hz) PSFαSMC has better performance than the other controllers, to some extent.

The phase plot (body velocity vs. body acceleration) is shown in Fig. 23 as the limit cycles, which represented the improved ride comfort performance of the 2-DOF SA suspension body vertical vibration with controllers. The curves, which corroborated the 2-DOF SA suspension system's interpretations of steady-state, started from the initial value point of (0, g) and gathered to the stable area around (0,0) in close-wise direction. The PSFαSMC goes faster than FLC, and smoother than SkyhookSMC to the steady-state area.

Fig. 24 shows that all the 2-DOF SA suspension system's sliding surfaces are switching and going around $s = 0$, and the PSFαSMC has the smaller and smoother switching behaviour than the FLC and the SkyhookSMC for the ride comfort control.

8 Conclusions

A polynomial function supervised fuzzy skyhook surface sliding mode control (PSFαSMC) has been presented for the ride comfort control on a 2-DOF semi-active vehicle suspension system, in which the MOμGA has been utilised to optimise the parameters for the polynomial functions in the offline step and the polynomial functions have been applied to supervise the PSFαSMC in the online step for the 2-DOF SA vehicle suspension system's ride comfort.

According to the simulation results, it has been demonstrated that the PSFαSMC can adjust control effects. It switches the factor α between the FLC and the SkyhookSMC. In the online step, the polynomial functions are supervising the control effects for the ride comfort control of the 2-DOF SA suspension system. In the offline step, the MOμGA has been applied as an optimiser for the parameters of the polynomial functions.

Also, the architecture for the PSFαSMC (controller) can be applied to the control applications for other dynamical systems and industrial processes.

References

1. Chen, Y.: Studies on SC6350C Driveline Torsional Vibration. Master Thesis, State Key Laboratory of Mechanical Transmission, Chongqing University (2004)
2. Yi, K., Wargelin, M., Hedrick, K.: Dynamic Tire Force Control by Semi-active Suspensions. Journal of Dynamic Systems, Measurement, and Control 115(3), 465–474 (1993)
3. Karnopp, D.C., Crosby, M.J., Harwood, R.A.: Vibration Control using Semi-Active Force Generators. Journals of Engineering for Industry, Transactions of the ASME 94, 619–626 (1974)
4. Jalili, N.: A Comparative Study and Analysis of Semi-Active Vibration-Control Systems. Journal of Vibration and Acoustics 124(4), 593–605 (2002)
5. Stanway, R.: The Development of Force Actuators using ER and MR Fluid Technology. Actuator Technology: Current Practice and New Developments, IEE Colloquium on (Digest No: 1996/110), 6/1-6/5 (1996)
6. Spencer Jr., B.F., Dyke, S.J., Sain, M.K., Carlson, J.D.: Phenomenological Model of Magnetorheological Damper. Journal of Engineering Mechanics 123(3), 230–238 (1997)
7. Caracoglia, L., Jones, N.: Passive Hybrid Technique for the Vibration Mitigation of Systems of Interconnected Stays. Journal of Sound and Vibration 307(3-5), 849–864 (2007)
8. Zhou, Q., Nielsen, S., Qu, W.: Semi-Active Control of Shallow Cables with Magnetorheological Dampers under Harmonic Axial Support Motion. Journal of Sound and Vibration 311(3-5), 683–706 (2008)
9. Emelyanov, S.V.: Variable Structure Control Systems. Nauka, Moscow (1967) (in Russian)
10. Itkis, Y.: Control Systems of Variable Structure. Wiley, New York (1976)
11. Utkin, V.A.: Sliding Modes and Their Application in Variable Structure Systems. Nauka (in Russian), Moscow (also Moscow: Mir, in English) (1978)
12. Hung, J.Y., Gao, W., Hung, J.C.: Variable Structure Control: A Survey. IEEE Transactions on Industrial Electronics, 2–22 (1993)
13. Zadeh, L.A.: Fuzzy Sets. Information and Control 8(3), 338–353 (1965)
14. Ishigame, A., Furukawa, T., Kawamoto, S., Taniguchi, T.: Sliding Mode Controller Design Based on Fuzzy Inference for Non-Linear System. In: International Conference on Industrial Electronics, Control and Instrumentation, Kobe, Japan, 28 Oct. - 1 Nov, vol. 3, pp. 2096–2101 (1991)
15. Ishigame, A., Furukawa, T., Kawamoto, S., Taniguchi, T.: Sliding Mode Controller Design Based on Fuzzy Inference for Nonlinear Systems. IEEE Trans. Industrial Electronics 40(1), 64–70 (1993)

