Studies in Computational Intelligence 610

Stefka Fidanova *Editor*

Recent Advances in Computational Optimization

Results of the Workshop on Computational Optimization WCO 2014

Springer

Studies in Computational Intelligence

Volume 610

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

About this Series

The series "Studies in Computational Intelligence" (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the worldwide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at http://www.springer.com/series/7092

Stefka Fidanova
Editor

Recent Advances in Computational Optimization

Results of the Workshop on Computational Optimization WCO 2014

Springer

Editor
Stefka Fidanova
Institute of Information and Communication
Technology
Bulgarian Academy of Sciences
Sofia
Bulgaria

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-319-21132-9 ISBN 978-3-319-21133-6 (eBook)
DOI 10.1007/978-3-319-21133-6

Library of Congress Control Number: 2015943370

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Preface

Many real-world problems arising in engineering, economics, medicine and other domains can be formulated as optimization tasks. Every day we solve optimization problems. Optimization occurs in the minimizing time and cost or the maximization of the profit, quality and efficiency. Such problems are frequently characterized by non-convex, non-differentiable, discontinuous, noisy or dynamic objective functions and constraints which ask for adequate computational methods.

This volume is a result of very vivid and fruitful discussions held during the Workshop on Computational Optimization, WCO-2014. The participants agreed that the relevance of the conference topic and quality of the contributions clearly suggest that a more comprehensive collection of extended contributions devoted to the area would be very welcome and would certainly contribute to a wider exposure and proliferation of the field and ideas.

The volume includes important real problems like parameter settings for controlling processes in bioreactor and other processes, resource constrained project scheduling, infection distribution, molecule distance geometry, quantum computing, real-time management and optimal control, bin packing, medical image processing, localization the abrupt atmospheric contamination source and so on.

Some of these problems can be solved applying traditional numerical methods, but others need a huge amount of computational resources. Therefore, for them it is more appropriate to develop algorithms based on some metaheuristic method like the evolutionary computation, ant colony optimization, constrain programming, etc.

April 2015 Stefka Fidanova

Organising Committee

Workshop on Computational Optimization, WCO-2014, was organized within the Federated Conference on Computer Science and Information Systems FedCSIS-2014.

Workshop Co-chairs

Stefka Fidanova, IICT (Bulgarian Academy of Sciences, Bulgaria)
Antonio Mucherino, IRISA (Rennes, France)
Daniela Zaharie, West University of Timisoara (Romania)

Program Committee

David Bartl, University of Ostrava, Czech Republic
Mihaela Breaban, University of Iasi, Romania
Paolo Cremonesi, Politecnico di Milano, Italy
Stefano Gualandi, University of Pavia, Italy
Le Thi Hoai Ann, University of Lorraine, France
Hiroshi Hosobe, National Institute of Informatics, Japan
Hideaki Iiduka, Kyushu Institute of Technology, Japan
Carlile Lavor, IMECC-UNICAMP, Campinas, Brazil
Pencho Marinov, Bulgarian Academy of Science, Bulgaria
Carla Michini, ETH Zürich, Switzerland
Kaisa Miettinen, University of Jyväskylä, Finland
Stelian Mihalas, West University of Timişoara, Romania
Ionel Muscalagiu, Politehnica University Timişoara, Romania
Giacomo Nannicini, University of Technology and Design, Singapore
Konstantinos Parsopoulos, University of Patras, Greece
Petrica Pop, Technical University of Cluj-Napoca, Romania
Olympia Roeva, Bulgarian Academy of Sciences, Bulgaria
Patrick Siarry, Université Paris XII Val de Marne, France

Dominik Ślęzak, University of Warsaw and Infobright Inc., Poland
Stefan Stefanov, Neofit Rilski University, Bulgaria
Tomas Stuetzle, Université Libre de Bruxelles, Belgium
Ponnuthurai Suganthan, Nanyang Technological University, Singapore
Tami Tamir, The Interdisciplinary Center (IDC), Israel
Josef Tvrdik, University of Ostrava, Czech Republic
Michael Vrahatis, University of Patras, Greece
Roberto Wolfler Calvo, Université Paris 13, France
Antanas Zilinskas, Vilnius University, Lithuania

Contents

Finding Optimal Discretization Orders for Molecular Distance Geometry by Answer Set Programming

Douglas Gonçalves, Jacques Nicolas, Antonio Mucherino and Carlile Lavor

Abstract The Molecular Distance Geometry Problem (MDGP) is the problem of finding the possible conformations of a molecule by exploiting available information about distances between some atom pairs. When particular assumptions are satisfied, the MDGP can be discretized, so that the search domain of the problem becomes a tree. This tree can be explored by using an *interval* Branch & Prune (*i*BP) algorithm. In this context, the order given to the atoms of the molecules plays an important role. In fact, the discretization assumptions are strongly dependent on the atomic ordering, which can also impact the computational cost of the *i*BP algorithm. In this work, we propose a new partial discretization order for protein backbones. This new atomic order optimizes a set of objectives, that aim at improving the *i*BP performances. The optimization of the objectives is performed by Answer Set Programming (ASP), a declarative programming language that allows to express our problem by a set of logical constraints. The comparison with previously proposed orders for protein backbones shows that this new discretization order makes *i*BP perform more efficiently.

D. Gonçalves (✉)
CCFM, Universidade Federal de Santa Catarina, Florianopolis, Brazil
e-mail: douglas.goncalves@ufsc.br

J. Nicolas
INRIA, Rennes Bretagne Atlantique, Rennes, France
e-mail: jacques.nicolas@inria.fr

A. Mucherino
IRISA, University of Rennes 1, Rennes, France
e-mail: antonio.mucherino@irisa.fr

C. Lavor
IMECC-UNICAMP, Campinas, SP, Brazil
e-mail: clavor@ime.unicamp.br

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_1

1 Introduction

Experiences of Nuclear Magnetic Resonance (NMR) are able to estimate distances between some pairs of atoms of a given molecule [12]. The problem of finding the possible conformations of the molecule that are compatible with this distance information is known in the scientific literature as the Molecular Distance Geometry Problem (MDGP) [6, 12, 18]. Generally, the distance information is given through a list of lower and upper bounds on the distances, i.e. by a list of real-valued intervals [19]. The MDGP, by its nature, is a constraint satisfaction problem, which is NP-hard [24].

Let $G = (V, E, d)$ be a simple weighted undirected graph where the vertices in V represent the atoms of the molecule and $d : E \to \mathbb{R}_+$ assigns positive weights $d(i, j)$ to edges which are in E if the distance between the atoms i and j is available. The MDGP asks therefore to find an embedding $x : V \to \mathbb{R}^3$ satisfying the distance information, i.e. a conformation in the three-dimensional space such that:

$$\underline{d}(i, j) \leq \|x_i - x_j\| \leq \overline{d}(i, j), \quad \forall (i, j) \in E, \tag{1}$$

where $\underline{d}(i, j)$ and $\overline{d}(i, j)$ denote, respectively, the lower and upper bounds for the distance $d(i, j)$ (notice that $\underline{d}(i, j) = \overline{d}(i, j)$ if $d(i, j)$ is exact). In this paper, we suppose that the given set of distances is actually embeddable in $\mathbb{R}^3$; in other words, it is supposed that no distance is affected by errors.

Over the years, the solution of MDGPs has been attempted by formulating global optimization problems in a continuous space, where a penalty objective function is generally employed in order to measure the violation of the distance constraints for given molecular conformations. More recently, a new class of MDGPs has been introduced, for which the search domain of the optimization problem can be reduced to a discrete space having the structure of a tree. Instances belonging to this class can be solved by employing an *interval* Branch & Prune (*i*BP) algorithm [15, 17].

In order to perform the discretization, MDGP instances need to satisfy some particular assumptions. In practice, the atoms of the molecule must be sorted in a way that there are at least three *reference atoms* for each of them. We say that an atom z is a reference for another atom y when z precedes y in the given atomic order, and the distance $d(z, y)$ is known. In such a case, indeed, candidate positions for y belong to the sphere centered in z and having radius $d(z, y)$. When the *reference distance* $d(z, y)$ is given through a real-valued interval, a spherical shell centered in z can be instead defined. If three reference atoms are available for y, then candidate positions (for y) belong to the intersection of three Euclidean objects. The easiest situation is the one where the three available distances are exact, and the intersection gives, in general, two possible positions for y [14]. However, if only one of the three distances is allowed to take values into a certain interval, then the intersection gives two curves, generally disjoint, where sample points can be chosen [15]. In both situations, the discretization can be performed. More details about the discretization process are given in Sect. 2.

Necessary preprocessing step for solving MDGPs by this discrete approach is therefore the one of finding suitable atomic orders that allow each atom y to have at least three reference atoms. We refer to such orders as *discretization orders*. In previous works, discretization orders have been either handcrafted [5, 15], or designed by using a pseudo de Bruijn graph representation of the cliques in G [21], or even automatically obtained by a greedy algorithm [13, 20]. In the present paper, we propose an approach for selecting, among all discretization orders that might exist for a given MDGP instance, the ones that are able to improve *i*BP performances. To this aim, we define some objectives (with a given priority) to be optimized during the search for the discretization orders. These objectives, together with the discretization assumptions, define a multi-level optimization problem for finding optimal discretization orders. Although this optimization problem can be cast as a mathematical programming problem (see for example [4]), we consider instead in this work an Answer Set Programming (ASP) approach [7, 9]. ASP provides in fact more flexible tools for managing the constraints we need to deal with.

The rest of the paper is organized as follows. The basic idea behind the discretization and the sketch of the *i*BP algorithm are presented in Sect. 2. In Sect. 3, we present our ASP model for the generation of *partial orders* for the MDGP. Given one unique partial order provided as a result by the ASP solver, several total orders for the protein backbones can be extracted. By considering one of such total orders, we present in Sect. 4 some computational experiments where we compare the properties of this new order to those of previously proposed orders for protein backbones. Finally, Sect. 5 concludes the paper.

2 Discretization Orders and the *i*BP Algorithm

Let $G = (V, E, d)$ be a simple weighted undirected graph representing an instance of the MDGP. In order to perform the discretization, G needs to satisfy the following assumptions. Let $E' \subset E$ be the subset of edges for which the associated weight is an exact distance.

Definition 2.1 (*The* interval *Discretizable DGP in dimension 3* (i$DDGP_3$)) Given a simple weighted undirected graph $G = (V, E, d)$, we say that G represents an instance of the *i*DDGP$_3$ if and only if there exists an order relationship "$<$" on the vertices of V verifying the following assumptions:

(a) $\{1, 2, 3\}$ is a clique and $\{(1, 2), (2, 3), (1, 3)\} \subset E'$;
(b) $\forall i \in \{4, \ldots, |V|\}$, there exists a subset $ref(i) = \{i', i'', i'''\}$ such that

1. $i''' < i, i'' < i, i' < i$;
2. $\{(i'', i), (i', i)\} \subset E'$ and $(i''', i) \in E$;
3. $d(i', i''') < d(i', i'') + d(i'', i''')$.

We refer to orders satisfying (a) and (b) as "discretization orders". These orders can be either partial or total.

Assumption (a) allows us to fix the positions of the first three atoms, avoiding to consider congruent solutions that can be obtained by rotations and translations. Assumption (b.1) ensures the existence of the three reference atoms for every atom $i > 3$, and assumptions (b.2) ensures that at most one of the three reference distances is represented by an interval. We say that the reference distances are the ones used in the discretization process; additional distances that might be available can be used in the pruning process (see below). Finally, assumption (b.3) avoids the reference atoms to be co-linear. Note that assumption (b.3) cannot always be verified a priori, because some of the necessary distances may not be available (the corresponding edges may not be in E). However, this assumption can fail to be satisfied with probability 0 when considering either molecular instances or even generic graphs, and therefore we do not really need to verify it in advance [13, 22].

Under the assumptions (a) and (b), the MDGP can be discretized. The search domain becomes a tree containing, layer by layer, the possible positions for a given atom. In this tree, the number of branches increases exponentially layer by layer. After the discretization, the MDGP can be seen as a combinatorial problem.

The discretization assumptions strongly depend on the existence of an order for the vertices of G, i.e. for the atoms of the considered molecule. When working on molecules with known chemical structure such as proteins, some distance information is always available, because related to this chemical structure. In this case, orders, that are valid for an entire class of instances, can be obtained, where only "guaranteed" distance information is exploited. This kind of information includes bond lengths and angles, and also structural information, such as peptide plane induced distances [19]. In this work, we focus our attention on protein backbones, which consist of the set of atoms that, in proteins, are common to all amino acids (the side chains are not considered).

In previous works, discretization orders were identified by employing different approaches. In [5, 15], handcrafted orders were presented for the protein backbone and the side chains belonging to the 20 amino acids that can take part to the protein synthesis. More recently, in [21], orders were identified by searching for total paths on pseudo de Bruijn graphs containing cliques of the original MDGP graph G. In both cases, these orders were conceived for satisfying an additional assumption on the reference atoms: i''', i'', i' and i have to be consecutive (*consecutivity assumption*). Because of this additional assumption, the calculation of the atomic coordinates was possible by using the method described in [14], which is more stable than the general approach based on the solution of a sequence of quadratic systems (each for one sphere intersection). Successively, however, it was proved that the same efficiency and the same accuracy can be obtained by employing another method for which the consecutivity assumption does not have to be satisfied [10].

Another way to construct discretization orders is given by the greedy algorithm firstly proposed in [13] and subsequently extended for interval distances in [20]. This algorithm is able to find orders where the consecutivity assumption is not ensured. In fact, requiring the consecutivity assumption be satisfied makes the problem of finding a discretization order NP-hard [4]. A heuristic has also been proposed for finding discretization orders without consecutivity assumption, which outperformed

Algorithm 1 The iBP algorithm.

```
1: iBP(j, n, order, d, discr_factor)
2: if (j > n) then
3:   print current conformation;
4: else
5:   define ref(j) = {j', j'', j'''};
6:   if (d(j''', j) is a nondegenerate interval) then
7:     let [l, u] = d(j''', j); set N = discr_factor;
8:   else
9:     let l = u = d(j''', j); set N = 1;
10:  end if
11:  for (h = 1 ... N, h equally spaced distances in the interval [l, u]) do
12:    compute the candidate positions: x_j^h and x_j^{-h};
13:    if (x_j^h is feasible) then
14:      iBP(j + 1, n, order, d, discr_factor);
15:    end if
16:    if (x_j^{-h} is feasible) then
17:      iBP(j + 1, n, order, d, discr_factor);
18:    end if
19:  end for
20: end if
```

the greedy algorithm on large instances, but for which there are no guarantees of convergence [11].

This work finds its place in the context of this last approach for the identification of discretization orders without the consecutivity assumption. Our aim is not only to find orders that allow for the discretization, but also to *optimize* these orders in a way that the corresponding search domain (the tree) is easier to explore by the iBP algorithm. Section 3 provides details about the objectives that we optimize for improving the search on the tree.

Algorithm 1 is a sketch of the iBP algorithm [15]. The algorithm recursively calls itself for exploring the search tree. The *if* structure in line 6 discriminates between the two situations: there are three exact reference distances/only two of these distances are exact. In the first case, two atomic positions are generated by the sphere intersection, and two new branches are added to the tree at the current layer (see Introduction). In the second case, instead, the intersection gives 2 disjoint curves, from which sample points need to be taken for discretizing. The parameter of the algorithm named *discr_factor* gives in fact the (fixed) number of samples that are taken from each curve. The other parameters of the algorithm are: j, the current atom; n, the total number of atoms; *order*, the available discretization order; d, the set of weights on the edges of G. The discretization *order* allows to define, for each j, the set $ref(j) = \{j', j'', j'''\}$ of reference atoms for j.

After the computation of possible positions for the current atom j, the feasibility of these atomic positions are verified by applying what we call *pruning devices* (see lines 13 and 16 of Algorithm 1). Even if tree branches grow exponentially layer by layer, the pruning devices allow iBP to focus the search on the feasible

parts of the tree. The easiest and probably most efficient pruning device is the Direct Distance Feasibility (DDF) criterion [14], which consists in verifying the ε-feasibility of the constraints:

$$\underline{d}(k, j) - \varepsilon \leq ||x_k - x_j|| \leq \overline{d}(k, j) + \varepsilon, \quad \forall (k, j) \in E, \quad \text{with } k < j. \tag{2}$$

All distances related to edges (k, j), with $k < j$ and that are not used in the discretization, are named *pruning distances*, because they can be used by DDF for discovering infeasibilities. Several pruning devices can be integrated in *i*BP, that can be based on either pure geometric features of molecules, or rather on chemical and biological properties [3]. In this work, in order to focus the attention on the discretization orders, only the pruning device DDF will be considered in our computational experiments.

3 Looking for Optimal Orders for the Protein Backbone

Besides satisfying the assumptions in Definition 2.1, discretization orders can be optimized with the aim of improving the performances of the *i*BP algorithm. We propose in this work some objectives to be optimized during our search for discretization orders for protein backbones. These objectives are sorted with their priority levels: higher priority objectives are optimized first.

The protein backbone is defined by the sequence of atomic subgroups common to all amino acids, which are bonded together to form the protein chain [19]. Side chains are omitted in this work, and we consider proteins formed by only one chain. Only four different atoms compose protein backbones: carbons (C), nitrogens (N), hydrogens (H) and oxygens (O). We will use the symbol C_α for the "central" carbon of the amino acid, and we will use the symbol H_α for the hydrogen bonded to this C_α. Other atom types appear only once per amino acid, so that it is not necessary to use special characters when referring to them. Superscripts associated to atom's labels will denote the position of the amino acid to which the atom belongs in the protein chain. For example, H^i_α denotes the hydrogen bonded to C^i_α, in the amino acid at position i. The superscript "0" is associated to the second hydrogen H^0 belonging to the first amino acid; the superscript "$n + 1$" is instead associated to the second oxygen belonging to the last amino acid, where n is the total number of amino acids.

Recall that, when three exact reference distances are available for the discretization, there are two possible positions for the current atom. Otherwise, when one of such distances is instead represented by an interval, $2 \times$ *discr_factor* possible candidate positions for the atom can be computed (see Sect. 2).

Our list of objectives is given below, together with their priority levels. Together with the assumptions (a) and (b) in Definition 2.1, these objectives define a multi-level optimization problem.

4. *Maximize the number of exact distances used in the discretization.*
   In order to keep the tree width as small as possible, the situation where all three

reference distances used in the discretization are exact is preferable. This way, the number of branches on each layer can be at most the double of the number of branches on the previous one.

3. *Maximize the number of distances* (H_α^i, H^{i+1}) *and* (H^i, H_α^i) *used in the discretization.*
   The interval distances (H_α^i, H^{i+1}) and (H^i, H_α^i) can always be estimated by studying the structure of protein backbones. Even if represented by real-valued intervals, these distances can be employed for the discretization. Moreover, it is likely NMR experiments can provide a better estimate for them, i.e. sharper intervals [16].
2. *Postpone the interval distances used in the discretization.*
   When an interval distance is used in the discretization process, potentially $2 \times discr\_factor$ new branches are added to the tree at the current layer, while, at most, only two branches need to be added when all distances are exact. Therefore, if an interval distance is used for discretization at deeper layers of the tree, this phenomenon can be delayed, so that fewer internal nodes in the search tree are generated, and this may favor early pruning.
1. *Anticipate the pruning distances.*
   Pruning distances are employed for verifying the feasibility of generated candidate positions. The sooner they appear, the earlier infeasible positions can be discovered and branches of the tree can be pruned. Since the set of pruning distances is instance-dependent, it is not possible to find orders where this objective is optimal for an entire class of instances. Therefore, we only consider NMR distances that are likely to be available in every protein instance: the distance between pairs of hydrogens (H^i, H^{i+1}) and $(H_\alpha^i, H_\alpha^{i+1})$ belonging to consecutive amino acids [16]. Note that, since there are two hydrogens bonded to N^1 in the first amino acid, the distances related to both hydrogens are supposed to be known.

In order to consider the above objectives, we discriminate among: (i) exact distances; (ii) generic interval distances; (iii) interval distances between hydrogen pairs (H_α^i, H^{i+1}) and (H^i, H_α^i); (iv) pruning distances. While looking for optimal orders, we will fix the length of the protein backbone to three. In fact, the regular structure of the protein backbone will allow us to easily extend the obtained orders to protein backbones of any length.

4 Computational Experiments and Discussion

We present and discuss in this section two sets of computational experiments. First of all, by using ASP, we generate an optimal partial order for which the objectives detailed in Sect. 3 are optimized (see Sect. 4.1). Secondly, we extract one total order from the optimal partial one, and we compare the performances of the *i*BP algorithm

while using this new order and some previously proposed ones (see Sect. 4.2). Both experiments will be commented in details.

4.1 Finding an Optimal Partial Order by ASP

We use the ASP framework to find solutions to the multi-level optimization problem described in Sect. 3. ASP is a form of declarative programming adapted to combinatorial and optimization problems [2], which offers a language of clauses to express constraints on given variables. Once all constraints are expressed as logical formulas, a grounder transforms such constraints into a (large) set of Boolean equations, while a solver looks for possible models (possible *answers*) for this set. From the set of obtained answers, solutions to the initial problem can be derived.

To the four objectives given in Sect. 3, we also added the one of minimizing the number of ranks in the partial order. This actually makes the searched order a *partial order*, where the same rank can be associated to more than one atom. We point out that the expressiveness and efficiency of ASP modeling allowed us to test multiple objective functions and to converge towards a single solution with a few simple objectives. In other words, the list of objectives presented above was tested by ASP and refined several times before defining the current list (given in Sect. 3) for which only one partial order is given as optimal by ASP. In our experiments, we employ the grounder Gringo and the solver Clasp developed in Potsdam University [8].

We focus our attention on the protein backbone of a short polypeptide consisting of three amino acids (without side chains). The regular structure of protein backbones allows to extend any solution found for three amino acids to protein backbones of any length. There is only one slight variation on the backbone in presence of one single amino acid: the proline. This amino acid has no hydrogen H bonded to its nitrogen N.

Before presenting the results obtained by ASP, we briefly present two previously proposed orders for protein backbones. In [15], the following handcrafted order was introduced:

$$\begin{array}{c} |N^1, H^1, H^0, C_\alpha^1, N^1, H_\alpha^1, C_\alpha^1, C^1|N^2, C_\alpha^2, H^2, N^2, C_\alpha^2, H_\alpha^2, C^2, C_\alpha^2| \\ |N^3, C^2, C_\alpha^3, H^3, N^3, C_\alpha^3, H_\alpha^3, C^3, C_\alpha^3, O^3, C^3, O^4|. \end{array} \tag{3}$$

Recall that superscripts are used for referring to the amino acid the atom belongs to. Recall also that H^0 refers to second hydrogen belonging to the first amino acid and bonded to N^1, and that O^4 is one of the oxygens belonging to the last amino acid. Repetitions are allowed in this order, because of the consecutivity assumption (see Sect. 2). Repetitions increase MDGP instances in length because atoms can appear in the order more than once; this, however, does not imply any increase in complexity, because there is no branching in *i*BP when copies of already placed atoms are considered. The symbol "|" is used for separating the amino acids. Since there

are repetitions, however, some atoms belonging to one amino acid can be repeated in the next one.

In [20], another order was generated by using a greedy algorithm:

$$|N^1, H^1, H^0, C_\alpha^1, H_\alpha^1, C^1|N^2, H^2, C_\alpha^2, H_\alpha^2, C^2|N^3, H^3, C_\alpha^3, H_\alpha^3, C^3, O^3, O^4|. \tag{4}$$

Repetitions are not necessary in this order and the consecutivity assumption is not satisfied (see Sect. 2). The employed greedy algorithm iteratively constructs the order by placing at the current position the atom that has at least three reference atoms and for which the total number of references is maximized. When it exists, the greedy algorithm is able to find a discretization order for a given instance.

Our ASP execution provided us with one unique solution: the partial order is depicted in Fig. 1. For the 18 atoms of our 3-amino acid protein backbone, the greatest rank is 13 in this partial order. There are in fact clusters of atoms where the relative positioning of the atoms is irrelevant. In the initial amino acid, there is a cluster with 4 atoms; in the other amino acids, the positions in the order for the atoms N^i and C_α^i can be exchanged. When considering three amino acids, there are therefore 96 ($4! \cdot 2! \cdot 2!$) total orders corresponding to the partial order in Fig. 1.

In order to verify whether these 96 total orders have similar properties, we generated 96 iDDGP$_3$ instances related to the 3-amino acid backbone used in the ASP simulation. For all of them, we ran the iBP algorithm and monitored the total number of iBP recursive calls necessary to explore the entire tree. This analysis showed that all generated total orders are equivalent (same number of solutions, same number of iBP calls). Note that, even if the theoretical number of solutions should be order-independent, the branching over intervals currently implemented in iBP (see Algorithm 1) could lead to different numbers of *found* solutions when using different orders. Nevertheless, we did not have any variation for any instance of our set.

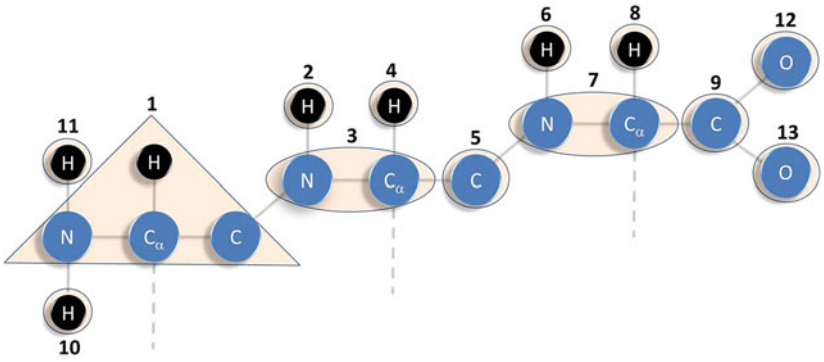


Fig. 1 A partial order, for the discretization of protein backbones, found by our ASP model

In order to perform computational experiments on realistic protein instances (see Sect. 4.2), we selected, among the 96 total orders that can be extracted from the partial order in Fig. 1, one total order:

$$|N^1, C^1_\alpha, H^1_\alpha, C^1|H^2, N^2, C^2_\alpha, H^2_\alpha, C^2|H^3, N^3, C^3_\alpha, H^3_\alpha, C^3|H^0, H^1, O^3, O^4|. \tag{5}$$

In this order, the two hydrogens H^0 and H^1 bonded to N^1, as well as the two oxygens O^3 and O^4 beloging to the last amino acid, are placed at the end of the order. As it is easy to remark, the specific orders for the second and the third amino acids are identical. For backbones composed by a longer list of amino acids, it is necessary to repeat this generic order as many times as the total number of amino acids in the molecule. In the special case of proline, where the hydrogen bonded to N^i, with $i > 1$, does not appear, we can just omit H^i in the order. This change does not make the order non-discretizable.

4.2 Comparison to Other Orders

We present some experiments on a subset of realistic protein instances where, for each instance, we consider the three orders (3), (4), and (5), and we compare the performances of *i*BP. The *i*BP algorithm was implemented in C programming language (GNU C compiler v.4.0.1 with `-O3` flag), and the executions were carried out on an Intel Core 2 Duo @ 2.4 GHz with 2GB RAM, running Mac OS X.

The instances that we consider in our experiments have been generated as follows. We consider a subset of proteins from the Protein Data Bank (PDB) [1] related to human immunodeficiency. Together with the PDB coordinates of the atoms, we consider the chemical structure of the protein, i.e. information about its bond lengths and angles. All distances between atom pairs are computed, and a distance is included in our instances if it is between

1. two bonded atoms (considered as exact);
2. two atoms that are bonded to a common atom (considered as exact);
3. two atoms belonging to a quadruplet of bonded atoms forming a torsion angle (considered as an interval);
4. two hydrogen atoms (considered as an interval, if the distance lies in the interval [2.5, 5] Å).

The first three items are related to the chemical structure of the molecule; only the last item concerns distances that simulate NMR data. The distances that are related to item 3 are generally intervals; however, one of the possible torsion angles is related to the peptide bond between two amino acids: in such a case, the distance is considered as exact, because the peptide bond forces all atoms to lie on the same plane. Interval distances coming from torsion angles are computed so that all corresponding

Table 1 Some properties of the instances that we generated from PDB files related to human immunodeficiency

Instance	1niz	2jnr	2pv6	1zec	2m1a	2me1	2me4	1dsk
$\|V\|$	69	98	112	123	132	137	137	142
$\|E\|$	333	460	570	628	695	700	694	746
$\|E'\|$	186	263	307	336	363	377	377	391

values for the torsion angle are allowed. The interval distances related to item 4 have instead width equal to 2 Å, and their bounds were generated so that the *true* distance is randomly placed inside the interval. After the calculation of the distance information, the atoms in every instance have been reordered by considering three total orders: the handcrafted order (3), the greedy order (4), and our new order (5). Table 1 contains some information about the generated instances.

Table 2 presents a comparative analysis of the three orders. The table provides the minimum value for $discr\_factor$ (d_f in the table) necessary to obtain at least one solution with the *i*BP algorithm, the total number of *i*BP calls, and the CPU time, in seconds. In order to obtain the minimum value for the discretization factor, we gradually increased $discr\_factor$ from 3 to 9, until one solution was found in less than 30 s. In these experiments, we considered $\varepsilon = 0.1$ in the feasibility test (see Eq. (2)) given by DDF. All found solutions are "good-quality" solutions: we considered as index for measuring the solution quality the LDE, which ranges between 10^{-5} and 10^{-4} in our solutions. The reader can refer to [10] for the definition of the LDE index.

As the table shows, the total order that we found in this work is generally the optimal one (in terms of both *i*BP calls and CPU time, while the quality of the solutions remains unchanged). There are a few examples, however, where the performances of *i*BP are better when using the other orders. This can be due to the fact that sample points are chosen during the discretization from the curves that are generated in presence of interval data (see Sect. 2). Not considering entire curves, but some sample points only, makes *i*BP a heuristic, whose performances can drastically vary on the basis of the sample points that are chosen. Nevertheless, the experiments in Table 2 show that the order (5) is almost always the best one.

In order to validate this result, we performed all experiments for a second time, and we replaced the deterministic choice on line 11 of Algorithm 1 for the selection of the sample distances with a random choice of ($discr\_factor =$) 5 uniformly distributed samples on the two obtained curves. Table 3 shows the average performances of *i*BP over 10 runs. Only the runs taking less than 10 s are considered in the average: S denotes the number of considered runs. This table shows a more regular pattern for the greedy order (4) and our new order.

Table 2 Comparison among three total discretization orders

	Order (3)			Order (4)			Order (5)		
	d_f	iBP calls	Time (s)	d_f	iBP calls	Time (s)	d_f	iBP calls	Time (s)
1niz	6	6602398	10.77	**6**	**353052**	**0.49**	6	1180273	3.12
2jnr	3	32663	0.06	**5**	**17337**	**0.02**	5	44403	0.10
2pv6	5	1342941	2.49	6	376608	0.51	**5**	**47903**	**0.11**
1zec	7	5943667	12.20	6	821970	1.47	**5**	**41291**	**0.15**
2m1a	–	–	–	5	204475	0.35	**6**	**38572**	**0.11**
2me1	9	3568317	6.35	5	1667991	2.36	**5**	**135574**	**0.36**
2me4	6	974913	1.94	5	1213948	2.11	**4**	**281922**	**0.81**
1dsk	7	10098038	19.16	6	2186234	2.88	**6**	**365728**	**1.41**

d_f is $discr\_factor$ in Algorithm 1. The last column gives the performances of iBP for the new optimal order. Experiments lasting more than 30 s are considered as unsuccessful (details not reported)

Table 3 Experiments with *i*BP and the random selection of the samples from the intervals

Instance	Order (3)		Order (4)		Order (5)	
	S	Avg. Time (s)	S	Avg. Time (s)	S	Avg. Time (s)
1niz	8	1.06	7	0.84	**10**	**0.27**
2jnr	9	0.71	**10**	**0.43**	10	0.99
2pv6	10	0.92	10	0.11	**10**	**0.07**
1zec	7	4.63	10	0.91	**10**	**0.23**
2m1a	9	4.38	10	0.55	**10**	**0.14**
2me1	5	2.84	10	0.86	**10**	**0.25**
2me4	10	1.63	10	0.09	**10**	**0.05**
1dsk	7	3.48	10	0.47	**10**	**0.44**

Experiments running more than 10 s are not considered in the computation of the average time

5 Conclusions

We proposed a new discretization order for distance geometry with interval data that allows for the discretization of protein backbones. This order comes from a complete search of the space of possible discretization orders, implemented through an ASP model. This model ensures that the assumptions for the discretization are satisfied, and that some additional objectives (aimed at improving *i*BP performances) are optimized. Priorities were assigned to the considered objectives. We compared the proposed discretization order to two orders that were previously proposed in the literature [15, 20]. Our computational experiments on protein instances showed that *i*BP performances actually improve when using the new proposed order.

Future works will be devoted to strategies for reducing the degrees of freedom of our protein backbones. In general, there are two (continuous) degrees of freedom per amino acid (on the protein backbone), which can be modeled by using two torsion angles ϕ and ψ [19]. One possibility for an improvement is to explore the *favorable* regions of the so-called Ramachandran map to reduce the discretization intervals in length [23]. Moreover, an interesting property of the new proposed order is that it allows to easily set the reference atoms so that the *planarity* of the peptide plane and the *chirality* around the carbons C_α can be exploited [3]. As a consequence, this new order could be used for branching only over interval distances (the planarity and the chirality properties allow to uniquely position an atom when three reference distances are exact). We point out that this feature of the new order was not considered in the experiments presented in this paper because the two older orders (3) and (4) do not allow for an easy implementation of a method based on this feature.

Finally, notice that this work can be extended to protein *side chains*. Side chains generally have constrained structures and preferred configurations, for which optimal discretization orders might be identified. Subject of future research is also the problem of finding the optimal discretization orders that consider the chemical structure of

the whole molecule *at once*, as well as the entire set of pruning distances that might be available (obtained by NMR).

Acknowledgments We are thankful to the Brittany Region (France) and to the Brazilian research agencies FAPESP, CNPq for financial support.

References

1. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne, The protein data bank. Nucleic Acid Res. **28**, 235–242 (2000)
2. G. Brewka, T. Eiter, M. Truszczyński, Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011)
3. A. Cassioli, B. Bardiaux, G. Bouvier, A. Mucherino, R. Alves, L. Liberti, M. Nilges, C. Lavor, T.E. Malliavin, An algorithm to enumerate all possible protein conformations verifying a set of distance restraints. BMC Bioinform. (2015, to appear)
4. A. Cassioli, O. Gunluk, C. Lavor, L. Liberti, Discretization vertex orders in distance geometry. Discrete Appl. Math. (2015, to appear)
5. V. Costa, A. Mucherino, C. Lavor, A. Cassioli, L.M. Carvalho, N. Maculan, Discretization orders for protein side chains. J. Glob. Optim. **60**(2), 333–349 (2014)
6. G.M. Crippen, T.F. Havel, *Distance Geometry and Molecular Conformation* (Wiley, New York, 1988)
7. T. Eiter, G. Ianni, T. Krennwallner, Answer set programming: a primer. Reason. Web **5689**, 40–110 (2009)
8. M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: from theory to practice. Artif. Intell. **187**, 52–89 (2012)
9. M. Gelfond, *Answer Sets*, Handbook of Knowledge Representation, Chapter 7 (Elsevier, Amsterdam, 2007)
10. D.S. Gonçalves, A. Mucherino, Discretization orders and efficient computation of Cartesian coordinates for distance geometry. Optim. Lett. **8**(7), 2111–2125 (2014)
11. W. Gramacho, D. Gonçalves, A. Mucherino, N. Maculan, A new algorithm to finding discretizable orderings for distance geometry, in *Proceedings of Distance Geometry and Applications (DGA13)*, Manaus, Amazonas, Brazil, pp. 149–152 (2013)
12. T.F. Havel, in *Distance Geometry*, Encyclopedia of nuclear magnetic resonance, ed. by D.M. Grant, R.K. Harris (Wiley, New York, 1995), pp. 1701–1710
13. C. Lavor, J. Lee, A. Lee-St.John, L. Liberti, A. Mucherino, M. Sviridenko, Discretization orders for distance geometry problems. Optim. Lett. **6**(4), 783–796 (2012)
14. C. Lavor, L. Liberti, N. Maculan, A. Mucherino, The discretizable molecular distance geometry problem. Comput. Optim. Appl. **52**, 115–146 (2012)
15. C. Lavor, L. Liberti, A. Mucherino, The interval branch-and-prune algorithm for the discretizable molecular distance geometry problem with inexact distances. J. Glob. Optim. **56**(3), 855–871 (2013)
16. C. Lavor, A. Mucherino, L. Liberti, N. Maculan, On the computation of protein backbones by using artificial backbones of hydrogens. J. Glob. Optim. **50**(2), 329–344 (2011)
17. L. Liberti, C. Lavor, N. Maculan, A branch-and-prune algorithm for the molecular distance geometry problem. Int. Trans. Oper. Res. **15**, 1–17 (2008)
18. L. Liberti, C. Lavor, N. Maculan, A. Mucherino, Euclidean distance geometry and applications. SIAM Rev. **56**(1), 3–69 (2014)
19. T.E. Malliavin, A. Mucherino, M. Nilges, Distance geometry in structural biology: new perspectives, in *Distance Geometry: Theory, Methods and Applications*, ed. by A. Mucherino, C. Lavor, L. Liberti, N. Maculan (Springer, Berlin, 2013), pp. 329–350

20. A. Mucherino, On the Identification of Discretization Orders for Distance Geometry with Intervals, Lecture Notes in Computer Science 8085, in *Proceedings of Geometric Science of Information (GSI13)*, ed. by F. Nielsen, F. Barbaresco, Paris, France, 2013, pp. 231–238
21. A. Mucherino, A Pseudo de Bruijn Graph Representation for Discretization Orders for Distance Geometry, Lecture Notes in Computer Science 9043, Lecture Notes in Bioinformatics series, ed. by F. Ortuño, I. Rojas, *Proceedings of the 3rd International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO15)*, Part I, Granada, Spain, 2015 pp. 514–523
22. A. Mucherino, C. Lavor, L. Liberti, The discretizable distance geometry problem. Optim. Lett. **6**(8), 1671–1686 (2012)
23. G.N. Ramachandran, C. Ramakrishnan, V. Sasisekharan, Stereochemistry of polypeptide chain conformations. J. Mol. Biol. **7**, 95–99 (1963)
24. J.B. Saxe, Embeddability of weighted graphs in k-space is strongly NP-hard, *Proceedings of 17th Allerton Conference in Communications, Control and Computing*, 480–489 (1979)

Estimation of Edge Infection Probabilities in the Inverse Infection Problem

András Bóta, Miklós Krész and András Pluhár

Abstract Several methods have been proposed recently to estimate the edge infection probabilities in infection or diffusion models. In this paper we will use the framework of the Generalized Cascade Model to define the Inverse Infection Problem—the problem of calculating these probabilities. We are going to show that the problem can be reduced to an optimization task and we will give a particle swarm based method as a solution. We will show, that direct estimation of the separate edge infection values is possible, although only on small graphs with a few thousand edges. To reduce the dimensionality of the task, the edge infection values can be considered as functions of known attributes on the vertices or edges of the graph, this way only the unknown coefficients of these functions have to be estimated. We are going to evaluate our method on artificially created infection scenarios. Our main points of interest are the accuracy and stability of the estimation.

1 Introduction

The study of infection processes has roots in two seemingly different fields of research: sociology and the medical sciences. In the latter, it was used to model the spread of epidemics [10]. Applications focused on prevention, and the identification of the "choke points" during an epidemic. In the former, the spreading of information or opinions came into focus. One of the earliest models in sociometry, Granovetter's Linear Threshold [13] model is still considered to be a viable description of information diffusion.

A. Bóta (✉) · A. Pluhár
Institute of Informatics, University of Szeged, P. O. Box 652, Szeged 6701, Hungary
e-mail: bandras@inf.u-szeged.hu

A. Pluhár
e-mail: pluhar@inf.u-szeged.hu

M. Krész
Gyula Juhász Faculty of Education, University of Szeged, Boldogasszony Sgt. 6,
Szeged 6701, Hungary
e-mail: kresz@jgypk.u-szeged.hu

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_2

In economics, Domingos and Richardson developed the Independent Cascade model (IC) [11] for the purpose of viral marketing. They proposed the influence maximization problem, that is to find the set of k initial infectors for any k that results in the largest expected infection. Kempe et al. [16, 17] proved the influence maximization problem was NP-hard, proposed a greedy algorithm for it, and also showed that the generalization of the IC model is in fact an equivalent of the Linear Threshold model. They also used random simulations to approximate the vertex infection probabilities, and they choose an arbitrary constant for edge infection probabilities. This result stresses the importance of the exact computation of vertex infection probabilities. This problem was proven to be #P-complete by Cao et al. [6].

Computing maximal infections or the exact probabilities of infection with any kind of model requires a weighted network, that is the edge infection probabilities must be available. This information is usually not known beforehand. In most real-life applications, the edges are considered to be some constant, or estimated using intuition guided trial-and-error methods based on known edge or vertex attributes. Recently, a few papers were published in this topic discussing systematic approaches for the estimation of edge infection probabilities. In some of them [12, 18], the steps or iterations of the infection process are assumed to be known, which is realistic in the case of twitter or blog-based networks.

In this paper we are going to propose a different approach to the problem. Our approach uses the framework of the Generalized Cascade (GC) model [3] to define the Inverse Infection Problem, the task of estimating the edge infection probabilities in diffusion or infection models. The problem can be reduced to an optimization task and a Particle Swarm based meta-heuristic is proposed to solve it. The Inverse Infection Problem and its solution is different from the previously mentioned approaches because it does not require information on the individual steps of the infection process. Instead it builds on the probabilities of infection of vertices both before and after the process itself.

In order to reduce the complexity of the task and allow the algorithm to handle large networks an additional simplification will be proposed. If additional information is available on both the vertices and edges of the graph in the form of attributes, attribute functions can be used to calculate the edge infection probabilities from the attributes themselves. The functions have unknown coefficients, and the task of the optimization is the estimation of these coefficients. It is also worthwhile to independently estimate the infection probability of each edge, although this is only possible in small graphs. This way we can shed more light on the behavior of the problem and we can explore the boundaries of the optimization algorithm. The usefulness of the Inverse Infection Problem has been proven in the recent publication of the authors in [1].

Based on the good results of this method in applications, our goal in this paper is to provide a solid foundation to the Inverse Infection Problem in a more controlled environment. The paper is constructed as follows. In the next section we will give a short introduction into infection mechanisms, define the GC model, the Inverse Infection Problem and its two variants: direct and coefficient estimation. In Sect. 3 we will describe several ways to accurately estimate the infection probabilities, including gradient-based methods and Particle Swarm Optimization. We will

evaluate the performance of the optimization method on artificially generated infection scenarios in Sect. 4. The stability and the accuracy of the optimization will be examined, we will review the available infection heuristics and we will also identify the minimum number of patterns required to accurately estimate the infection probabilities. A preliminary version of this paper appeared in the proceedings of the Federated Conference on Computer Science and Information Systems [4].

2 Problem Definition

The process of infection takes place on a graph G, where $V(G)$ denotes the set of vertices, and $E(G)$ denotes the set of edges. While most traditional models require directed edges, depending on the application, they can be easily modified to handle undirected ones. We also need to know the edge infection probabilities, that is a weight $w_e \in [0, 1]$ for each edge e.

The notion of states is important. Each vertex of the network has a state of infection. The number of states and the transitions between them are governed by the specific model. One of the most basic approaches, the SIR model, [10] has three states: Susceptible, Infected and Recovered. Infected nodes infect susceptible ones, but after a certain period, which is usually a parameter of the model, they may recover, no longer infectious. Models in epidemics have a variety of states and the transitions between them are often more complicated. Models in economics or models describing information diffusion can be considered simpler. In the case of the Independent Cascade model, there are three states loosely corresponding to the ones in the SIR model and the infection period only lasts for one iteration. These three states are: susceptible, just infected (and still infectious), infected (but no longer infectious).