16. O'Dell, B.: Fuzzy Sliding Mode Control: A Critical Review, Oklahoma State University, Advanced Control Laboratory, Technical Report ACL-97-001 (1997)
17. Ng, K.C., Li, Y., Murray-smith, D.J., Sharman, K.C.: Genetic Algorithm Applied to Fuzzy Sliding Mode Controller design. In: First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA, September 12-14, pp. 220–225 (1995)
18. Kung, C., Kao, W.: GA-based grey fuzzy dynamic sliding mode controller design. In: The 1998 IEEE International Conference on Fuzzy Systems Proceedings, IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, vol. 1, pp. 583–588 (1998)
19. Chen, P.C., Chen, C.W., Chiang, W.L.: GA-Based Fuzzy Sliding Mode Controller for Nonlinear Systems. Expert Systems with Applications: An International Journal 36(3), 5872–5879 (2009)
20. Slotine, J.J.E., Li, W.P.: Applied Nonlinear Control. Prentice-Hall International, Englewood Cliffs (1991)
21. Lo, J.C., Kuo, Y.H.: Decoupled Fuzzy Sliding-mode Control. IEEE Transactions Fuzzy Systems 6, 426–435 (1998)
22. Choi, B.J., Kwak, S.W., Kim, B.K.: Design of a Single-Input Fuzzy Logic Controller and its Properties. Fuzzy Sets and Systems 106, 299–308 (1999)
23. Kung, C., Chen, T., Kung, L.: Modified Adaptive Fuzzy Sliding Mode Controller for Uncertain Nonlinear Systems. IEICE Transactions Fundamentals E88-A(5), 1328–1334 (2005)
24. Wang, J., Rad, A.B., Chan, P.T.: Indirect Adaptive Fuzzy Sliding Mode Control: Part I: fuzzy switching. Fuzzy Sets and Systems 122(1), 21–30 (2001)
25. Wang, C.H., Liu, H.L., Lin, T.C.: Direct Adaptive Fuzzy-Neural Control with State Observer and Supervisory Controller for Unknown Nonlinear Dynamical Systems. The IEEE Transactions on Fuzzy Systems 10, 39–49 (2002)
26. Yau, H.T., Chen, C.L.: Chattering-free Fuzzy Sliding-Mode Control Strategy for Uncertain Chaotic Systems. Chaos, Solitons and Fractals 30, 709–718 (2006)
27. Eksin, I., Güzelkaya, M., Tokat, S.: Self-Tuning Mechanism for Sliding Surface Slope Adjustment in Fuzzy Sliding Mode Controllers. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 216(5), 393–406 (2002)
28. Iglesias, E., García, Y., Sanjuan, M., Camacho, O., Smith, C.: Fuzzy Surface-Based Sliding Mode Control. ISA Transactions 46(1), 73–83 (2007)
29. Roopaeia, M., Zolghadrib, M., Meshksarc, S.: Enhanced Adaptive Fuzzy Sliding Mode Control for Uncertain Nonlinear Systems. Communications in Nonlinear Science and Numerical Simulation 14(9-10), 3670–3681 (2009)
30. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
31. Krishnakumar, K.: Micro-genetic Algorithms for Stationary and Non-Stationary Function Optimization. In: SPIE: Intelligent Control and Adaptive Systems, Philadelphia, PA, vol. 1196, pp. 289–296 (1989)
32. Goldberg, D.: Sizing Populations for Serial and Parallel Genetic Algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, vol. 1196, pp. 70-79 (1989)
33. Ogata, K.: Modern Control Engineering. Prentice-Hall, Englewood Cliffs (1996)
34. Chen, Y.: Skyhook Surface Sliding Mode Control on Semi-active Vehicle Suspension Systems for Ride Comfort Enhancement. Engineering 1(1), 23–32 (2009)

35. Chen, Y., Cartmell, M.P.: Hybrid Fuzzy and Sliding-Mode Control for Motorised Tether Spin-Up When Coupled with Axial Vibration. In: 7th International Conference on Modern Practice in Stress and Vibration Analysis, New Hall, Cambridge, September 8-10 (2009)
36. Chen, Y., Cartmell, M.P.: Hybrid Fuzzy Skyhook Surface Sliding Mode Control for Motorised Space Tether Spin-up Coupled with Axial Oscillation. In: Advanced Problems in Mechanics, 30 June - 5 July. Russian Academy of Sciences, St. Petersburg, Russia (2009)
37. Slotine, J.J.E.: Tracking Control of Non-Linear Systems using Sliding Surfaces with Application to Robot Manipulations. PhD Dissertation, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology (1982)
38. Burns, R.: Advanced Control Engineering. Butterworth-Heinemann, Oxford (2001)
39. Passino, K.M., Yurkovich, S.: Fuzzy Control. Addison Wesley Longman, Menlo Park (1998)
40. Mamdani, E.H., Assilian, S.: An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. International Journal of Man-Machine Studies 7(1), 1–13 (1975)
41. Mamdani, E.H.: Advances in the Linguistic Synthesis of Fuzzy Controllers. International Journal of Man-Machine Studies 8(1), 669–678 (1976)
42. Mamdani, E.H.: Applications of Fuzzy Logic to Approximate Reasoning using Linguistic Synthesis. IEEE Transactions on Computers 26(12), 1182–1191 (1977)
43. Zadeh, L.A.: Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. IEEE Transactions on Systems, Man, and Cybernetics 3(1), 28–44 (1973)
44. Chen, Y., Fang, Z., Luo, H., Deng, Z.: Simulation Research on Real-time Polynomial Function Supervising PID Control Based on Genetic Algorithms. Journal of System Simulation 16(6), 1171–1174 (2004)
45. Coello Coello, C.A., Pulido, G.: A Micro-Genetic Algorithm for Multiobjective Optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 126–140. Springer, Heidelberg (2001)
46. Deb, K., Goldberg, D.E.: An Investigation of Niche and Species Formation in Genetic Function Optimization. In: Schaffer, J.D. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 42–50. Morgan Kaufmann, San Mateo (1989)
47. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Forrest, S. (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 416–423. Morgan Kaufmann, San Mateo (1993)
48. Lo, K.L., Khan, L.: Hierarchical Micro-Genetic Algorithm Paradigm for Automatic Optimal Weight Selection. H_∞ Loop-Shaping Robust Flexible AC Transmission System Damping Control Design 151(1), 109–118 (2004)
49. Tam, V., Cheng, K.Y., Lui, K.S.: Using Micro-Genetic Algorithms to Improve Localization in Wireless Sensor Networks. Journal of Communications 1(4), 1–10 (2006)
50. Davidyuk, O., Selek, I., Ceberio, J., Riekki, J.: Application of Micro-Genetic Algorithm for Task Based Computing. In: Proceeding of International Conference on Intelligent Pervasive Computing (IPC 2007), Jeju Island, Korea, pp. 140–145 (October 2007)
51. Szőllős, A., Šmíd, M., Hájek, J.: Aerodynamic Optimization via Multi-Objective Micro-Genetic Algorithm with Range Adaptation, Knowledge-Based Reinitialization. Crowding and ε-Dominance 40(6), 419–430 (2009)
52. Sen, A.: Markets and Freedom: Achievements and Limitations of the Market Mechanism in Promoting Individual Freedoms. Oxford Economic Papers 45(4), 519–541 (1993)