Most infection processes are also iterative, that is the process takes place in discrete time steps. Those models, that allow nodes to become susceptible again some time after becoming infected, may not terminate. It is easy to see, that the IC model terminates in finite steps.

2.1 Infection Models

Any infection model can be described as a process, that has two inputs: the first one is a weighted graph, where the edge weights are probabilities. The second input is the set of initial infectors $A_0 \subset V(G)$. These nodes are considered as infected at the beginning of the process. The process terminates at iteration t, and results in the set of infected nodes $A = \bigcup_{i=0}^{t} A_i$.

The specific way one vertex infects another varies depending on the model. In the case of the IC model [11], let $A_i \subseteq V(G)$ be the set of nodes newly activated in iteration i. In the next iteration $i + 1$, each node $u \in A_i$ tries to activate its inactive neighbors $v \in V \setminus \cup_{0 \le j \le i} A_j$ according to the edge infection probability $w_{u,v}$, and v

becomes active in iteration $i+1$, if the attempt is successful. If more than one node is trying to activate v in the same iteration, the attempts are made independently of each other in an arbitrary order within iteration $i+1$. If $A_t = \emptyset$, the process terminates in iteration t. It is easy to see, that the process always terminates in a finite number of iterations.

2.2 Generalized Cascade Model

Following the works of Bóta et al. [3], we can generalize this model in the following way. Instead of using vertex sets for representing the initial infectors, we work with two probability distributions. The *a priori* distribution defines the probability, that a vertex becomes infected on its own, independently of other vertices at the beginning of the process. The *a posteriori* defines the probability, that a vertex becomes infected at the end of the process. For all vertices $v \in V(G)$, we will denote the a priori probability of infection as p_v, the a posteriori as p'_v.

In some applications, an estimate of one or both of the above described probability distributions is available. For example, in the case of the banking application [1, 5] an accurate estimation of the probability of default for each company was given by standard models used by the bank.[1] Another application in telecommunications uses estimations for the probability of churn using similar methods. If such estimations are not available we can resort to a crude but effective method. Suppose we can observe the beginning and the end of the infection process k times. By counting the frequencies of infection, for all vertices v, how many times did v belong to A_0 or A we can construct the respective probability distributions. The accuracy of the estimation obviously depends on k, but k does not have to be a large number. We will show in Sect. 4, that 6–8 observations are enough to produce outputs with acceptable quality.

Based on these remarks and formulations, we can define the Generalized Cascade model [2]:

The Generalized Cascade Model: *Given an appropriately weighted graph G and the a priori infection distribution p_v, the model computes the a posteriori distribution p'_v* for all $v \in V(G)$.

The infection process itself is the IC model, although other models might also be used for different applications. We have chosen the Independent Cascade model as the basis of our method, because it performs well in modeling infection-like processes in business applications [5]. Alternatively, this model can be considered as a general framework of infection.

Unfortunately, the computation of the a posteriori distribution in the IC model is #P-complete. There are several existing heuristics to provide estimations of p'_v [7, 8], including the ones the authors proposed in [2]. Two of these are Monte Carlo based simulations. *Complete Simulation* is a direct adaptation of the idea of Kempe

[1] The BASEL II default probabilities were computed using vertex attributes.

et al. to the framework of the GC model. The basis of the idea is the notion of reachability. By selecting the edges $(u, v) \in E(G)$ independently of each other according to their infection probabilities $w_{u,v}$, they construct an unweighted graph which is a realization of the infection process. Any vertex, that can be reached from any initially infector is considered to be infected. We can adapt this process into the GC model by computing a large number of individual runs of the model and counting the frequencies of infections (both a priori and a posteriori). The process has an unfortunate property: the frequency (or sample size) must be high enough to reduce the standard deviation characteristic of Monte Carlo based methods. The *Edge Simulation* method decreases the standard deviation of the previous method. In each run, a subgraph containing all of the vertices able to infect the individual vertex v is constructed for all vertices $v \in V(G)$. This way the a posteriori infection of v can be computed directly in each run. The results of individual runs are averaged.

The authors have proposed two additional heuristics: In the *Neighborhood Bound Heuristic* a tree is constructed from the 2-neighborhood of a given vertex v representing all possible routes of infection. Both the tree and the a posteriori infection of v can be computed in a short time, resulting in a very fast heuristic. The *Aggregated Linear Effect* model is a linear approximation of the mechanism of the IC model. A more detailed description of these methods can be found in [2].

2.3 Inverse Infection Problem

Based on the framework of the Generalized Cascade model, we can define the Inverse Infection Problem.

Inverse Infection Problem: *Given an unweighted graph G, the a priori and the a posteriori probability distributions p_v and p'_v, compute the edge infection probabilities w_e for all $e \in E(G)$.*

If we are looking for each edge weight independently, $|E(G)|$ severely limits the applicability of the method. It is possible to optimize for several thousand different values, but a graph with this amount of edges only fits into the small category. Even so, these small examples might provide us with important information about the problem and its resolvability.

To make the problem tractable for large real-life networks we are going to suggest simplification of the problem. In many real-life applications the probability of infection between vertices can be considered as a combination of some properties of the edges, vertices and the graph itself. We are going to take advantage of this fact to make the problem solvable on large graphs in reasonable time. We are going to assume, that on each edge there are several *attributes*,[2] and the infection probability of the edge is a parametrized *function* of these attributes.[3]

[2] Vertex attributes can be easily converted into edge attributes.

[3] It is possible, that some of these attributes have no influence on the infection probability, but we expect the method to ignore the effect of these.

If there is only one attribute, this function can be expressed as $f(a_1(e))$, where f is the attribute function and $a_1(e)$ represents the attribute of edge e. We have used low-degree polynomials or simply a linear functions like $c_0 + c_1 a_1(e)$, where c_0 and c_1 are unknown coefficients. The degree of these polynomials should be low, but we allow different polynomials on different edges, with possibly different degrees. If the maximum degree of these polynomials is f_{max}, then there are at most $(f_{max} + 1)\ell$ coefficients to estimate. If there are multiple attributes it is necessary to summarize the effect of them, and even in the case of a single one, normalization is required to get valid probability values between zero and one. Therefore it is necessary to use two functions, the attribute function is applied to the individual attributes on each edge, then the results of these functions are summarized and normalized: $w_e = g(f(a_1(e)), f(a_2(e)), \ldots, f(a_n(e)))$, where w_e is the edge weight of e, g is the summarizer-normalizer function, f denotes the attribute function and $a_i(e)$ represents the i-th attribute of edge e. The attribute and the normalizer function is the same for all edges, this way we only have to estimate the *coefficients* of these functions, and since the number of attributes and coefficients is limited, the problem becomes tractable.

According to the above we are going to define two variants of the problem.

1. The *direct estimation* of edge weights.
2. The attribute function *coefficient estimation*.

We are going to evaluate the properties of both variants in the results section.

3 Estimation with Learning Methods

To provide a solution for the Inverse Infection Problem, we have developed the following learning algorithm. The problem definition states, that the a posteriori distribution is required as an input of any algorithm. In the case of a learning algorithm, it is considered as a test or reference dataset. By choosing some initially random edge weights or coefficients and taking the a priori distribution we can compute an estimation of the a posteriori distribution. Then, an error function calculates the difference between the reference set and the newly calculated infection values. Our goal is finding the global minimum of this error function by repeatedly adjusting the edge weights or coefficients and computing a new estimation of the a posteriori distribution. This is a typical task for global optimization, i.e. finding the minimum of an unknown multidimensional surface, where the points of this surface can be accurately estimated.

3.1 Previous Approaches and Experiences

We have tried several optimization algorithms, including more simple ones, like grid search, gradient-based methods and meta-heuristics. Our first analysis was performed

on banking data where we used grid search for optimization and the attribute formulation discussed above. While this early approach provided some results [5], it was clear that the search method was unable to solve but the most trivial tasks. Later, we have implemented a multi agent gradient based method, and compared the performance of it with our previous results [3]. The gradient method provided more accurate estimations, but highlighted several unfortunate properties of the problem itself.

Perhaps the most obvious problem is the dimensionality. Even if we consider coefficient estimation, the number of attributes and thus the number of parameters to estimate is too large for grid search to handle.

Another observation was that the error function was noisy. This comes from using Monte Carlo methods to approximate the IC model, since the deviation of these simulations makes different runs with the same coefficient values have different results. The noise can be reduced by increasing the frequency parameter of the simulation, but this also increases the time complexity of the method [3].

The third observation was that the problem is underdetermined. Different edge weight configurations can result in the same infection pattern, the same a posteriori distribution. This results in alleys and plateaus on the error surface. In the case of the example on Fig. 1, the global minimum is in the middle of the alley. Even in this simple example neither algorithms are able to reliably find the best solution.

Grid search had serious performance issues both in finding the global minimum and in time complexity. Due to its search pattern, its precision is simply not enough to tackle with this problem, and it also scales exponentially with the number of coefficients. The simple gradient method also performs poorly: it easily gets lost on the alleys and plateaus especially if they are noisy as well. As a consequence, it rarely finds a solution close to the global minimum, and the number of steps it takes to find a solution at all can be quite high.

We have tried several error functions, mainly vector distance measurements, and ROC evaluation. One of our first experiences was, that the latter is not enough to properly guide the optimization method to the global minimum, so we have shifted our attention to other measurements, and finally settled on the root mean squared error. In this work we are going to use the RMSE as an error measurement, that is we are looking for the minimum of

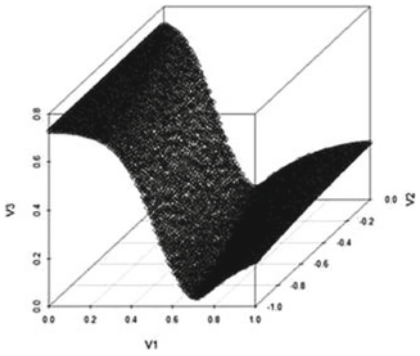


Fig. 1 The error surface of an IIP with one edge attribute and $f_1(a_1) = c_0 * a_1 + c_1$ as the attribute function. Root mean squared error was used for the evaluation

$$\sqrt{\frac{1}{|V(G)|}\sum_{v\in V(G)}(\hat{\mathbf{p}}'_v - \mathbf{p}'_v)^2}, \tag{1}$$

where $\hat{\mathbf{p}}'_v$ denotes the estimated a posteriori infection of vertex v.

3.2 Particle-Swarm Optimization

In order to handle the above mentioned problems, we have decided to implement the particle swarm optimization algorithm of Kennedy [14]. This is an iterative method based on the interaction of multiple agents or "particles". Each agent corresponds to a different edge weight or coefficient configuration, representing a coordinate in the parameter space of the problem.

Apart from the coordinates themselves, the agents also have a velocity. In each iteration the position of an agent is updated by adding its velocity to it. The velocity of the agent is computed using the best solution the agent has found and the best solutions of the neighboring agents; the goodness of the solution is measured by evaluating the error function on the coordinates visited by the particles. Agents are connected to each other according some topology describing the neighborhood of each agent.

The specific way the velocities of the agents are updated and the topology itself is not fixed: there are various approaches in the literature for specific applications and for more general problem solving. In our work we have followed the recommendations of Kennedy and Mendes [15], and found, that it performs well in finding configurations close to the global minimum.

Algorithm 1 Particle Swarm Optimization

1: **for all** a_i **do**
2: Initialize $\mathbf{x}_i$ for agent a_i within the boundaries of the search space
3: Initialize $\mathbf{v}_i$ for agent a_i
4: Set $\mathbf{b}_i \leftarrow \mathbf{x}_i$
5: Select the neighbors of a_i according to the topology
6: **end for**
7: **repeat**
8: **for all** a_i **do**
9: Update $\mathbf{v}_i$ according to Eq. 2
10: Update $\mathbf{x}_i$ according Eq. 3
11: Calculate the error function $e(\mathbf{x}_i)$ in position $\mathbf{x}_i$
12: **if** $e(\mathbf{x}_i) < e(\mathbf{b}_i)$ **then**
13: $\mathbf{b}_i \leftarrow \mathbf{x}_i$
14: **end if**
15: **end for**
16: **until** termination criterium is met

We have used the Fully Informed Particle Swarm published in [15] with a von Neumann neighborhood.[4] The position and the velocity of the agents are updated according to the following equations:

$$\mathbf{v}_i \leftarrow \chi \left(\mathbf{v}_i + \sum_{n=1}^{N_i} \frac{U(0,\varphi)(\mathbf{b}_{nbr(n)} - \mathbf{x}_i)}{N_i} \right), \tag{2}$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i, \tag{3}$$

where $\mathbf{x}_i$ and $\mathbf{v}_i$ denotes the coordinate and velocity of particle i, $U(min, max)$ is a uniform random number generator, $\mathbf{b}_i$ is the best location found so far by particle i, N_i is the number of neighbors i has and $nbr(n)$ is the nth neighbor of i. The formula has two parameters: χ is the constriction coefficient and φ is the acceleration constant. Again, we have used the recommendations of Kennedy et al., and set $\chi = 0.7298$ and $\varphi = 4.1$.

At the beginning of the search, the agents are initialized with zero velocities and random starting coordinates within some reasonable bounds of them. Then in each iteration these two vectors are updated according to Eqs. 2 and 3 in a synchronized manner. The search is completed if the global minimum found considering all agents does not change for five consecutive iterations. We have experimented with other values and found, that increasing it does not improve the quality of the results, and decreasing it does not reduce the running time considerably.

4 Evaluation on Artificial Infection Scenarios

In this section we will test the performance of the Inverse Infection Problem and its estimation on artificially created infection scenarios. These scenarios take place on graphs with sizes from $|V(G)| = 1000$ to 100000. We will give a detailed discussion of the scenarios in Sect. 4.1. In most subsequent sections of the evaluation we will make a distinction between the direct and coefficient estimations and present result for both of them For the latter approach, a description of the used attributes and attribute functions will be given in Sect. 4.1.1.

We will begin the evaluation of the Inverse Infection Problem and its estimation method by testing the stability and running time of the optimization algorithm. We can show, that the latter can be divided into two parts: the evaluation of the error surface and the speed of the optimization mechanics. Corresponding to this we will start with the review of the complexities of the used infection heuristics and only then we will test the stability of the optimization. The most natural was to do this is by counting the average and maximum number of iterations the method takes before it finishes. In attribute estimation the number of values to estimate in our examples is at

[4] Each agent has four neighbors in a grid, connected to the upper, lower, left and right, while wrapping around the edges.

most 21, while in direct estimation there can be thousands of values. Understandably the focus is on the latter.

The next point of interest is the effect of the infection heuristic on the precision of the estimation. Our findings correspond with those in [2], we will elaborate on this in Sect. 4.3. The results are mostly the same for each estimation type.

In case the exact a priori and a posteriori infection probabilities are not available, the only thing to do is to rely on counting the frequencies of infections. We will show the behavior of our method when the number of learning patterns is minimal. In real life we cannot hope to witness an infection process on any network in more than a handful of times. It is therefore necessary to investigate the sensitivity of our method to low-quality inputs. We will discuss this in Sect. 4.4.

4.1 Test Dataset

As a basis of our analysis we have used graphs generated with the forest fire method of Leskovec et al. [19]. We have created a series of graphs of sizes $n = 1000, \ldots, 100000$, with parameters $p = 0.37$ and $p_b = 0.32$, for forward and backward burning probabilities, respectively. We have assigned a number of edge attributes $a_i, i = 2, \ldots, 10$ to these networks. These attributes are randomly generated: they were drawn independently from a uniform distribution between $[0, 0.5]$. We have also used randomly generated a priori infections. The expected size of the set of initial infectors is $0.3 * n$. If v is selected, then an a priori infection probability was drawn from an uniform distribution between $[0, 0.5]$, otherwise $p_v = 0$. We have used various attribute functions, description of these will be given in the next subsection. Finally for each network and each attribute function we have created an a posteriori infection distribution as the reference dataset. For this purpose we have used Complete Simulation with sample size $k = 10000$, because this method gives the best approximation of the original IC model.

4.1.1 Attribute Functions

We have seen, that in coefficient estimation, the edge infection probabilities are computed from some additional information on the edges in the form of edge attributes by so-called attribute functions. The choice of these attribute functions is an important part of our method. A natural requirement of this choice is, that it must result in infection probabilities: it must map into the $[0, 1]$ interval.

There are two approaches to this problem: the first one is to construct problem-specific functions, taking into account the structure of the network, the nature of the infection model and the number and domain of the attributes. This way it is possible to calculate the infection probabilities directly, without any form of additional normalization. This is the obvious choice if the above mentioned information is available.

If we do not have this information, we can try a more user-friendly approach. We can apply functions to the individual attributes, summarize them and finally normalize them. A variety of functions might be considered for this purpose. In our work, we have used low-degree polynomials for the individual attributes and simple addition or multiplication to join them. We have normalized the resulting edge infection probabilities according to

$$\text{norm}(\mathbf{e}) = \frac{\mathbf{e} - \min(\mathbf{e})}{3(\max(\mathbf{e}) - \min(\mathbf{e}))}, \tag{4}$$

where $\mathbf{e}$ is a vector containing the infection probabilities for each individual edge. The reason why we have used the multiplier 3 in the denominator is, that according to our findings in the prediction of default events on banking data, the edge infection probabilities are low [5]. The normalizer function obviously distorts the shape of the individual attribute functions, but in real-life problems a simple weighted, normalized sum of attributes is sufficient to produce acceptable results.

In this paper, we have used seven attribute function configurations, a_i denotes attribute i and c_j denotes coefficient j:

- Weighted sum of two attributes: $c_1 a_1(e) + c_2 a_2(\text{e})$, two coefficients in total.
- Weighted sum of four attributes: $\sum_i c_i a_i(e), i = 1, 2, 3, 4$, four coefficients in total.
- Weighted sum of six attributes: $\sum_i c_i a_i(e), i = 1, \ldots, 6$, six coefficients in total.
- Weighted sum of eight attributes: $\sum_i c_i a_i(e), i = 1, \ldots, 8$, eight coefficients in total.
- Weighted sum of ten attributes: $\sum_i c_i a_i(e), i = 1, \ldots, 10$, ten coefficients in total.
- Sum of quadratic polynomials with eight attributes $c_1 + \sum_i (c_{2i} a_i(e)^2 + c_{2i+1} a_i(e))$, $i = 1, \ldots, 8$, 17 coefficients in total.
- Sum of quadratic polynomials with ten attributes $c_1 + \sum_i (c_{2i} a_i(e)^2 + c_{2i+1} a_i(e))$, $i = 1, \ldots, 10$, 21 coefficients in total.

4.2 Computational Time

The time complexity of the inverse infection estimation is the sum of two distinct parts of the algorithm: evaluating points on the error surface and the search method itself. The latter consists of the repeated evaluation of the formula in Sect. 3.2, after initializing the neighborhood and the starting coordinates. Since the number of agents is relatively small, this part of the algorithm is very fast, and most of the running time is spent with the evaluation of error surface.

In each iteration every agent evaluates the error function. This evaluation is the computation of the GC model using the coordinates[5] of the given agent. The time complexity of this step heavily depends on the used heuristic. Altogether, we can

[5] Again, these are the edge weights or the coefficients of the attribute function.

say, that the time complexity of a single run is $s * a * h$, where s is the number of iterations, a is the number of agents (a constant) and h is the time complexity of the infection heuristic. This also means, that we can describe the time complexity of the algorithm by measuring the average or maximum number of iterations and multiplying it with the time complexity of the infection method and the number of agents. Breaking the time complexity of the method into different factors makes sense because of another reason: the individual runs of the infection heuristics may be run on multiple threads simultaneously, significantly improving the speed of the method. In the subsequent subsections we will discuss all of these starting with the evaluation of the error surface—the speed of the infection heuristics, the stability of the optimization algorithm and finally we will give a short example on the total running time of our method on an average personal computer.

4.2.1 Speed of the Heuristics

Previously, in Sect. 2.2, we have given short descriptions of some heuristics of the GC model published in [2]. We will give a short reminder on their running times on the networks discussed in the previous section. The mentioned heuristics are:

- Complete Simulation (CS) with sample size $k = 10000$, a very accurate simulation.
- Complete Simulation (CS) with sample size $k = 1000$, a very fast simulation.
- Edge Simulation (ES) with sample size $k = 100$, a simulation based heuristic.
- Neighborhood Bound Heuristic (NBH), an extremely fast lower approximation.
- Aggregated Linear Effect (ALE) model, a DeGroot [9] based simplification of the infection process.

We are going to use two different datasets to evaluate the running time. First, small networks with $|V(G)| \leq 5000$ will be used to make general observations, then we will test the more robust heuristics on large networks with $|V(G)| = 10000, \ldots, 100000$. Our largest network has 100000 vertices and 2.3 million edges.

Our results on the running times[6] of these heuristics on small networks correspond with our previous findings [2]. The speed of the simulations are governed by the sample size. Complete Simulation is considerably faster than ES[7] because the latter focuses on the fast computation of smaller infections. By decreasing the sample size CS can tackle larger networks as well. The Neighborhood Bound Heuristic is able to compute the a posteriori infections of the largest networks within a minute, enabling the heuristic to scale upwards and handle real-life datasets and networks with possibly millions of nodes and edges (Figs. 2 and 3).

[6] We have implemented the methods in JAVA, and we have used a computer with an Intel i7-2630QM processor, and 8 Gb of memory.

[7] Note the sample sizes.

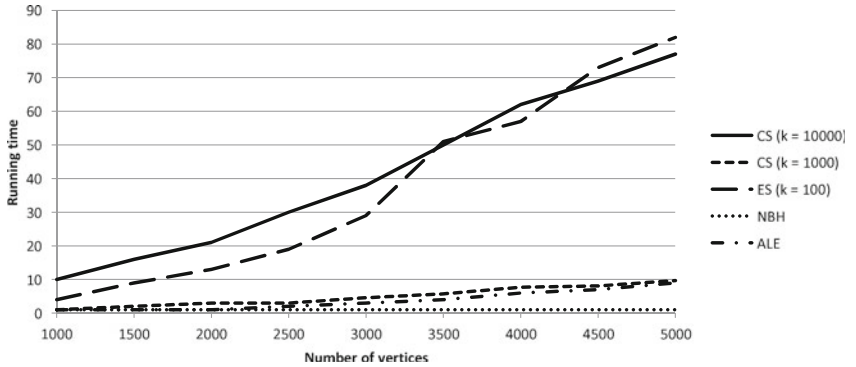


Fig. 2 The running time of the infection heuristics with different network sizes measured in seconds. Figure used with the permission of the authors [2]

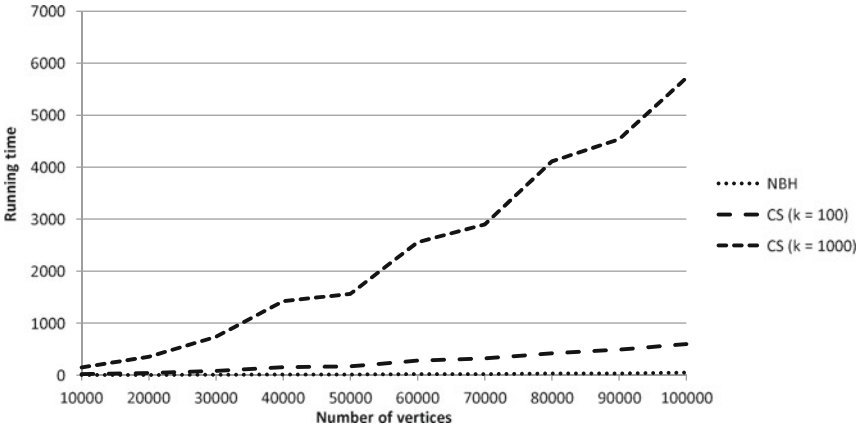


Fig. 3 The running time of the infection heuristics on large networks measured in seconds

4.2.2 Stability of the Optimization

Let us take a look at the stability of the optimization. We have to make a distinction between direct and coefficient estimations, because both the number of agents required to handle the task and the expected number of iterations are different.

On Fig. 4 we can see the average number of iterations for different numbers of coefficients. We have used a small network with $|V(G)| = 1000$. The point of interest here is, starting from a simple problem with only two coefficients to more complex ones, the expected number of iterations grows slowly, and stabilizes around 12. The maximum number of iterations remains bounded as well, even in the experiment with 21 coefficients, it does not go beyond 30. The results shown on Fig. 4 were computed with 9 agents. We have tried this problem with 16 agents as well and got similar results.

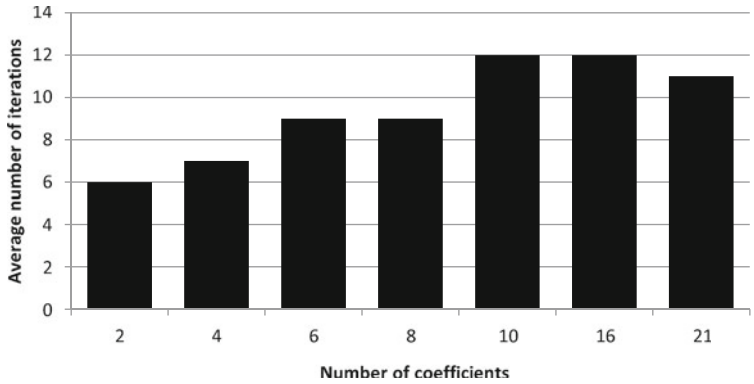


Fig. 4 The average number of iterations with different attribute functions

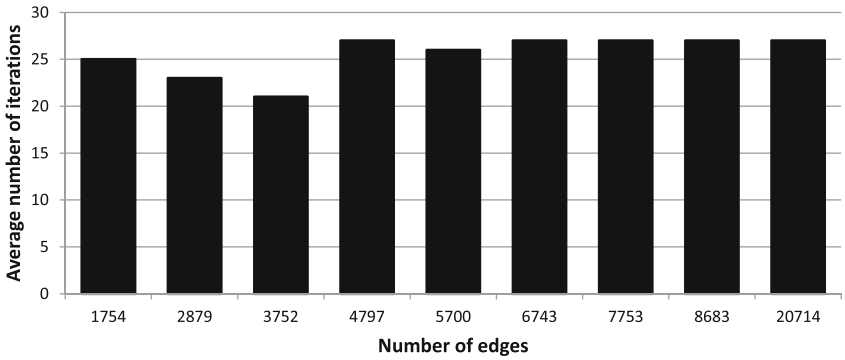


Fig. 5 The average number of iterations with direct estimation

If we compare the different infection heuristics, they perform similarly, with the non-Monte Carlo methods finishing slightly sooner, usually by 4–5 iterations.

A more interesting question is the stability of direct optimization, where dimensionality of the problem is in the thousands. On Fig. 5 the magnitude of the optimization task is represented as the number of edges $|E(G)|$ which is of course is equal to the dimensionality of the problem. The values correspond to networks with sizes of $|V(G)| = 1000, \ldots, 5000$. We can see that the average number of iterations is surprisingly low considering the task, it settles around 27, but the process rarely takes more than 50 iterations to conclude. The number of agents was 100. We have experimented with smaller and greater values, but they had little effect on the stability and the precision of the optimization.

Unlike before, there is a difference between the used heuristics. With this many agents, Edge Simulation is too slow to produce results even with $k = 100$. Complete Simulation performs well both in speed and accuracy, but with the other Monte Carlo method, the sample size should be significantly decreased. The Neighborhood Bound heuristic is very fast, but it fails to produce acceptable accuracy, as we will see in Sect. 4.3.

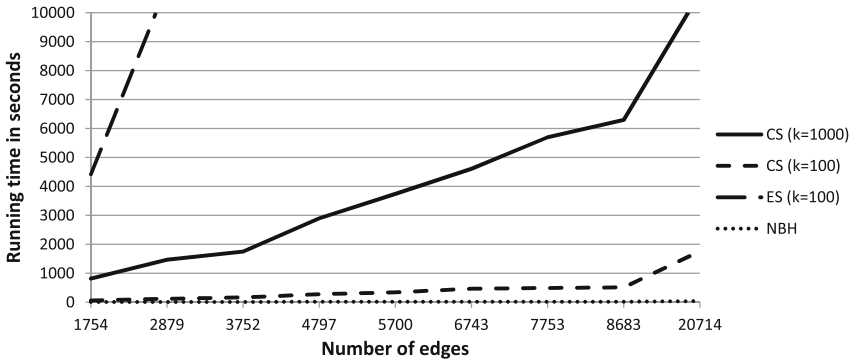


Fig. 6 Total running time of direct estimation

We can conclude, that the Particle-Swarm Optimization method described in this section is able to give an estimation of the Inverse Infection Problem with satisfying results. The algorithm is very stable, and even in the worst case of direct estimation, it finishes within 50 iterations.

4.2.3 Total Running Time

Finally let us take a look at the total running time of the algorithm. As we have mentioned before, the total running time is the function of the infection heuristic, the number of agents and the number of iterations and computations can be simplified by evaluating the points of the error surface in a parallel fashion. Therefore Fig. 6 is simply an example meant to illustrate that our method is able to produce results on an average personal computer. We can see the running times of the more demanding direct estimation with a hundred agents, and eight threads. We can confirm our previous observations: Edge Simulation is too slow to be of any use, and even though NBH is extremely fast we will see, that its accuracy is not acceptable. This leaves us with Complete Simulation.

4.3 Accuracy of the Inverse Infection Estimation

We are going to continue our discussion with the precision of the inverse infection estimation itself: we are going to test both direct and coefficient estimations on the artificial infection scenarios of Sect. 4.1. For our computations we will use the infection heuristics described in the previous section with the same settings. As we have mentioned before, we will use CS with sample size $k = 10000$ to create an a posteriori distribution as a reference set. Then we will use each heuristic with the optimization method to estimate the edge infection probabilities.

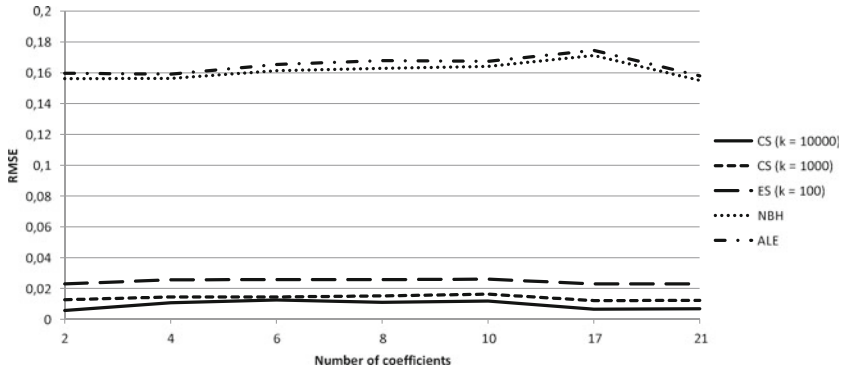


Fig. 7 The RMSE of coefficient estimation on a small network with $n = 1000$

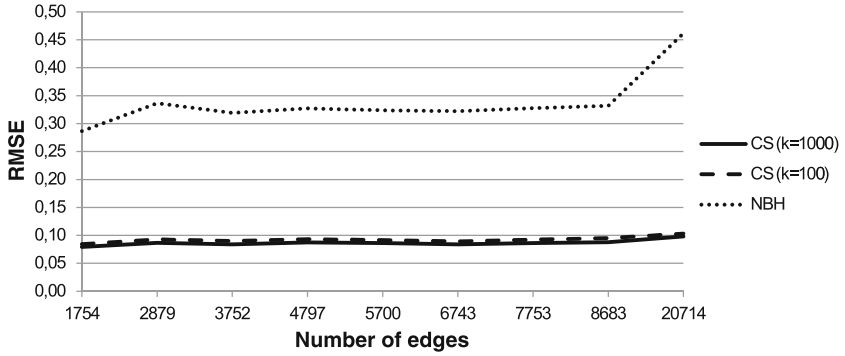


Fig. 8 The RMSE of direct estimation on networks with $|V(G)| = 1000, \ldots, 5000$

We can see the difference between the estimated and reference a posteriori infection values for coefficient estimation on Fig. 7. We have used a small network with $|V(G)| = 1000$. As we can see in this example, the Monte Carlo based simulations (CS and ES) are able to estimate the reference distribution well, with the measured error between 0.01 and 0.03. The other two heuristics (NBH and ALE) are tailored to small edge infection probabilities with rare infections, hence they do not perform so well on this dataset. Note, that in some cases even an error of this magnitude is acceptable, and the time complexity of these methods allows them to handle larger networks. Finally, if we compare the results computed with different attribute functions, we can see that they have minimal effect on the accuracy of the methods. Yet this is only coefficient estimation so this is not surprising.

We can see the precision of direct estimation on Fig. 8. Again, the dimensionality of the task is represented as the size of $|E(G)|$ for graphs with $|V(G)| = 1000, \ldots, 5000$. Edge Simulation was left out because of the computational reasons discussed in the previous section, and ALE failed to produce acceptable output. We can see, that even though the Neighborhood Bound Heuristic is very fast, in direct

estimation it is unable to solve the optimization task. The only working heuristic is Complete Simulation, and even though it produces less accurate output than with coefficient estimation its precision is acceptable, with the added property of the accuracy remaining mostly the same as the dimensionality increases.

Considering both problem variants, we can conclude, that the use of Complete Simulation is recommended. Both its precision and accuracy are good for both direct and coefficient estimations. If the infection probabilities are lower than our current dataset, the use of Edge Simulation is also advisable for coefficient estimation. The Neighborhood Bound Heuristic is able to handle even larger networks, yet this comes at the cost of a significantly lower precision which only makes it applicable for coefficient estimation.

4.4 Number of Patterns

In many real-life applications the a priori or a posteriori probabilities of infections are unavailable. In this section we are going to assume, that the initial infection probabilities are given, but we only have a small number of observations on the a posteriori infections. We are going to simulate this on a small network by generating an a posteriori distribution using CS with $k = 1, \ldots, 10$, corresponding to $1, \ldots, 10$ observations.

We can see, that the proposed method gives a rough estimate of the vertex infection probabilities in only a few iterations. If we consider a threshold of 0.15 as an acceptable estimation, our method only requires 6 observations to reach it. However, it is important to keep in mind, that the method tries to create a posteriori infections close to the reference. The problem is underdetermined even with exact possibilities of vertex infection, with only a handful number of observations many attribute function configurations (and edge weights) may result in the same infection. The results in this section only imply, that our method is able to give one of these.

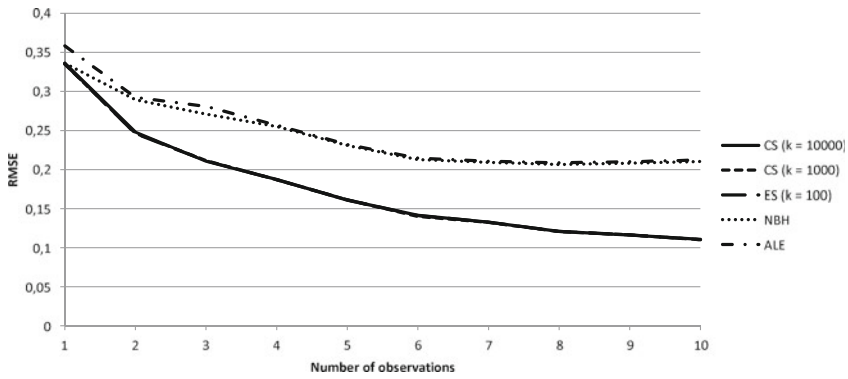


Fig. 9 The precision of the infection heuristics with a limited number of observations of the reference distribution

Different infection heuristics are shown on Fig. 9, one can see, that the simulations have identical performance regardless of the sample size. As before, the non-Monte Carlo based methods perform poorly, their use is not recommended with low-quality inputs.

5 Conclusions

Our goal on this paper was to extend the previous results of the Inverse Infection Problem and its solution. We have given a detailed description and analysis of the Generalized Cascade Model and the Inverse Infection Problem. We have given two variants of the latter. The first one is the direct estimation of the edge weights of the given graph. We have also proposed coefficient estimation, where the edge weights are considered to be a function of known attributes on the edges, thus reducing the dimensionality of the problem.

We have formulated the Inverse Infection Problem as an optimization task and given a Particle-Swarm Optimization algorithm capable of giving a good estimation. Several aspects of the method were investigated: We have tested the stability and accuracy of the optimization method, we have examined the implications of choosing between the heuristics of the GC model and we have tested our method in low-quality inputs as well.

The results were given for both direct and coefficient estimations. For both variants, the optimization method is able to predict the edge infection probabilities in a small number of iterations, while the number of attributes and the shape of the attribute functions have only a small effect on this. There are differences between the two variants however. It is only possible to use direct estimation if the graph is small enough, not just because the dimensionality of the problem, but because the number of agents required. This means the slower heuristics like Edge Simulation should not be used. The precision of the heuristics should be high, leaving NBH and ALE out of the picture, with Complete Simulation remaining to be the only recommended heuristic. Even with Complete Simulation, the accuracy of the method is less than with coefficient estimation. Still, if there are no attributes available and the graph is small enough, direct estimation can be used.

Coefficient estimation significantly reduces the dimensionality of the problem. It is faster, since less agents are required, and is less dependent on the used heuristic. Precision is the highest with the Monte Carlo heuristics, and even the Neighborhood Bound heuristic is able to produce acceptable results. The latter heuristic is extremely fast, with it is possible to handle large graphs with millions of nodes and edges. Coefficient estimation also handles low-quality inputs well.

We have shown the usefulness of our method in a recent paper, where we have given an application of this method in the prediction of credit default on bank transaction networks [1]. We were able to predict the bankruptcies of the riskiest 5 % clients 10 % more accurately than the currently used method. Our model was implemented in August 2013 into the OTP Bank of Hungary's credit monitoring process.

Acknowledgments The first and second authors were supported by the European Union and co-funded by the European Social Fund. Project title: "Telemedicine-focused research activities on the field of Mathematics, Informatics and Medical sciences. Project number": TÁMOP-4.2.2.A-11/1/KONV-2012-0073.

The third author was supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TÁMOP-4.2.2.C-11/1/KONV-2012-0013).

References

1. A. Bóta, A. Csernenszky, L. Győrffy, G. Kovács, M. Krész, A. Pluhár, Applications of the inverse infection problem on bank transaction networks. Cent. Eur. J. Oper. Res. 1–12 (2014)
2. A. Bóta, M. Krész, A. Pluhár, Approximations of the generalized cascade model. Acta Cybern. **21**, 37–51 (2013)
3. A. Bóta, M. Krész, A. Pluhár, Systematic learning of edge probabilities in the Domingos-Richardson model. Int. J. Complex Syst. Sci. **1**(2), 115–118 (2011)
4. A. Bóta, M. Krész, A. Pluhár, The inverse infection problem, in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems* (IEEE, 2014), pp. 75–83. http://dx.doi.org/10.15439/978-83-60810-58-3
5. A. Csernenszky, Gy. Kovács, M. Krész, A. Pluhár, T. Tóth, The use of infection models in accounting and crediting. Challenges for Analysis of the Economy, the Businesses, and Social Progress Szeged (2009), pp. 617–623
6. T. Cao, X. Wu, T.X. Hu, S. Wang, Active learning of model parameters for influence maximization, in *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, ed. by D. Gunopulos, et al. (Springer, Berlin, 2011), pp. 280–295. doi:10.1007/978-3-642-23780-5_28
7. W. Chen, Y. Yuan, L. Zhang, Scalable influence maximization in social networks under the linear threshold model, in *Proceeding ICDM'10 Proceedings of the 2010 IEEE International Conference on Data Mining* (IEEE Computer Society, 2010), pp. 88–97. doi:10.1109/ICDM.2010.118
8. W. Chen, C. Wang, Y, Wang, Scalable influence maximization for prevalent viral marketing in large-scale social networks, in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2010), pp. 1029–1038. http://doi.acm.org/10.1145/1835804.1835934
9. M.H. DeGroot, Reaching a consensus. J. Am. Stat. Assoc. **69**(345), 118–21. http://www.tandfonline.com/doi/pdf/10.1080/01621459.1974.10480137
10. O. Diekmann, J.A.P. Heesterbeek, Mathematical epidemiology of infectious diseases, *Model Building, Analysis and Interpretation* (Wiley, Chichester, 2000)
11. P. Domingos, M. Richardson, Mining the network value of costumers, in *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining* (ACM, 2001), pp. 57–66. http://doi.acm.org/10.1145/502512.502525
12. A. Goyal, F. Bonchi, L.V.S. Lakshmanan, Learning influence probabilities in social networks, in *Proceedings of the Third ACM International Conference on Web Search and Data Mining* (ACM, 2010), pp. 241–250. http://doi.acm.org/10.1145/1718487.1718518
13. M. Granovetter, Threshold models of collective behavior. Am. J. Sociol. **83**(6), 1420–1443 (1978). http://psycnet.apa.org/doi/10.1086/226707
14. J. Kennedy, Particle swarm optimization. Encyclopedia of Machine Learning (Springer, US, 2010), pp. 760–766. doi:10.1007/978-0-387-30164-8_630
15. J. Kennedy, R. Mendes, Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **36**(4), 515–519 (2006). doi:10.1109/TSMCC.2006.875410

16. D. Kempe, J. Kleinberg, E. Tardos, Maximizing the spread of influence though a social network, in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2003), pp. 137–146. http://doi.acm.org/10.1145/956750.956769
17. D. Kempe, J. Kleinberg, E. Tardos, Influential nodes in a diffusion model for social networks, in *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)* (Springer, 2005), pp. 1127–1138. doi:10.1007/11523468_91
18. M. Kimura, K. Saito, Tractable models for information diffusion in social networks, *Knowledge Discovery in Databases*, Lecture Notes in Computer Science (Springer, Berlin, 2006), pp. 259–271. doi:10.1007/11871637_27
19. J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations, in *Proceedings of the 1st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2005), pp. 177–187. http://doi.acm.org/10.1145/1081870.1081893

On a Quantum Algorithm for the Resolution of Systems of Linear Equations

J.M. Sellier and I. Dimov

Abstract The numerical resolution of systems of linear equations is an important problem which recurs continously in applied sciences. In particular, it represents an indispensable tool in applied mathematics which can be utilized as a foundation to more complicated problems (e.g. optimization problems, partial differential equations, eigenproblems, etc.). In this work, we introduce a solver for systems of linear equations based on quantum mechanics. More specifically, given a system of linear equations we introduce an equivalent optimization problem which objective function defines an electrostatic potential. Then, we evolve a many-body quantum system immersed in this potential and show that the corresponding Wigner quasi-distribution function converges to the global energy minimum. The simulations are performed by using the time-dependent, ab-initio, many-body Wigner Monte Carlo method. Finally, by numerically emulating the (random) process of measurement, we demonstrate that one can extract the solution of the original mathematical problem. As a proof of concept we solve 3 simple, but different, linear systems with increasing complexity. The outcomes clearly show that our suggested approach is a valid quantum algorithm for the resolution of systems of linear equations.

1 Introduction

Quantum computing is the art of exploiting quantum effects for computational purposes. While it is clear, from a theoretical perspective, that incredible speedup could be achieved with such technology, how to actually build such devices is still an important open problem. Different approaches have been proposed such as the quantum gate and the adiabatic models, just to mention a few. In particular, the quantum gate model is very similar to the classical electronics paradigm. Its aim is, indeed, to process (quantum) information by means of (quantum) gates [1], thus allowing the implementation of more complex circuits which can achieve practical and useful

J.M. Sellier (✉) · I. Dimov
Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences, Acad. G. Bonchev 25A, 1113 Sofia, Bulgaria
e-mail: jeanmichel.sellier@parallel.bas.bg; jeanmichel.sellier@gmail.com

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_3

functionalities. Classical gate electronics has been very successful in this respect, and it is more than natural that the vast majority of the scientific community thinks in terms of gate models when refering to quantum computing. Furthermore, mathematical investigations have clearly shown that a quantum computer based on the gate model would offer an incredible advantages over classical computers, representing an additional motivation [1]. Thus it is not surprising that a great deal of efforts have been spent in trying to physically build the basic blocks of quantum gate electronics (qubits and gates) [2–7]. Unfortunately, this paradigm necessitates technological exploits that we may not be reachable yet. Indeed, it requires an incredibly high control of the quantum states which are rapidly destroyed by the environmental noise [1, 8].