53. Barr, N.: Economics of the Welfare State. Oxford University Press, New York (2004)
54. Ehrgott, M.: Multicriteria Optimization, 2nd edn. Springer, Berlin (2005) ISBN 3-540-21398-8
55. Santana-Quintero, L.V., Montaõ, A.A., Coello Coello, C.A.: A Review of Techniques for Handling Expensive Functions in Evolutionary Multi-Objective Optimization. In: Tenne, Y., Goh, C.-K. (eds.) Computational Intelligence in Expensive Optimization Problems, vol. 2, pp. 29–59. Springer, Heidelberg (2010)
56. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Kluwer Academic Publishers, Boston (1989)
57. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Genetic Algorithms: Proceedings of the Fifth International Conference, pp. 416–423. Morgan Kaufmann, San Mateo (1993)
58. International Organization for Standardization: Mechanical Vibration and Shock – Evaluation of Human Exposure to Whole-Body Vibration – Part 1: General Requirements, ISO 2631-1:1997 (1997)
59. Sayers, M.W.: Interpretation of Road Roughness Profile Data. Final Report UMTRI-96-19 (1996)
60. Wei, L., Fwa, T.F., Zhe, Z.: Wavelet Analysis and Interpretation of Road Roughness. Journal of Transportation Engineering 131(2), 120–130 (2005)
61. International Organization for Standardization: ISO 8608:1995 Mechanical Vibration-Road Surface Profiles-Reporting of Measured Data (1995)
62. Wong, J.Y.: Theory of Ground Vehicles, 3rd edn. John Wiley & Sons, Chichester (2001)
63. International Organization for Standardization: ISO/TC108/SC2/WG4 N57 Reporting Vehicle Road Surface Irregularities (1982)
64. Elbeheiry, E.M., Karnopp, D.C.: Optimal Control of Vehicle Random Vibration with Constrained Suspension Deflection. Journal of Sound and Vibration 189(5), 547–564 (1996)
65. Ramji, K., Gupta, A., Saran, V.H., Goel, V.K., Kumar, V.: Road Roughness Measurements using PSD Approach. Journal of the Institution of Engineers 85, 193–201 (2004)
66. Liu, Y., Matsuhisaa, H., Utsunoa, H.: Semi-active Vibration Isolation System with Variable Stiffness and Damping Control. Journal of Sound and Vibration 313(1-2), 16–28 (2008)
67. Chen, Y.: Simple Genetic Algorithm Laboratory Toolbox for MATLAB (2009), http://www.mathworks.co.uk/matlabcentral/fileexchange/5882

Notations

m_1	the unsprung mass
m_2	the sprung mass
k_1	tyre deflection stiffness coefficient
k_2	the suspension stiffness coefficient
c_2	the suspension damping coefficient
c_e	the semi-active suspension damping coefficient
z_1	the displacements for unsprung mass
z_2	the displacements for sprung mass
q	the road disturbance
v_0	the vehicle speed
g	the acceleration of gravity
f_d	the semi-active control force
f_{tyre}	the tyre load
X	the state matrix
Y	the output matrix
U	the input matrix
Q	the external road disturbance matrix
A	the coefficient matrices
B	the coefficient matrices
C	the coefficient matrices
D	the coefficient matrices
E	the coefficient matrices
F	the coefficient matrices
y_1	the output state variable - the body acceleration
y_2	the output state variable - the tyre deformation
y_3	the output state variable - the suspension deformation
$y_1\|_{ref}$	the reference state variable for y_1
$y_2\|_{ref}$	the reference state variable for y_2
$y_3\|_{ref}$	the reference state variable for y_3
x_1	the state variable - the tyre deformation
x_2	the state variable - the suspension deformation
x_3	the state variable - the unsprung mass velocity
x_4	the state variable - the sprung mass velocity
n	the order number of a non-linear system
N	the degree of a polynomial function
α	the switching factor
s	the sliding surface
V	the Lyapunov function
c_0	the positive damping ratio for SkyhookSMC
e	error
e_i	error state variable

ec	change-in-error
$H(*)$	the error state function
E	fuzzified error
EC	fuzzified change-in-error
$u_{F\alpha SMC}$	the control force by the FαSMC
u_{FLC}	the control force by the FLC
$u_{SkyhookSMC}$	the control force by the SkyhookSMC
K_e	the factor for e
K_{ec}	the factor for ec
K_u	the factor for u
δ	the thickness of the sliding mode boundary layer
λ	the slope of the sliding surface
J_1	the fitness function for y_1
J_2	the fitness function for y_2
J_3	the fitness function for y_3
a_i	the constant coefficient for the polynomial functions
$\Gamma(*)$	the weighted index
Ω	the spatial frequency
Ω_0	the reference spatial frequency
$S_g(\Omega_0)$	degree of roughness
w_1	the body acceleration weight factor
w_2	the suspension deformation weight factor
w_3	the tyre load weight factor
RMS	root mean square
$ITAE$	the integral of time times the absolute error

Subject Index

Author Index