On the other hand, quantum computing based on the adiabatic model seems to be within our reach already. As a matter of fact, on may 2011 the D-Wave company announced the first commercial quantum computer [9], based on an implementation of the quantum annealing algorithm [10–13] and empowering 128 flux qubits, and a second generation machine, with 512 flux qubits, is already being benchmarked by Google, NASA and a consortium of universities showing that the machine is an actual quantum machine [14]. The question of speedup still remains an open discussion [15], though one should note that calculation times for this very *young* technology are already comparable to the biggest clusters available (based on a *mature* technology) [15]. This is certainly a very encouraging fact which opens the door toward achievable quantum computing. These machines are special purpose machines which solve combinatorial optimization problems only. In practice, calculations can be achieved since the adiabatic paradigm does not necessitate extreme control over the environment as it is, instead, required by the gate model. In this respect, the situation is very similar to the early days of electronic computers where analog designs were implemented successfully to construct special purpose machines.

In this paper, we take inspiration from the actual situation in quantum computing and present an analog numerical quantum algorithm for the resolution of systems of linear equations. One cannot stress enough on the importance of such mathematical problems. The resolution of systems of linear equations appears in a plethora of important scientific and engineering practical situations. For instance, in Chemistry the simulation of the electrostatic potential is made possible by the resolution of a system of linear equations (due to the discretization of the Poisson equation). In Physics and Engineering, most of the time the simulation of models expressed as a set of partial differential equations eventually reduces to the resolution of linear systems. Not to mention that in applied Mathematics a linear solver represents one of the most powerful numerical tool for the resolution of even more complex numerical tasks such as optimization problems, eigenproblems, etc. It is, therefore, not surprising that quantum algorithms for the resolution of such linear systems have been already (recently) proposed [16, 17], and the reader should give particular attention to the fact that, while these works are completely based on the gate paradigm, the main purpose is the actual construction of a quantum machine.

In this study, our goal is mainly the suggestion of a new numerical approach to the resolution of linear systems. In practice, this algorithm is based on the simulation of

quantum many-body problems and, as such, strongly relies on the time-dependent many-body Wigner Monte Carlo method [22]. A way to depict (virtual) quantum systems, which evolution in time provide the solution of a given linear system, is introduced. In particular the Wigner quasi-distribution function of the system is shown, for long enough times, to converge towards a state which points to the global energy minimum. Conequently, a set of (random) measurements is emulated which recovers the solution of the original algebraic problem. Finally, in order to show the reliability of such proposed quantum algorithm, we solve relatively simple linear systems, as a proof of concept, involving one, two and 16 unknowns. Although further mathematical investigation is still needed (and some interesting comments are reported in the conclusion section), we believe that these promising (and preliminary) results show a valid alternative to other more classical methods, and offer a way to bypass typical numerical problems such as, for example, the presence of local minima.

2 A Quantum Solver for Linear Systems

In this section, we introduce the fundamental concepts needed to solve a system of linear equations by simulating a corresponding quantum system. The numerical experiments to show the feasibility of such approach are performed and discussed in the next section.

The initial problem. Let us suppose to have a $n \times n$ real and invertible matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ (with n being a natural number) and $\mathbf{b} \in \mathbb{R}^{n \times 1}$ a n-dimensional real vector (the generalization to complex systems is trivial). Our problem consists of finding the real n-dimensional vector $\mathbf{x}$ (unknown) such that

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}, \tag{1}$$

in other words the resolution of a system of linear equations. The fact that $\mathbf{A}$ is invertible guarantees the uniqueness of the solution.

Due to the high significance of this problem in practical applications, an incredible number of solvers have been developed all based on deterministic approaches/concepts. For example, nowadays available methods are the Gauss-Jordan eliminations, the LU decomposition, the Singular Value method, the Cholesky method, and QR decompositions, Relaxation methods such as the Gauss-Seidel and Jacobi methods, just to mention a few of them. Moreover, a plethora of numerical libraries are available, e.g. [19, 20]. The situation is quite different on the other side of the spectrum. A few quantum algorithms exist too, but based on the gate model and not available for any practical application yet [17].

The optimization problem. Now, one can think of the previous problem as an optimization problem. Indeed, if one defines the following objective function

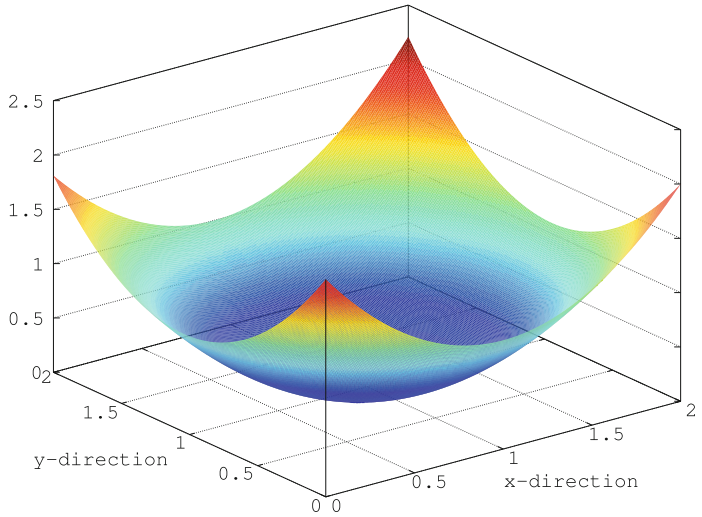


Fig. 1 Typical landscape of a two-dimensional quadratic objective function given at a certain time $t > 0$. If interpreted as an electric potential, it is easy to see how the electrons reach the *bottom* of the main (and unique) valley

$$F = F(\mathbf{x}) = \sum_{i=1}^{n} (a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - b_i)^2, \tag{2}$$

it is trivial to show that F is quadratic non-negative definite, has a unique global minimum which corresponds to the (unique) solution of the system (1), and $F(\mathbf{x}^*) = 0$ where $\mathbf{x}^*$ is the solution of (1). Thus, one can think of the problem (1) as the (equivalent) optimization problem which objective function is (2). The two problems have both the same unique solution and the same complexity $O(n^2)$. A typical shape for a (two-dimensional) objective function is shown in Fig. 1.

The quantum solver. Having reformulated the problem (1) in terms of an optimization problem, one can physically interpret the objective function (2) as a fixed electrostatic potential in which a system of electrons evolves. However in practice, the objective function is multiplied by a non-decreasing function $s = s(t)$, where $s(0) = 0$ and $s(t) = 1, \forall t > T_C$, and T_c equal to an (arbitrary) fraction of the final time T_f. Thus, the potential evolves in time until T_c:

$$V = V(\mathbf{x}; t) = s(t) \times F(\mathbf{x}), \tag{3}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. At this point, two approaches are conceivable. One may immagine a rather abstract electron defined in n-dimensional space which evolves in the n-dimensional (increasing in time) potential (3). Another (more realistic) possibility is to think of a system of n electrons defined in a one-dimensional space and which dynamics is shaped by the time-dependent formula (3). This last interpretation

has several important advantages. First of all, it defines a physically sound system which *in principle* could be realized, leading to a special-purpose quantum machine able to solve systems of linear equations. However, how to build such a machine is not the goal of this work and, unfortunately, remains an unclear task to us at the present time. In spite of it, this interpretation represents a very suggestive perspective and several further comments are given in the conclusions of this paper. Secondly, it gives the opportunity to introduce an artificial dissipative background [23] which assists the trapping of electrons in the global energetic minimum (somehow miming phonon assisted charge trapping).

A final note must be given for the sake of clarity. In order to deal with practical numerical problems, the function (3) is rescaled to have reasonable, and physically meaningful, values. Obviously, this procedure does not change the position of the global minimum which corresponds to the solution of the linear system we aim to solve. Moreover, while one could have thought of a time-independent potential such as $V = V(\mathbf{x}) = F(\mathbf{x})$, it is observed from our numerical tests that the introduction of the function $s = s(t)$ in (3) speeds up the convergence up (rapidly pushing the electrons towards the global minimum).

3 Simulations and Validation

In this section, we perform several numerical experiments to show how a physical system of electrons in the potential defined in (3) can actually reach the (unique) bottom of the energy valley, thus, providing the solution of the linear system (1) when their position is measured. For every $n \times n$ system of linear equations it is possible to construct a corresponding physical system consisting of n electrons evolving in the potential defined by (2) and (3). First, we simulate such systems, i.e. n-body problems, for simple case studies ($n = 1, 2$) to show that the physical structure actually provides the correct solution. Thus we proceed with the more interesting case $n = 16$. To this goal, we apply the many-body Wigner Monte Carlo method described in [22].

3.1 Numerical Technique

In the Wigner formulation of quantum mechanics (mathematically equivalent to the Schrödinger formalism) [18] the description of a system is based on the concept of a quasi-distribution function defined over the $d \times n$-dimensional phase-space (with $d = 1, 2, 3$ the dimensionality and n the number of electrons). For a one-dimensional system of n electrons, its time-dependent evolution equation reads:

$$\frac{\partial f_W}{\partial t}(\mathbf{x};\mathbf{p};t) + \sum_{k=1}^{n} \frac{p_k}{m}\frac{f_W}{x_k} = \int d\mathbf{p}\, f_W(\mathbf{x};\mathbf{p};t) V_W(\mathbf{x};\mathbf{p};t), \tag{4}$$

with $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ the set of coordinates of the particles, $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ the momenta of the particles, $\int d\mathbf{p} = \int dp_1 \int dp_2 \ldots \int dp_n$, m the mass of the electron, and the Wigner kernel $V_W = V_W(\mathbf{x};\mathbf{p};t)$ defined as

$$V_W(\mathbf{x};\mathbf{p};t) = \frac{i}{\pi^n \hbar^{n+1}} \int d\mathbf{x}' e^{-\left(\frac{2i}{\hbar}\right)\sum_{k=1}^{n} x'_k p_k} \left[V(\mathbf{x} + \frac{\mathbf{x}'}{2};t) - V(\mathbf{x} - \frac{\mathbf{x}'}{2};t) \right]. \tag{5}$$

In this work, all simulated quantum many-body problems are described by the partial integro-differential equation (4) which is solved by means of the many-body Wigner Monte Carlo method presented in [22] and based on [24, 25].

We now proceed with several numerical experiments to show the efficiency of the proposed method.

3.2 The Case $n = 1$

In the first experiment, we start from one equation and one unknown. While this first example have no impact in practical calculations, it is an interesting case study to perform a validation of the proposed physical system, which in this case consists of a one-dimensional single electron.

We start from the following problem

$$ax = b,$$

where $a = 1$ and $b = 1$. The solution is trivial ($x = 1$). Now, our goal is to show that the proposed system can recover the solution correctly. In order to compute the solution physically, we prepare the following experiment. An electron is isolated in a box which dimensions are 250 nm and immersed in the potential (coming from the objective function defined above and rescaled on the spatial domain)

$$V(x;t) = s(t) \times F(x) = s(t) \times (ax - b)^2 .$$

This is a parabolic potential which has a unique global minimum corresponding to 0 eV at a position which corresponds to the solution of the linear system (when $t > T_c$). Concerning the electron, it is initially a Gaussian wave packet [21], with σ the dispersion of the packet equal to 32 nm and x_0 the position of the central peak of the packet equal to 125 nm. The system is evolved until a final time equal to 60 fs is reached. The experiment is performed 256 times at the end of which a random selection of the position is performed according to the probability density calculated during the simulation. This is equivalent to an 256 identical experiments which position measurements give 256 different outcomes.

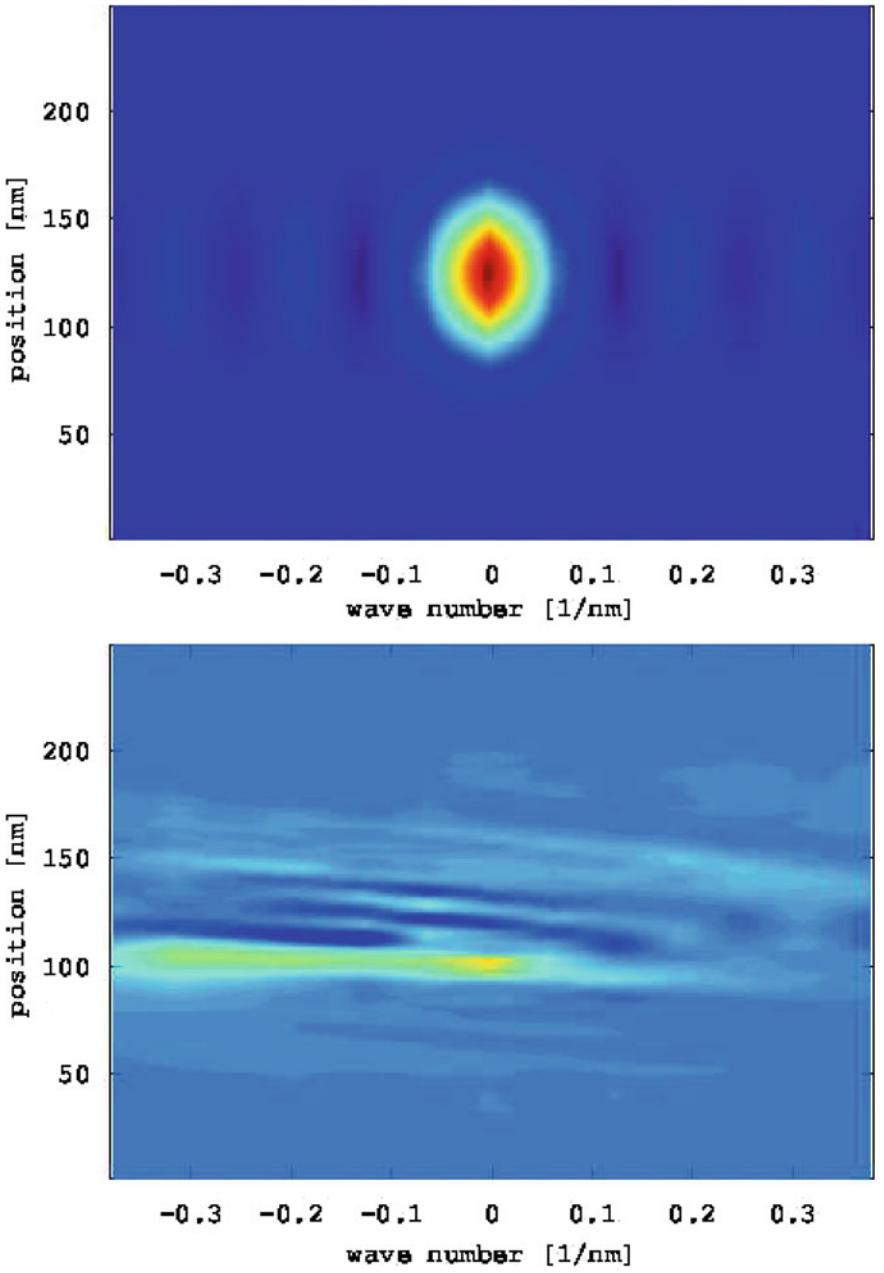


Fig. 2 Wigner quasi-distribution function at 0 fs (*top*) and 60 fs (*bottom*) respectively. It is clear that the distribution function tends to a δ-function, in time, which peak is in the proximity of the solution of the linear system (100 nm in this specific case)

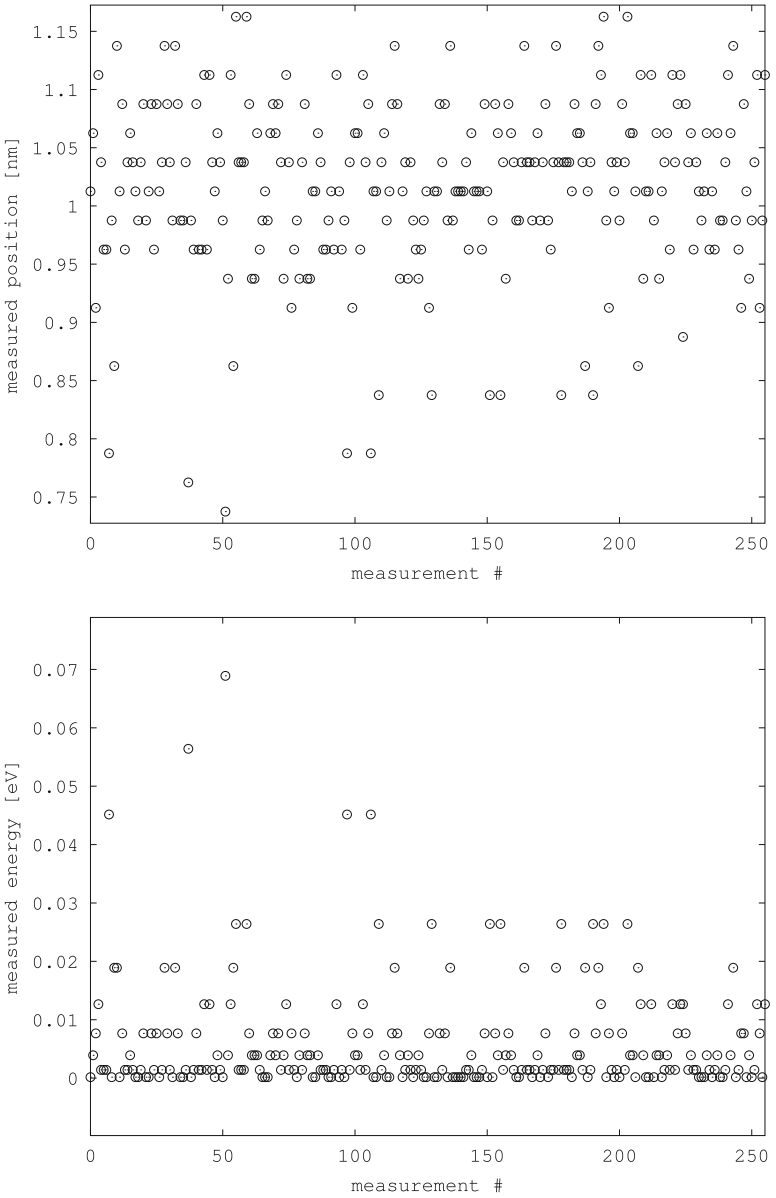


Fig. 3 Resolution of 1 linear equation in 1 unknown. 256 equivalent physical systems are simulated at the end of which a measurement of the position of the electron is performed. The *top* plot shows the outcomes for these measurements (in a rescaled space). One clearly sees that the probability of finding the electron at the position corresponding to the solution of the system is very high. The *bottom* plot shows the corresponding energy of the potential at the position of the electron. It is always close to 0 eV, i.e. the global minimum. This is a clear indication of the efficiency of the proposed method

The results of this preliminary simulation are shown in Figs. 2 and 3. In particular, Fig. 2 shows the initial Wigner quasi-distribution function at 0 fs (top) and at 60 fs (bottom). The distribution function at 60 fs has clearly evolved toward a δ-function in space, peaking exactly at the solution of the linear system (at 100 nm which, rescaled, corresponds to our solution $x = 1$). Figure 3 shows 256 simulated position measurement outcomes (rescaled) after the probability density has been evolved. The top plot shows the obtained solutions while the bottom plot shows the corresponding energies. It is clear that the average solution is correct. Indeed, the average is around 1 while the energy is clearly close to 0 eV for the vast majority of performed measurements. This first simple case study shows, as a proof of concept, that a physical system can be depicted to solve one equation in one unknown.

3.3 The Case $n = 2$

We now proceed with a slightly more complex case study for $n = 2$ and apply our proposed method to the system

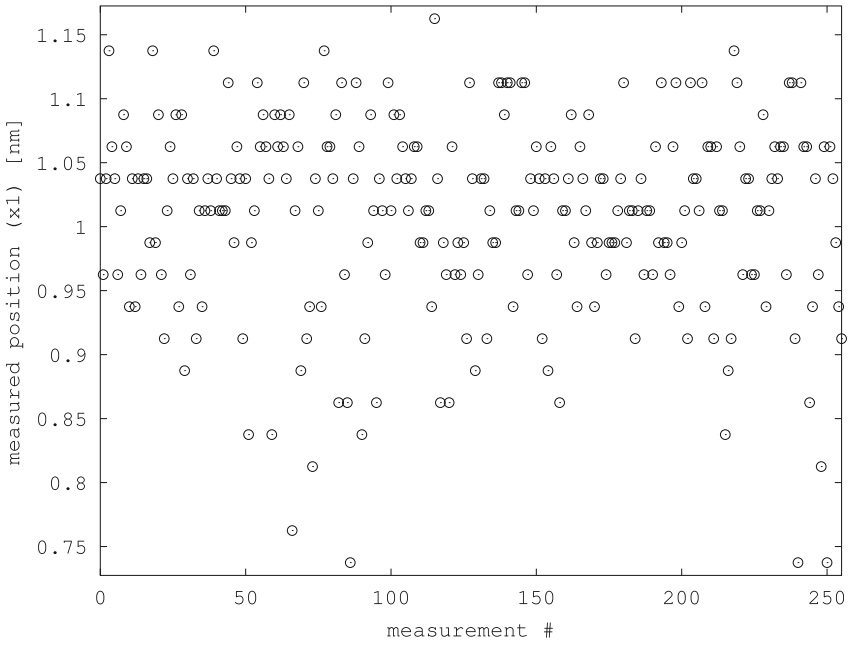


Fig. 4 Resolution of a system of 2 linear equations in 2 unknown. 256 equivalent physical systems are simulated at the end of which a measurement of the position of the electrons is performed. The plot shows the outcomes of measurements for the first electron in a rescaled space. The probability of finding the electron at the position corresponding to the first component of the solution of the system is very high

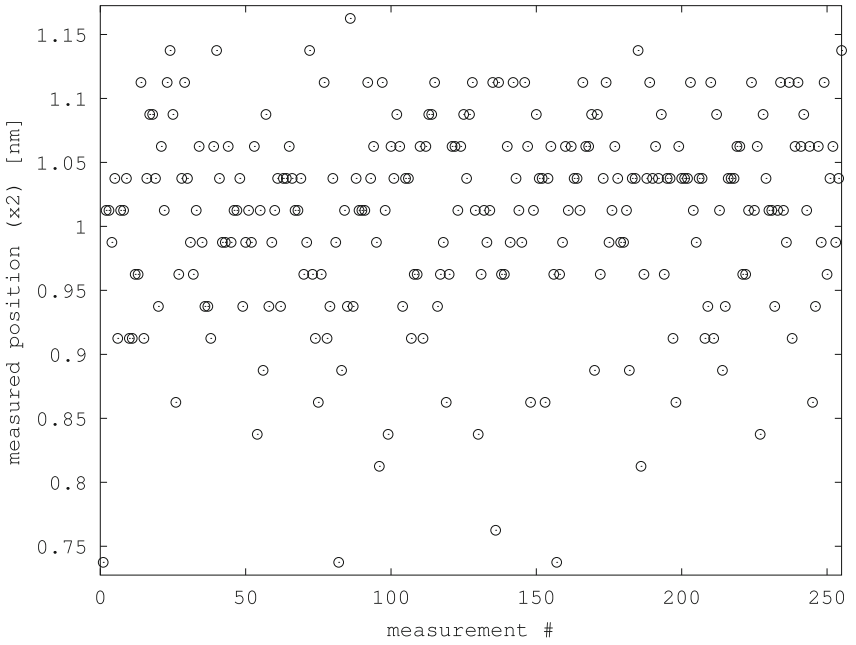


Fig. 5 Resolution of a system of 2 linear equations in 2 unknown. 256 equivalent physical systems are simulated at the end of which a measurement of the position of the electrons is performed. The plot shows the outcomes of measurements for the second electron in a rescaled space. The probability of finding the electron at the position corresponding to the second component of the solution of the system is very high

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix},$$

with $c_{11} = 1$, $c_{22} = 1$, $c_{12} = c_{21} = 0$ and $g_1 = g_2 = 1$. The solution is trivial ($x_1 = 1$, $x_2 = 1$). The case where c_{12} and c_{21} are different than 0 is a simple spatial translation in the reduced phase-space (x_1, x_2) and, thus, the complexity of the problem, from a computational point of view, remains the same.

This time, our physical system consists of 2 non-interacting electrons evolving in the following potential calculated from (2) and (3)

$$V(x_1, x_2; t) = s(t) \times F(x_1, x_2) = s(t) \times \left\{ (c_{11}x_1 - g_1)^2 + (c_{22}x_2 - g_2)^2 \right\}.$$

Again, this function is quadratic by definition and has a unique global minimum which corresponds to the value 0 eV located at the position corresponding to the solution of the linear system.

In order to compute the solution, two electrons, initially Gaussian, are evolved in a one-dimensional space where the potential is $V = V(x_1, x_2; t)$ defined above.

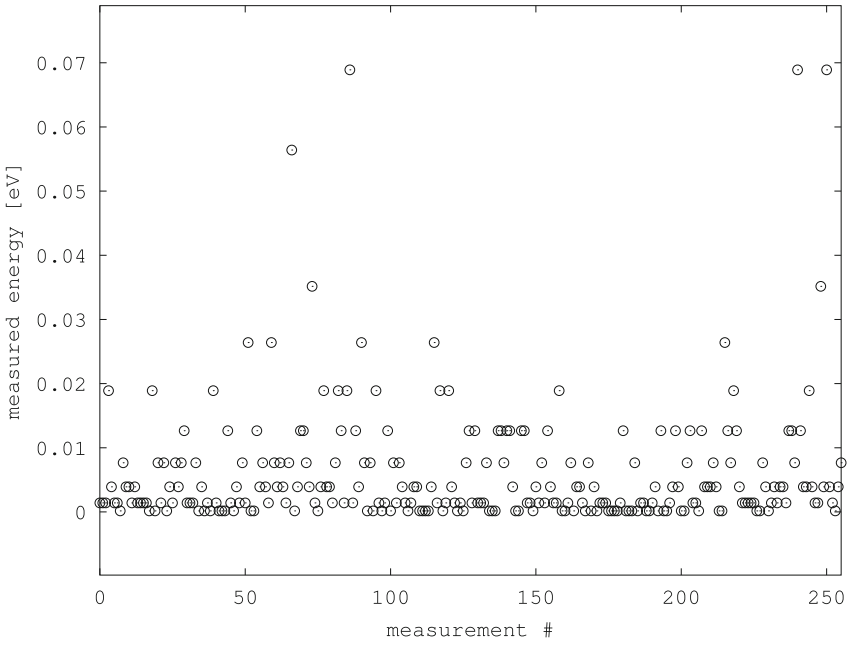


Fig. 6 Resolution of a system of 2 linear equations in 2 unknown. The plot shows the corresponding energy of the potential at the position of the electrons reported in Figs. 4 and 5. It is always close to 0 eV, i.e. the global minimum. This is a clear indication of the efficiency of the proposed method

This corresponds to simulate the evolution of the Wigner quasi-distribution function in a four-dimensional phase-space. The potential starts from a flat profile and increases in time until $s(t) = 1$ while the electrons feel a slightly dissipative background. The final time is fixed to be 60 fs and 256 equivalent experiments are performed at the end of which a measurement is performed. It is, thus, possible to extract the solution out of the 256 measurements as we did for the case $n = 1$.

The results are reported in Figs. 4, 5 and 6. In particular, Fig. 4 shows the outcomes of (rescaled) measurements for the position of the first electron, Fig. 5 shows the outcomes of (rescaled) measurements for the position of the second electron, and Fig. 6 shows the corresponding energies. It is clear, from Fig. 6, that the global minimum of the potential landscape has been reached, i.e. the solution of the system has been found. This demonstrates that, as a proof of concept, the proposed quantum algorithm works even for the case $n = 2$.

3.4 The Case n = 16

In order to show that our proposed quantum algorithm works for meaningful systems as well, we perform a simulation involving 16 one-dimensional electrons which aim is to solve a system of $n = 16$ linear equations. We consider the linear system

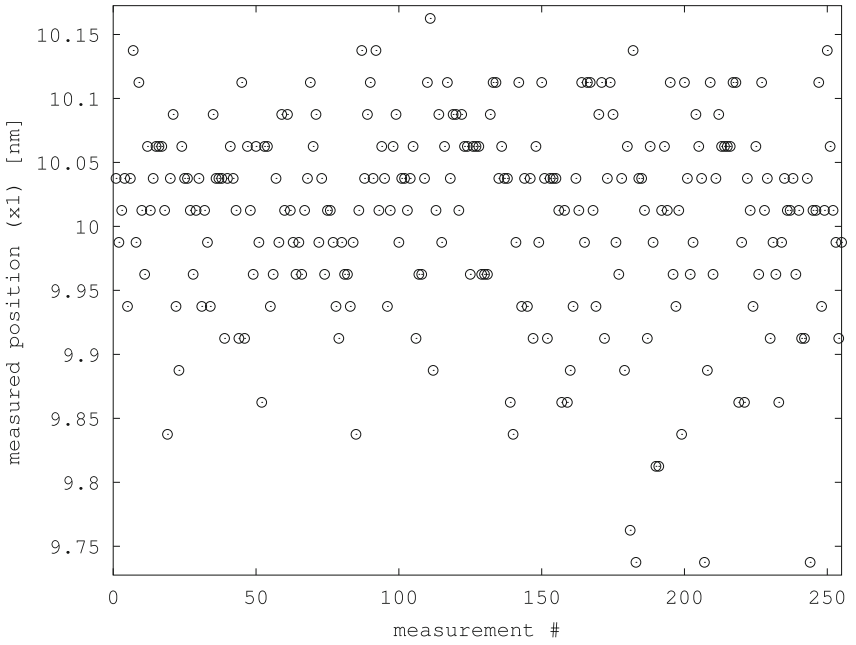


Fig. 7 Resolution of 16 linear equation in 16 unknown. The plot shows the outcomes for 256 measurements (in a rescaled space) for the x_1 component

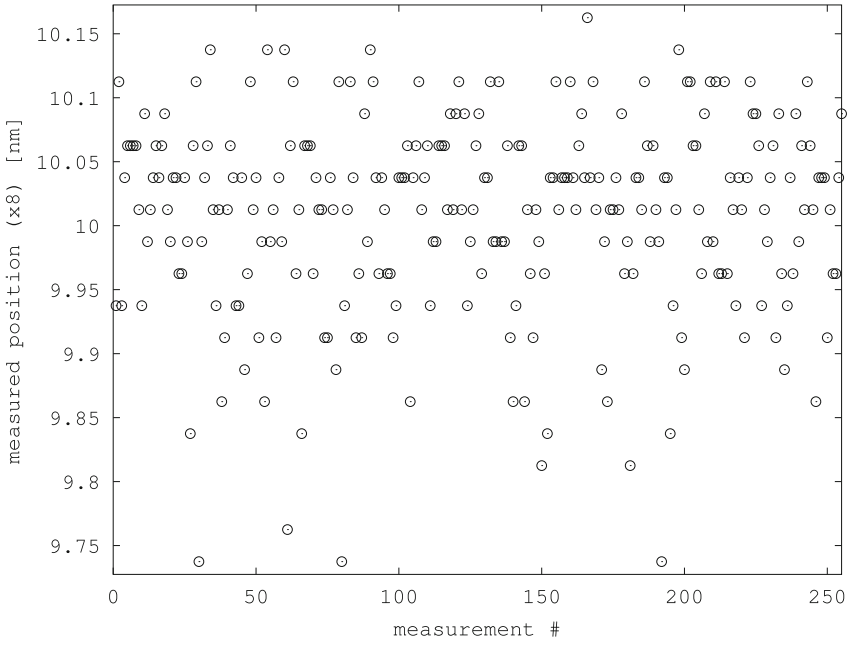


Fig. 8 Resolution of 16 linear equation in 16 unknown. The plot shows the outcomes for 256 measurements (in a rescaled space) for the x_8 component

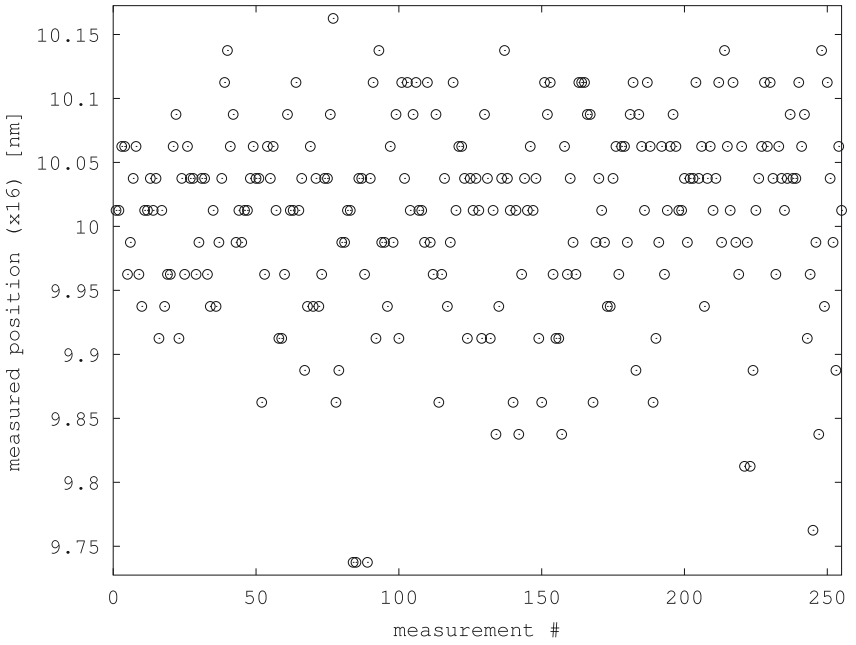


Fig. 9 Resolution of 16 linear equation in 16 unknown. The plot shows the outcomes for 256 measurements (in a rescaled space) for the x_{16} component

$$\mathbf{x} = \mathbf{H} \cdot \mathbf{x} + \mathbf{k},$$

where $h_{ij} = \frac{a}{n}$, $k_i = 1$, $a = 0.9$, $j = 1, 2, \ldots, 16$ and whose exact solution is known to be $x_i = \frac{1}{1-a}$ for every $i = 1, 2, \ldots, 16$. This is a classical example often utilized by the numerical community to validate new solvers.

The corresponding physical system consisting of 16 electrons evolving in the electrostatic potential shaped by the objective function of the linear system is simulated and 256 measurements of the position for each of the 16 electrons is performed at the end of each experiment. The outcomes are shown in Figs. 7, 8 and 9 for selected components of the solution (x_1, x_8 and x_{16} respectively). Even in this relatively more complex case, our proposed method appears to be efficient in finding the correct solution.

3.5 Notes on Convergence

It is important to comment on Fig. 10, reporting the evolution of the average of the position of an electron after every single measurement. This particular case is taken from a (randomly) selected component of the solution of the system $n = 16$.

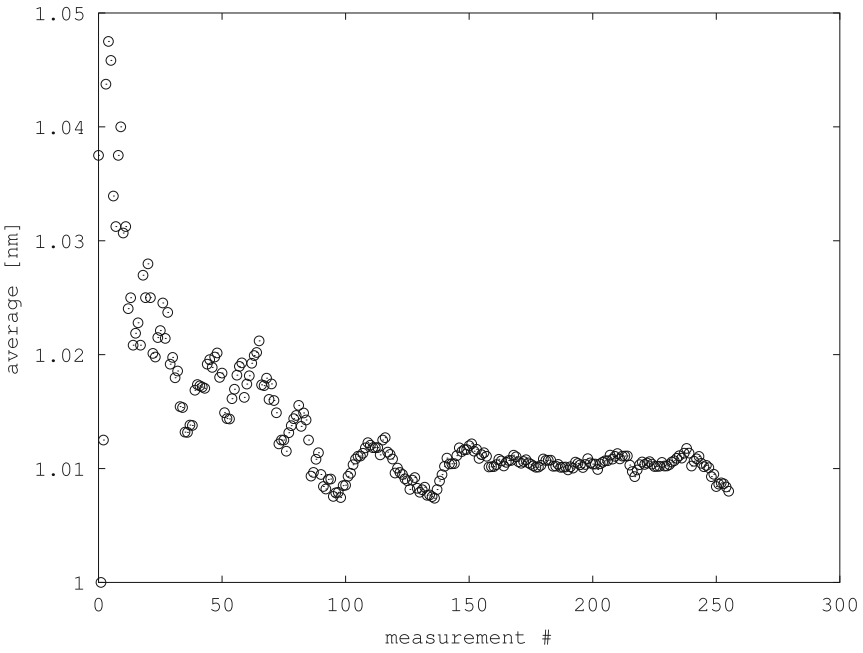


Fig. 10 Typical plot of the average of the electron position in function of the number of performed measurements. The plot shows a clear convergence towards the solution of the system (256 measurements)

This plot sheds light on the convergence properties of our suggested solver. It is important to note that typical convergence patterns like this one appear for *every* component of the solution of every test performed in this work. This is a clear indication that our quantum solver converges towards the correct answer and gives a glance about the speed of convergence of the algorithm.

3.6 Computational Aspects

The simulator used to obtain the results presented in this paper is a modified version of Archimedes, the GNU package for the simulation of carrier transport in semiconductor devices [26] which was first released in 2005 under the GNU Public License (GPL). In this particular project, named *nano-archimedes*, our aim has been to develop a full quantum time-dependent simulator. The code is entirely developed in C and optimized to get the best performance from the hardware. It can run on parallel machines using the OpenMP standard library. The results of the present version are posted on the nano-archimedes website, dedicated to the simulation of quantum systems [27]. The source code is available as well.

The results have been obtained using the HPC cluster deployed at the Institute of Information and Communication Technologies of the Bulgarian Academy of Sciences. This cluster consists of two racks which contain HP Cluster Platform Express 7000 enclosures with 36 blades BL 280c with dual Intel Xeon X5560 @ 2.8 GHz (total 576 cores), 24 GB RAM per blade. There are 8 storage and management controlling nodes 8 HP DL 380 G6 with dual Intel X5560 @ 2.8 GHz and 32 GB RAM. All these servers are interconnected via non-blocking DDR Infiniband interconnect at 20Gbps line speed. The theoretical peak performance is 3.23 Tflops.

4 Conclusions

In this paper, we introduced a numerical solver for systems of linear equations which exploits typical concepts of quantum mechanics without involving the use of quantum gates. Given a linear system of order n, the solver evolves n electrons in a (initially time-dependent) electric field shaped by an objective function (2) recovered from the initial linear system (1). As a proof of concept, we applied the proposed method to simple (but meaningful) case studies corresponding to $n = 1$, $n = 2$, and $n = 16$. We have shown that, by means of position measurements of the involved electrons, it is possible to extract the correct solution of the proposed linear systems. Figures 3 (top), 4, 5, 7, 8 and 9 clearly show that the solution of the respective linear systems is found correctly, while Figs. 3 (bottom) and 6 show that the global minimum of the corresponding electrostatic potential is reached, providing an indication of the quality of the found solutions. Finally, Fig. 10 shows the convergence speed of a randomly selected component of the solution for the case $n = 16$. Similar convergence patterns can be shown for other components of the solution and for other systems.

Here we mainly focused on providing new concepts and validate them. It is clear that further mathematical and theoretical investigations need to be carried out. For instance, one open question is related to one of the most fascinating effect of quantum mechanics: in all examples shown we started from non-entangled particles, but what would happen if entanglement were introduced? How that would affect the convergence of the solver? Another open question concerns the fine tuning of the dissipative background introduced in the many-body simulations. It is not clear, at this stage, how to automatically select the right amount of dissipation. As a matter of fact, when too much noise is introduced the system stops to be in quantum coherence destroying any hope to exploit effects such as tunnelling and entanglement. On the other hand, if too less background is introduced the system hardly reaches a stationary state. Moreover, it would be interesting to introduce more realistic dissipative models such as phonon scattering due to the lattice vibrations [28], and see how this affects the overall behavior of the solver. Another important point is how the solution provided by our solver is influenced by the (inevitable) errors introduced in the input. We believe that this question can be answered by utilizing techniques typical of the field of sensitivity analysis, and an investigation in this direction could bring interesting insights.

Finally, we would like to spend a few words on the possibility of building such a machine. In this paper, we have shown a correspondence between systems of linear equations and physical systems which are simulated by means of the many-body Wigner Monte Carlo approach. We did not provide any practical detail on how these systems could be achieved in real life as we have been more interested in showing its reliability as a first step. As a matter of fact, we do not even know if potentials such as in Eq. (3) are physically realizable. Despite the foreseeable technical challenges, we think that our quantum algorithm remains very suggestive and may shed some light on how to experimentally proceed. In a sense, the situation is very similar to the quantum annealing algorithm at its early stage [11]. It tooks several decades before a physical implementation of the algorithm could come into existence [9].

We firmly believe that by answering these important open questions, we could obtain novel mathematical and technological tool. This will be the subject of further papers in the next future.

Acknowledgments This work has been supported by the the project EC AComIn (FP7-REGPOT-2012-2013-1).

References

1. M.A. Nielsen, I.L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000)
2. B.E. Kane, A silicon-based nuclear spin quantum computer. Nature **393**, 133–137 (1998)
3. L.C.L. Hollenberg, A.S. Dzurak, C. Wellard, A.R. Hamilton, D.J. Reilly, G.J. Milburn, R.G. Clark, Charged-based quantum computing using single donors in semiconductors. Phys. Rev. B **69**, 113301 (2004)
4. K. Yokoi, D. Moraru, M. Ligowski, M. Tabe, Single-gated single-electron transfer in nonuniform arrays of quantum dots. Jpn. J. Appl. Phys. **48**, 024503 (2009)
5. E. Hamid, D. Moraru, J.C. Tarido, S. Miki, T. Mizuno, M. Tabe, Single-electron transfer between two donors in nanoscale thin silicon-on-insulator field-effect transistors. Appl. Phys. Lett. **97**, 262101 (2010)
6. D. Moraru, A. Udhiarto, M. Anwar, R. Nowak, R. Jablonski, E. Hamid, J.C. Tarido, T. Mizuno, M. Tabe, Atom devices based on single dopants in silicon nanostructures. Nanoscale Res. Lett. **6**, 479 (2011)
7. M. Simmons, S. Schofield, J. Obrien, N. Curson, L. Oberbeck, T. Hallam, R. Clark, Towards the atomic-scale fabrication of a silicon-based solid state quantum computer. Surf. Sci. **532–535**, 1209–1218 (2003)
8. E. Nielsen, R.W. Young, R.P. Muller, M.S. Carroll, Implications of simultaneous requirements for low-noise exchange gates in double quantum dots. Phys. Rev. B **82**, 075319 (2010)
9. M.W. Johnson, M.H.S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A.J. Berkley, J. Johansson, P. Bunyk, E.M. Chapple, C. Enderud, J.P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M.C. Thom, E. Tolkacheva, C.J.S. Truncik, S. Uchaikin, J. Wang, B. Wilson, G. Rose, Quantum annealing with manufactured spins. Nature **473**, 194–198 (2011)
10. T. Kadowaki, H. Nishimori, Quantum annealing in the transverse Ising model. Phys. Rev. E **58**, 5355 (1998)
11. A.B. Finilla, M.A. Gomez, C. Sebenik, D.J. Doll, Quantum annealing: a new method for minimizing multidimensional functions. Chem. Phys. Lett. **219**, 343 (1994)

12. G.E. Santoro, E. Tosatti, Optimization using quantum mechanics: quantum annealing through adiabatic evolution. J. Phys. A **39**, R393 (2006)
13. A. Das, B.K. Chakrabarti, Colloquium: quantum annealing and analog quantum computation. Rev. Mod. Phys. **80**, 1061 (2008)
14. S. Boixo, T.F. Rønnow, S.V. Isakov, Z. Wang, D. Wecker, D.A. Lidar, J.M. Martinis, M. Troyer, Evidence for quantum annealing with more than one hundred qubits. Nat. Phys. **10**, 218–224 (2014)
15. T.F. Rønnow, Z. Wang, J. Job, S. Boixo, S.V. Isakov, D. Wecker, J.M. Martinis, D.A. Lidar, M. Troyer, Defining and detecting quantum speedup. Science **345**, 420–426 (2014)
16. J. Pan, Y. Cao, X. Yao, Z. Li, C. Ju, H. Chen, X. Peng, S. Kais, J. Du, Experimental realization of quantum algorithm for solving linear systems of equations. Phys. Rev. A **89**, 022313 (2014)
17. X.D. Cai, C. Weedbrook, Z.E. Su, M.C. Chen, M. Gu, M.J. Zhu, L. Li, N.L. Liu, C.Y. Lu, J.W. Pan, Experimental quantum computing to solve systems of linear equations. Phy. Rev. Lett. **110**, 230501 (2013)
18. E. Wigner, On the quantum correction for thermodynamic equilibrium. Phys. Rev. **40**, 749 (1932)
19. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, *LAPACK Users' Guide* (Society for Industrial and Applied Mathematics, Philadelphia, 1999)
20. M. Galassi et al., *GNU Scientific Library Reference Manual*, 3rd edn. (Network Theory Limited, Bristol, 2009)
21. J.M. Sellier, M. Nedjalkov, I. Dimov, S. Selberherr, A benchmark study of the Wigner Monte-Carlo method. Monte Carlo Methods Appl. De Gruyter (2014). doi:10.1515/mcma-2013-0018
22. J.M. Sellier, I. Dimov, The many-body Wigner Monte Carlo method for time-dependent Ab-initio quantum simulations. J. Comput. Phys. **273**, 589–597 (2014)
23. J.M. Sellier, M. Nedjalkov, I. Dimov, S. Selberherr, Decoherence and time reversibility: the role of randomness at interfaces. J. Appl. Phys. **114**, 174902 (2013)
24. I. Dimov, Monte Carlo Algorithms for Linear Problems, Pliska (Studia Mathematica Bulgarica), Vol. 13 (2000), 1997, pp. 57–77
25. I. Dimov, T. Gurov, Monte Carlo Algorithm for Solving Integral Equations with Polynomial Non-linearity. Parallel Implementation, Pliska (Studia Mathematica Bulgarica), Vol. 13 (2000), 1997, pp. 117–132
26. J.M. Sellier, GNU Archimedes. Available www.gnu.org/software/archimedes. Accessed 08 Dec 2014
27. J.M. Sellier, Nano-archimedes. www.nano-archimedes.com. Accessed 08 Dec 2014
28. J.M. Sellier, I. Dimov, Wigner-Boltzmann Monte Carlo method applied to electron transport in the presence of a single dopant. Comput. Phys. Commun. **185**, 2427–2435 (2014)

Synthesis of Self-Adaptive Supervisors of Multi-Task Real-Time Object-Oriented Systems Using Developmental Genetic Programming

Krzysztof Sapiecha, Leszek Ciopiński and Stanisław Deniziak

Abstract This chapter presents a procedure for automatic creation of self-adaptive artificial supervisors of multi-task real-time object-oriented systems (MT RT OOS). The procedure is based on developmental genetic programming. Early UML diagrams describing a MT RT OOS are used as input data to the procedure. Next, an artificial supervisor which optimizes the system use is automatically generated. The supervisor is self-adaptive what means that it is capable of keeping optimality of the system in spite of disruptions that may occur dynamically in time of the system work. A representative example of creation of a supervisor of building a house illustrates the procedure. Efficiency of the procedure from the point of view of self-adaptivity of the supervisor is investigated.

1 Introduction

Real-time (RT) systems are present in all areas of human life. We can find RT systems in civil engineering, in traveling, in computer engineering, in banking, and so on. In the first case, a building enterprise is such a system. It owns resources, such as workers and building machinery, necessary to build a house according to requirements of a client. These usually comprise functionalities of the house, its cost and a deadline. In the second case a human being is an RT system. It knows which means of transportation may use to meet his requirements. From among flights, trains, buses, rented cars, and even walking he selects a set, so that to reach a target on time and at affordable cost. An embedded computer system may be another example of RT

K. Sapiecha
Department of Computer Engineering, Cracow University of Technology, Cracow, Poland
e-mail: krzysztof.sapiecha@gmail.com

L. Ciopiński (✉) · S. Deniziak
Department of Computer Science, Kielce University of Technology, Kielce, Poland
e-mail: l.ciopinski@tu.kielce.pl

S. Deniziak
e-mail: s.deniziak@tu.kielce.pl

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_4

system. A lot of such systems are dynamic i.e. a cost of their resources or processing time of executed tasks may change at any time.

Complex real life systems may be represented as multi task systems (MS), where more than one task can be processed at the same time. Each task requires some resources (processing components) for its execution. A designer or a manager of such a system has to decide what of the tasks of the system should be assigned to what of its processing components, so that to get maximum of the performance and do not surpass the cost. In RT MS the punctuality is an extra requirement which has to be satisfied. The problem arises when some of the tasks could be delayed, or a cost of some of the allocated resources could be increased. In such cases real time constraints might be violated or the system could be not optimal as far as the total cost is considered. To avoid this inconvenience, the system should be adapted to deal with the problem. Thus, for dynamic RT MS some capabilities of self-adaptivity are required.

Usually, RT systems should be optimized for cost versus speed of operation (speed of reaching a goal or a target). Therefore, a building enterprise, and a traveler, and hardware/software system designer, and other RT systems have to be endowed with optimization engine. We will call these engines: artificial supervisors (AS) or artificial managers (AM) of resources. An AS should find an optimum use of supervised resources, taking into account the requirements and the constraints. This means that the AS decides what functionalities should be allocated to what resources and in which order these functionalities should be executed. Actually, it has to find a solution for a specific case of the well-known Resource-Constrained Project Scheduling Problem (RCPSP) which consists in rescheduling the project tasks (RT system tasks) efficiently using limited renewable resources (components/objects of the RT system) minimizing the maximal completion time of all activities [1].

The RCPSP is an NP-complete problem which is computationally very hard [2, 3]. Möhring et al. [4] states that it is one of the hardest problems of Operational Research. Therefore, a skilled specialist with an assistance of the planner (Computer Aided System Engineering in case of the enterprise) might play a role of such an AS only for small systems containing a limited number of tasks and a moderate number of resources. No doubt, in case of real life systems, particularly RT MS, the AS must be a very powerful optimization engine.

In time of work of the RT MS it may occur that some disruption has occurred and a new schedule must be determined. Unforeseen delays may cause a violation of time constraints. Therefore, the AS should be able to adapt its schedule to any such perturbation. Actually, it should be self-adaptive. In this chapter, a method of automatic generation of the self-adaptive AS (SAAS) is introduced and evaluated with the help of representative example. The method is based on an idea derived from developmental genetic programming (DGP) [5]. The method is universal, but an SAAS must be well-fitted to a particular RT MS. The RT MS is a micro-world with its own functionalities and resources. Therefore, a formal specification of the RT MS is an input data to the method. RT MS, where a number of resources have punctually to execute a number of tasks are good micro-worlds for object-oriented modeling. Objects may play a role of resources that execute tasks in real time for some costs.

A widely accepted standard for modeling object-oriented systems (OOSs) is the Unified Modeling Language (UML) [6]. It shows how to write a system's blueprints, including conceptual things such as business processes and system functions. It encompasses OOSs of any kind, particularly real-time multitask OOSs (RT MOOSs). Using the UML for modelling RT OOS has been a subject of many publications [7–9]. Hence, this will be applied here.

Related work is briefly described in Sect. 2. In Sect. 3 the problem is stated. Section 4 briefly shows how early UML models should be used as an input data for the method, and Sect. 5 explains how DGP can create the supervisors and the initial solutions. In Sect. 6 computational experiments evaluating our approach are described. Experiments verify the self-adaptivity capabilities of the supervisor. Finally, Sect. 7 contains conclusions.

2 Related Work

Genetic programming (GP) is an extension of the genetic algorithm [10], in which the population consists of computer programs. In the DGP, strategies creating solutions evolve, instead of computer programs. In this approach a genotype and a phenotype are distinguished. The genotype is a procedure that constructs a solution of the problem. It is composed of genes representing elementary functions, constructing the solution. The phenotype represents a target solution. During evolution, only genotypes are evolved, while genotype-to-phenotype mapping is used in the fitness computation, which is required for the genotype selection process. Next, all genotypes are rated according to an estimated quality of the corresponding phenotypes. The goal of the optimization is to find the procedure constructing the best solution. The idea is based on the theory from the molecular biology, concerning protein synthesis that produces proteins (phenotype) from the DNA (genotype).

For the first time Developmental Genetic Programming was proposed by Koza et al. [11], to create electrical circuits. This methodology evolves circuit-construction tree, in which nodes correspond to functions defining the developmental process. The initial circuit consists of an embryo and a test fixture. The sample embryo is at least one modifiable wire while fixture is one or more unmodifiable wire or electrical components. The circuit is developed by progressively applying functions in the circuit-construction tree to the modifiable parts (wires and electronic components) of the embryonic circuit.

The similar methodology was used by Deniziak and Górski in the co-synthesis of embedded systems described by task graphs [12]. The system-construction tree is based on a task graph. Each node of the tree specifies an implementation of the corresponding task. The embryo is an allocation of the first task. First (initial) population is created randomly. Then after evolution, using crossover, mutation and reproduction, an optimal (or suboptimal) solution is found.

In [13] a list of 36 instances of human-competitive results produced by the GP is presented. A lot of them concern of synthesis of an analog electrical circuits,

developing quantum algorithms, designing controllers. According to our best knowledge, there is no approach concerning optimization of object-oriented real-time multitask systems using DGP methods.

Genetic approach was proved as very efficient for solving RCPSP problems. Ones of the most efficient genetic algorithms for RCPSP are presented in [14, 15]. Recently, Zhang et al. [16] proposed a modification of genetic algorithm in which a search space was reduced. As concerns the use of self-adapting, Hartmann in [17] proposed a genetic algorithm that chooses the best decoding procedure separately for every individual but not in real time. In [18] the method of improving the genetic algorithm for optimization of multi-task project scheduling was proposed. It was shown that the method is competitive in comparison with 11 other heuristic approaches. A method of solving a large scale RCPSP is presented in [19]. In this solution, a genetic algorithm is used and a method of encoding classical RCPSP problem in the chromosome is described. The results achieved by authors of [19] give a slight improvement, in comparison with other existing heuristics.

Two types of approaches are used for solving RCPSP problems that may dynamically change during the project execution: proactive scheduling and reactive scheduling [20]. The proactive scheduling tries to minimize the effects of any delays by maximization of the minimum or total free slacks of activity [21]. The reactive scheduling is based on rescheduling the tasks that did not start before the disruption. Usually rescheduling tries to minimize the perturbation of the original schedule by minimizing the number of activities that receive a different start time in the new schedule [22].

In [23] we presented a procedure for automatic creation of ASs for RT MOOSs. Representative example showed that the AS can develop the best schedule, which corresponds to the global optimum. However, the procedure did not consider the self-adaptivity of the AS.

3 Problem Statement

Let us assume that information about the functionalities of RT MOOS and resources available for implementation of these functionalities are specified with the help of UML early diagrams: use case, activity, and sequence. This is typical while designing OOS of any kind. To start the implementation (run the system) a supervisor is needed, which allocates the functionalities into the resources in such a way that the cost will be minimal while all real-time constraints will be satisfied. We assume that during the system execution, the costs of resources and the processing times of tasks may change or some resources may be unavailable. Thus the supervisor should try to reschedule tasks and/or reallocate resources in real-time, to minimize adverse effects of any changes. Both, a number of the functionalities and number of resources are large enough to exclude a human being as the supervisor. Therefore, an engine which optimizes supervising the system should be worked out. The engine will be named a self-adaptive artificial supervisor (SAAS), since it does what the supervisor should do. A SAAS should work as follows:

1. It should work out a schedule for RT MOOS which would be optimal under current operational conditions, and
2. Adjust the schedule, to keep its optimality and to avoid violation of real time constraints, when the conditions have changed (some tasks are delayed or the cost of any resource has increased, for example).

The goal of the research is to introduce a method of automatic generation of the SAAS from the diagrams. The procedure consists of two steps. In the first one information included in the UML diagrams are transformed into a task graph and a library of objects working for the system. These are input data to the second step. In this step (Sect. 5) the SAAS is created. To this end a universal method of evolution of a genotype of the SAAS is applied. Decision options, which may be contained in the genes of the SAAS, are defined and then the genotype is created developmentally, using DGP-like approach. An example of the generation of a supervisor in a building enterprise is used to illustrate the method. The enterprise is an RT MOOS because its resources may be dealt with as objects of different kinds, human or technical, which work in real time and in multi-task mode of operation. A user of the RT MOOS specifies the functionalities of the house, a deadline of the implementation and cost constraints. In the case of a small building enterprise, a contractor assisted by a CASE tool (Computer Aided Software Engineering) can elaborate optimal or semi-optimal schedule of building the house. However, big consortia own a large number of resources and implement many different constructions. Hence, this duty must be waived from the contractor and placed onto the SAAS. Not the contractor, but the SAAS, which is engaged in building the house, should generate an optimal schedule for management of enterprise resources.

4 From UML Early Models to Library of Resources

The first step of the method consists in the generation of input data for DGP, which in turn will create an adequate supervisor. To this end UML early models of RT MOOS, which will be under optimization, are used. In case of building a house the models describe all activities of the supervisor that controls the whole process of the construction.

4.1 Functionalities and Sequential Constraints

Functionalities are described with the help of a UML use case diagram where a Supervisor is the main actor. It owns resources (objects) performing tasks in real-time and may face orders of task executions. Use case diagram describing building a house is given on Fig. 1. Actually, it maintains the enterprise which works as a real RT MOOS.

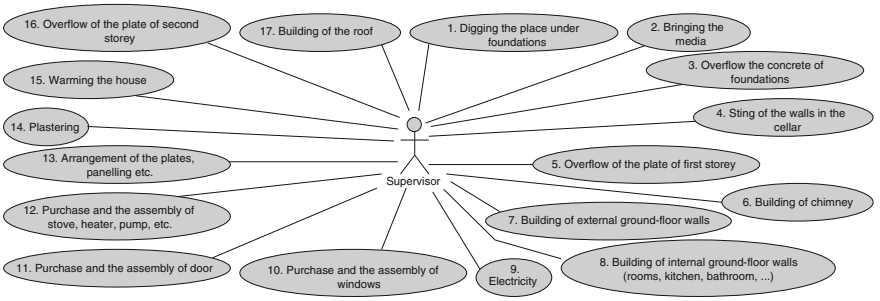


Fig. 1 Use case diagram for building a house

A task is an activity performed by a specific user of an RT MOOS. In the diagram, each of the use cases corresponds to one of such tasks, since a use case is an action performed by an object (objects) which aims to yield an observable goal for the user. Thus, each of the tasks has a use case that explains what the task is, and how it should function. Moreover, a use case may include statements about pre-conditions (required before the task began), post-conditions (valid when the task was successfully completed) and, if needed, exceptions.

The diagram on Fig. 1 contains 17 use cases (stages of a house building; numbered from 1 to 17 on Fig. 2) which should be scheduled for enterprise resources. Therefore, assignment of the use cases, to the resources, is a subject of optimization. However, the diagram may not say anything, that one of the cases must be used before another one. Digging foundations must precede their laying, and plastering must be done before warming a house, for example.[1] In general, an RT MOOS as an example of a multi-task system may have sequential constraints. Tasks should not be executed in arbitrary orders because some of the tasks need to be executed before others.

The Supervisor knows use case sequential constraints. This can be specified with the help of an extension and of an inclusion associations (<<extend>> or <<include>> stereotypes [6]) and on pre- and post-conditions defined for the use cases (sequential dependencies [24]). As the summary, a UML activity diagram [6] for an RT MOOS can be defined, and then transferred into a task graph (TG) showing an execution of the tasks in a real-time. The constraints for the example are shown on Fig. 2a whereas their corresponding TG is shown on Fig. 2b.[2]

4.2 Resources

The resource is an object required to execute a task. In general, it could be a human, a tool or any other object, which is reusable or renewable. If there are many resources of the same type, each of them should be presented as a separate resource.

[1] Numerical prefixes are introduced to identify the use cases and will be used later on.

[2] Automatic generation of TG from UML diagrams is possible but will not be discussed in the paper.

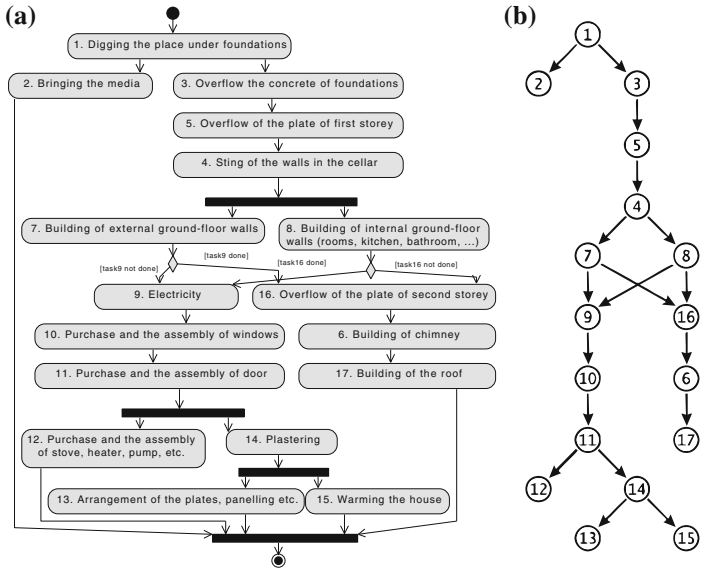


Fig. 2 Activity diagram (**a**) and task graph (**b**) of the system

In the example, two types of the resources are required for building a house. First are Workers. These are labourers like electricians, plasterers, and so on, that are able to execute some specific tasks. Also, a company, which could be used as an outsourcing, should be given as a resource. A worker could use the second type of the resources which are Tools. These are machines which could be used by workers during execution of tasks. Hardly ever one resource is able to complete a task itself. Thus resources are grouped into work teams, which will be described in the next subsection.

Not all resources are necessary for the execution of some tasks. This could be determined by the Supervisor with the help of the third kind of UML diagrams, namely sequence diagrams.

4.3 Scenarios of the Resource Cooperation

Inspired by real world, where most tasks are executed by a group of resources, a concept of a team is introduced. The team is a set of resources that are able to execute a task. Any task may be executed by more than one team. We assume that the execution cost and time of the task are known. One resource may belong to different teams, but teams having the same resources cannot be scheduled at the same time period.

A use case is refined into one or more sequence diagrams to show how the case might be implemented with the help of detailed actions. Therefore, each sequence

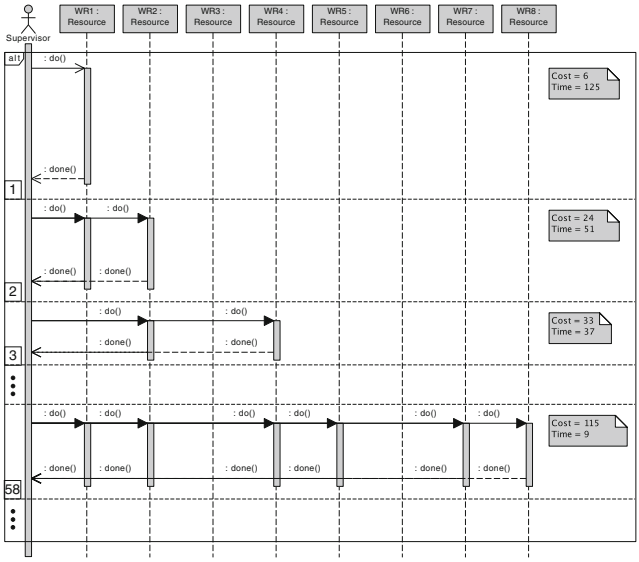


Fig. 3 Sequence diagram of bringing the media use case

of the actions defines an actual work-flow and reflects a sequence of decisions the supervisor should take to perform a single task. Every task is executed by teams (single worker team is also possible, but rarely). Different teams are able to finish their work faster or cheaper, using more or less resources. The sequence diagrams have to define these scenarios.

Figure 3 shows an example how the supervisor might interact with objects participating in the construction of the building.[3] Moreover, the sequence diagrams show options (with time and cost of their implementations) available to the supervisor of the enterprise.

It is characteristic for a Supervisor that while traversing a task graph it determines step by step what should be done at that point and that its selections are usually optional. This means that it actually decides what functionalities of the RT MOOS should be assigned to what objects and in which order these functionalities should be executed. Its decisions should be optimal, taking into account costs and the time of execution. Therefore, the quality of the supervisor should be as high as possible.

4.4 Library of Resources

Sequence diagrams specify how cooperating objects are organized in teams and how do they work. For example, Fig. 3 shows 4 teams. Each of the teams could bring media to a house under construction, but with different workload and costs.

[3] Remaining 16 sequences are very similar.

Table 1 A library of resources

Task #	Team # (for the task #)	Time	Cost	Time * Cost	Members
1:	0	125	6	750	WR1
1:	1	51	24	1224	WR1, WR2
1:	2	37	33	1221	WR1, WR2
…	…	…	…	…	…
1:	58	9	115	1035	WR1, WR2, WR4, WR5, WR7, WR8
…	…	…	…	…	…
1:	62	82	183	15006	WR1, WR2, WR3, WR4, WR5, WR7, WR8
2:	0	57	62	3534	WR3
…	…	…	…	…	…
2:	3	41	82	3362	WR1, WR2, WR5
…	…	…	…	…	…
2:	62	10	189	1890	WR3, WR4, WR5, WR7, WR8
…	…	…	…	…	…

From sequence diagrams a table is derived which determines a binding of tasks with teams. For the example, this is given in Table 1.

Columns “Time”, “Cost” and “Members” of the table are filled in with data from sequence diagrams. Units of “Time” or “Cost” are inessential. It could be a day or an hour, dollar or euro. It is only important that all costs and all times are defined using the same units.

Column “Time * Cost” does not give any new information, but it is helpful to accelerate the time of the computations.

5 Creation of Supervisors

The second step of the method consists of initiating and evolving genotypes, corresponding to the supervisors, with the help of DGP. It is assumed that the supervisor selects options defining the strategy of the allocation of resources. The way in which it does, it is a specific feature of its mind, and it is contained in its genotype. A supervisor with the best genotype (allocating the resources optimally) will be

generated with the help of DGP. DGP evolves genotypes, while genotype-to-phenotype mapping is used in the fitness computation, which is required for the genotype selection process. It is possible, that one phenotype may be created from two different genotypes, because genotype-to-phenotype mapping always generates systems that meet the system requirements.

A genotype corresponding to the supervisor has a form of a tree engineering the system. A root of the tree specifies a construction of an embryonic system, while all other nodes correspond to functions which progressively build up the whole system. If the system is defined by a task graph, then an embryo is a system executing the first task from the task graph. Thus, the number of possible embryos equals the number of teams, in the library of resources, which are capable of executing the first task. Embryonic systems are selected randomly for each attempt to create an initial population of supervisors.

5.1 Supervisor's Options

The supervisor undertakes the following two actions:

- resource allocation and task assignment, that send an appropriate team to execute a particular task and hence, allocate members of the team,
- task scheduling (only when more than one task is assigned to the same resource), that schedules the tasks assigned to the resources. When the resource is unavailable, the execution of the task is delayed as long as the resource is not released.

Initial population of supervisors consists of randomly generated genotypes. It selects one of the options given in part 1 of its decision table. Table 2 contains the options which the supervisor may choose. The last column in Table 2 shows a probability of the selection.

The first option prefers a team, which requires the smallest period of time to execute a task. Second one prefers a team, which brings the lowest cost increase. Third option prefers a team with the best ratio of the costs to the time of the execution.

Table 2 Supervisor's options

Step	Option	P
1	a. The fastest team	0.16(6)
	b. The cheapest team	0.16(6)
	c. The lowest time * cost	0.16(6)
	d. Determination by second gene	0.16(6)
	e. The fastest starting team	0.16(6)
	f. The fastest ending team	0.16(6)
2	List scheduling	1

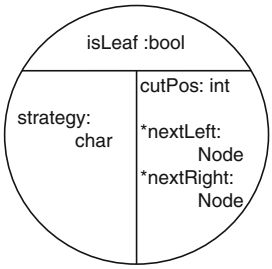


Fig. 4 A node of the genotype

Fourth option works in a different way. It allows us to use "a little pushed" teams, what cannot be obtained as a result of the remaining options. The next option prefers a team, which could start an execution of the task as soon as possible (other teams might be busy). The last option prefers a team whose members could be the first to finish a task (be freed). For the second action only one option is available, namely the list scheduling method.

5.2 Genotype

The genotypes have forms of binary trees corresponding to various procedures of synthesis of phenotypes (target solutions). Every node has the same structure presented on Fig. 4.

The first field *isLeaf* determines a role of the node in a tree. When it is true (the node is a leaf), the strategy for tasks is described in the field named *"strategy"*, which stores an option from Table II. In this case information from the other fields is omitted. When the node is not a leaf, a content of the field *"strategy"* is not important. In this case, *cutPos* contains a number describing which group of tasks should be scheduled by the left node and which one by the right node. Thus, *nextLeft* and *nextRight* must not be null pointers.

The simplest genotype consists of only one node, which is also a leaf and a root. A simple genotype and the corresponding phenotype are presented on Fig. 5.

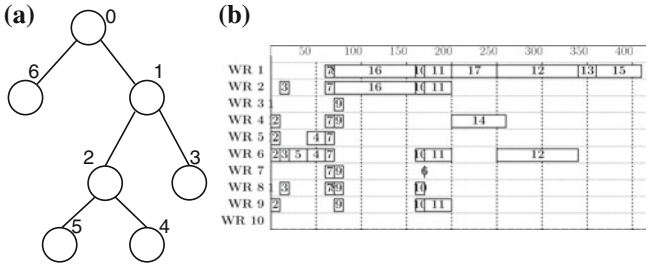


Fig. 5 A simple genotype (**a**) and the corresponding phenotype (**b**)

Table 3 Rules of Mutations

Is a leaf?			
Yes		No	
Draw: switch leaf/node or not?			
Yes	No	Yes	No
Set *isLeaf* as FALSE. If *nextLeft* or *nextRight* is NULL—create a new leaf for it	Draw new strategy	Set isLeaf as TRUE	Change value for a randomly chosen field: *cutPos*, *nextLeft* or *nextRight*

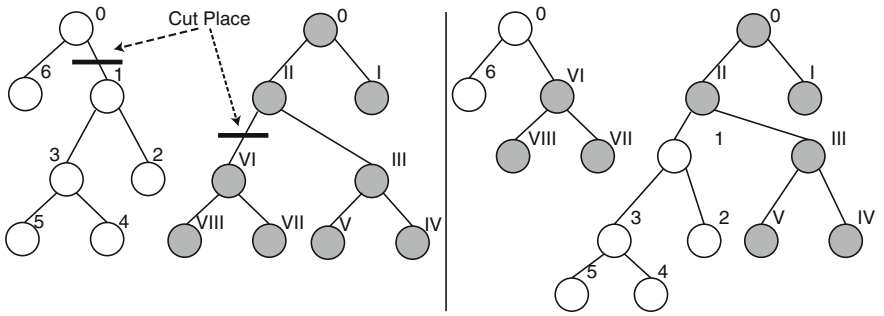


Fig. 6 An example of the crossover

During the evolution, a genotype grows but a size of the genotype tree is limited. If the tree exceeds the maximum size then too long branches are cut off. For example, if the maximum size is 6, every node on the sixth level which has a successor is changed into a leaf, and all its successors are destroyed.

An embryo of the tree could grow as an effect of genetic operators: mutation and crossover. An action associated with the mutation depends on the state of the node and is presented in the Table 3.

The crossover is used to exchange information between two chromosomes. It is necessary to draw a point of cut a tree in both chromosomes. An example of the crossover is presented on Fig. 6.

With every genotype an array is associated. Its size is equal to the number of tasks and contains indexes of teams. If for a task, strategy 'd' is chosen, the team with an index taken from the array is used. At the very beginning of the mutation, a place in the array is randomly chosen. Next a new index is randomly generated. During the crossover, parts of the arrays from both genotypes are swapped.

5.3 Genotype to Phenotype Mapping

The first step in a genotype-to-phenotype mapping is to assign strategies to tasks (that is teams from Table 1 to tasks from Fig. 2b, in the example). This step is illustrated

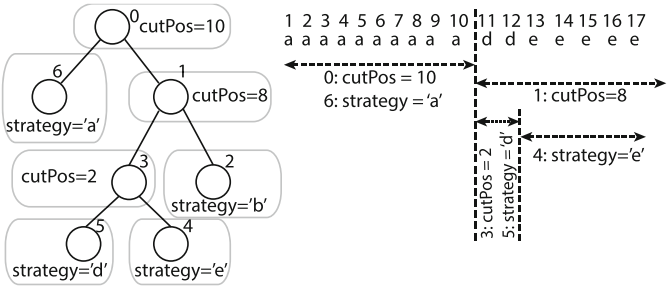


Fig. 7 The first step in genotype-to-phenotype mapping

on Fig. 7. Please note, that node 1 partitions tasks from 11 to 17 into two groups: from 11 to 18 and the rest. Because the first group is out of the range, in fact, there is only one group, which is taken by node 3.

In the second step all tasks without any predecessor, or with predecessors having already assigned teams, are being searched for. For these tasks, teams are assigned according to their strategy. Then the step is repeated as long as there are tasks without assigned teams.

In the third step, the total cost of the solution could be calculated. For this purpose, the cost of each team from the resource library (Table 1) is given.

5.4 Parameters of DGP

During the evolution, new populations of supervisors are created using genetic operations: reproduction, crossover (recombination) and mutation. After the genetic operations are performed on the current population, a new population replaces the current one. The number of individuals in each population is always equal:

$$\Pi = \alpha \prod_{i=1}^{n} s_i \tag{1}$$

where n is the number of tasks, s is the number of teams capable to solve specific problem and α is a constant between 0 and 1. If α is equal to 1, the population has as many individuals as many solutions of the problem exist. The evolution is controlled by parameters β, γ and δ, such that:

- $\Phi = \beta \cdot \Pi$ is the number of individuals created using the reproduction,
- $\Psi = \gamma \cdot \Pi$ is the number of individuals created using the crossover,
- $\Omega = \delta \cdot \Pi$ is the number of individuals created using the mutation and
- $\beta + \gamma + \delta = 1$.

The last condition ensures that each of the created population will have the same number of individuals.

Finally, the selection of the best individuals by a tournament is chosen [25]. In this method, chromosomes (genotypes) are drawn with the same probability in quantity defined as a size of the tournament. From the drawn chromosomes the best one is taken. Hence, the tournament is repeated as many times as the number of chromosomes for a reproduction, crossover and mutation is required. A size of the tournament should not be too high, because the selection pressure is too strong and the evolution will be too greedy. It also could not be too low, because the time of finding any better result would be too long.

5.5 Fitness Function

A fitness function determines the aim of DGP. In the presented approach, two options are possible. In the first one, the cheapest solution which has to be finished before a deadline is searched for. Such fitness function is applied when hard real time constraints have to be satisfied. In the second one, the DGP should find the fastest solution, which does not exceed a given budget. This case concerns systems with soft real-time requirements.

5.6 Self-adaptivity of the Genotype

A direct mapping genotype to phenotype does not guarantee that the target solution will satisfy RT constraints. Thus an adaptive mapping should be applied. For example, if in strategy 'b' (Table 2) the cheapest team does not allow finishing tasks before the deadline, it will be replaced by the next cheapest team, which allows finishing the task before the deadline. This is first contingency of self-adaptivity which is used during the initial scheduling.

The exact self-adaptivity is applied during the system work. After each disruption of the system (i.e. a change of the cost of any resource or a change of the time of execution of any finished task) the genotype to phenotype mapping is performed, but only for tasks that have not been started. The mapping modifies the schedule taking into consideration the RT constraints and minimization of the total cost. Since, for the creation of a new schedule selfsame genotype is again used (the same AS), one may say that AS is self-adaptive i.e. different phenotypes are obtained using the same genotype depending on the system environment.

When an execution time of the task increases, the deadline of the whole project may be exceeded (Fig. 8). Let us assume that for an example from Fig. 2 the deadline is equal to 700. Figure 8a presents a sample phenotype that satisfies the deadline. Suppose that the time of execution of task T3 increases by 60 % than it was planned. Thus, the whole project could not be finished on time without rescheduling (since

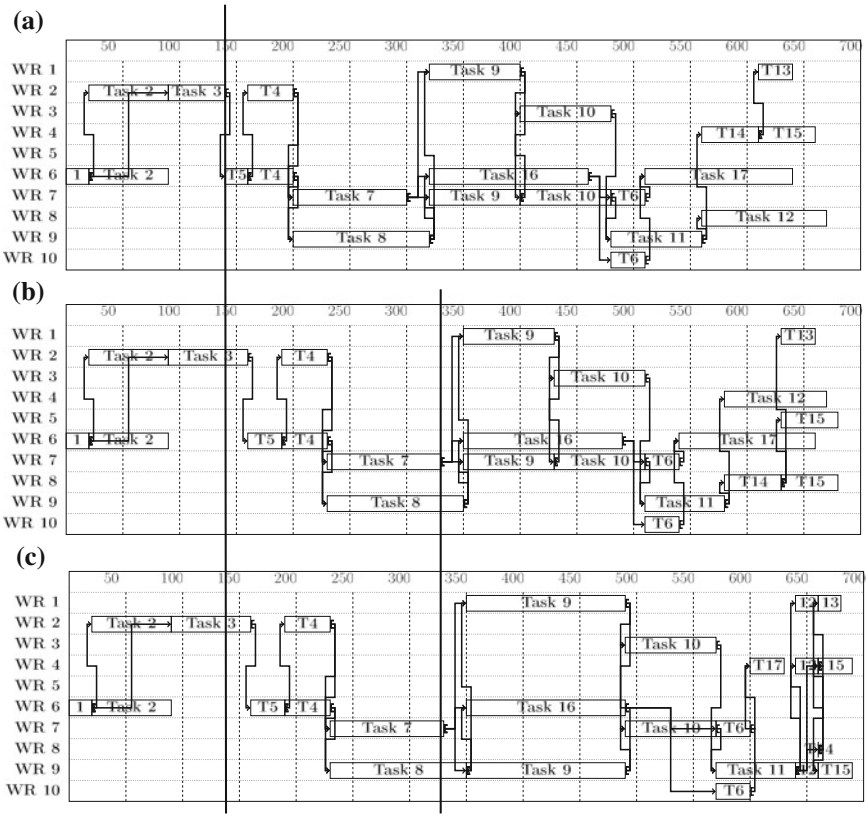


Fig. 8 The self-adaptivity: original phenotype (**a**) and its rescheduling after the first (**b**) and the second (**c**) changes

all successive tasks will be delayed by 30, and the deadline will be exceeded by 10 time units). Therefore, the self-adaptivity starts working. The genotype to phenotype mapping is performed again, for tasks T4–T17. Finally, tasks T12, T13, T14 and T15 are reallocated and rescheduled, and a makespan satisfying RT constraint is obtained (Fig. 8b).

In real world, it is also possible, that some resources are temporarily unavailable. For example, a labourer could be on sick leave or a tool could be in a service. Such a case is presented on Fig. 8c. Suppose that after finishing the task T7, the resource WR7 is temporarily unavailable. Then task T9 has to be executed by another team and the rescheduling of all successive tasks is necessary. The self-adaptivity of the SAAS works also in such cases.

6 Computational Experiments

The procedure was evaluated with the help of the example from Fig. 1. The deadline was equal 700 time units. The SAAS and the initial solution were generated using DGP. During the experiments, the following values of genetic parameters were used:

- the evolution was stopped after 100 generations,
- each experiment was repeated 7 times,
- parameter α was equal to $7 \cdot 10^{-30}$, thus the population size was equal to 102,
- tournament size was equal to 10,
- parameter δ, defining the number of mutants in each generation, was equal to 0.2,
- the crossover was applied for creation of 60 % genotypes ($\gamma = 0.6$),
- $\beta = 0.2$, i.e. in each generation, 20 % of individuals were created using reproduction.

All genetic parameters were adjusted by experimentally tuning the DGP. The convergence of the DGP is illustrated on Fig. 9. The evolution quickly found the best solution. After 30 generations in the worst case.

Next, self-adaptivity of the SAAS was evaluated. Times of processing for some randomly selected tasks were increased. In this way task delays were modeled. Single, double and triple delays were considered. The SAAS started to reschedule after finishing the last delayed task. The results are presented in Table 4. Only in two cases the rescheduling was not required. In all other cases the self-adaptivity of the SAAS was effective and after rescheduling the RT constraint was satisfied. The adaptation to new circumstances caused the cost increase.

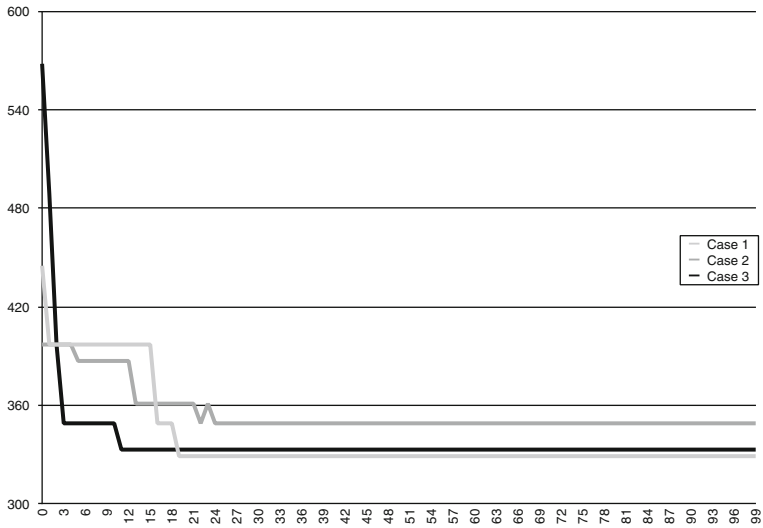


Fig. 9 Progress of the evolution of three specimens

Table 4 Self adaption for different delays

Case	Delay	Time without rescheduling	Time after rescheduling	Timeout (%)	Cost
0	(none)	674	–	0.0	333
1	T3 +60 %	701	682	0.14	407
2	T8 +32 %	712	696	1.71	342
3	T10 +27 %	696	696	0.00	333
4	T14 +66 %	702	689	0.28	381
5	T2 +18 %, T5 +33 %	694	694	0.00	333
6	T3 +23 %, T9 +17 %	701	678	0.14	407
7	T7 +54 %, T16 +42 %	708	697	1.14	337
8	T11 +18 %, T14 +47 %	714	691	2.00	390
9	T2 +18 %, T3 +41 %, T9 +37 %	736	691	5.14	338
10	T3 +32 %, T6 +11 %, T14 +52 %	718	695	2.57	390
11	T7 +18 %, T9 +44 %, T10 +15 %	738	692	5.43	390
12	T4 +46 %, T5 +62 %, T11 +27 %	729	700	4.14	390

Table 5 Self adaption for different extra times

Case	Time decrease	Time	Cost	Cost without rescheduling
0	(none)	674	333	333
1	T3: −50 %	651	333	333
2	T16: −50 %	674	333	333
3	T9: −50 %	700	533	333
4	T4: −50 %	674	333	333
5	T3: −50 %	651	333	333
6	T9: −50 %, T4: −50 %	651	333	333
7	T1:−50 %, T2: −50 %, T10: −50 %	683	329	333
8	T3: −50 %, T9: −50 %, T16: −50 %	696	377	333
9	T3: −50 %, T4: −50 %, T9: −50 %, T16: −50 %	686	329	333
10	T1–T16: time = 1	62	329	333

In real life systems the processing of some tasks may take less time than it was expected. Primarily this happens when the worst case analysis was used to estimate the complexity of tasks. The laxity may cause possibilities for the additional optimization, i.e. reduction of the cost. We may use the SAAS to perform such optimization. Table 5 presents the results obtained for systems where execution times of some tasks were shortened.

Table 6 Self adaption for different rises of costs

Case	Cost increase for previous chosen teams	Time	Cost	Cost without rescheduling
0	(none)	674	333	333
1	T2: +50 %	674	345	345
2	T5: +100 %	674	359	359
3	T7 T16: + 100 %	697	524	374
4	T3: + 100 %, T11: +100 %	657	353	369
5	T16: + 180 %	674	336	344
6	T3: + 75 %, T7: + 100 %	696	380	359
7	T4: + 120 %, T10: + 90 %	697	373	388
8	T9: + 100 %, T12: + 110 %	680	416	361
9	T8: + 100 %, T9: + 75 %, T11: + 120 %	696	632	378
10	T3: +80 %, T6: +120 %, T9: + 50 %	692	459	366
11	T2: + 120 %, T3: +120 %, T7: + 80 %	696	403	381

Table 7 Self adaption for different decreases of costs

Case	Cost increase	Time	Cost	Cost without rescheduling
0	(none)	674	333	333
1	T10: −50 %	697	318	333
2	T8: −75 %	695	354	333
3	T2: − 70 %	691	374	333
4	T12: −60 %	658	325	333
5	T9: −70 %	669	323	333
6	T10: −60 %	697	310	327[a]
7	T9: −50 %, T15: −70 %	691	325	333
8	T16: − 50 %, T8: −80 %	650	326	333
9	T6: − 50 %, T15:−70 %	674	322	333
10	T2: −70 %, T9: − 50 %, T12: −60 %, T16: −50 %	522	310	333

[a] In this case, both previous and current selected teams use the same resource for which the cost has fallen

The SAAS tries to reduce the cost whenever it is possible. However, the successes were rare (cases 7 and 9). To check the reason, in case 10 a time equal to 1 was assigned to each of the teams. The result means that the initial schedule was enough good, and therefore further optimization was very difficult. In cases 7 and 9 the best schedules were found. In cases 3 and 8 the rescheduling caused an increase of the cost. The SAAS is too greedy to deal with such cases. The problem may be fixed by adding more sophisticated options for the fitness function (Table 2).

During the system work, costs of the resources may also change. When a cost of allocated resource increases, the SAAS should try to find a cheaper resource and reschedule tasks, if it is necessary. Similarly, the SAAS should find benefits from a

decrease of costs of unused resources. The results of adaptation of the SAAS to the increase of costs are presented in Table 6.

Results from Table 6 show that the SAAS found better solutions only in 3 cases, while in 6 cases the attempt of reallocation and rescheduling led to worse solution. Like in the previous case (Table 5), an adaptivity of the SAAS features may be improved by limiting the greed of the supervisor options.

Table 7 presents results of reaction of the SAAS to the cost decrease. Similarly to the previous experiments (above Tables 5 and 6) the SAAS in few cases is too greedy. But in most cases better solutions were found.

7 Conclusions

The presented method of synthesis of the SAAS extends the idea of AS presented in [23]. The SAAS is created automatically using developmental genetic programming. It reacts to any changes of execution times of tasks or of costs of resources by reallocating resources and task rescheduling. In this way any violation of real-time constraints may be avoided and/or a cost of the system may be reduced.

The procedure was applied to create the SAAS managing teams which are able to solve tasks. The teams may share members. A team is able to start working when all of its members are idle. For this reason, it is possible, that choosing only the fastest teams do not yield the fastest solution. From the other hand, choosing only the cheapest teams one could cause to the deadline is exceeded.

Experimental results showed that the procedure is efficient. The rescheduling is performed by creation of the new solution from the same genotype, thus it does not require repeating the evolution and may be done in real-time. In all cases the SAAS adapted the RT MS to dynamic changes, and found schedules that meet real-time requirements. However, the results also showed that the SAAS might be less greedy, as far as the cost minimization is concerned.

References

1. C. Wei, P. Liu, Y. Tsai, Resource-constrained project management using enhanced theory of constraint. Int. J. Proj. Manag. **20**(7), 561–567 (2002). http://dx.doi.org/10.1016/S0263-7863(01)00063-1
2. J. Blazewicz, J.K. Lenstra, A.H.G. Rinnooy Kan, Scheduling subject to resource constraints: classification and complexity. Discret. Appl. Math. **5**, 11–24 (1983). http://dx.doi.org/10.1016/0166-218X(83)90012-4
3. G. Pawiński, K. Sapiecha, Cost-efficient project management based on distributed processing model, in *Proceedings of The 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Belfast (2013). http://dx.doi.org/10.1109/PDP.2013.30
4. R.H. Möhring, A.S. Schulz, F. Stork, M. Uetz, Solving project scheduling problems by minimum cut computations. Manag. Sci. **49**(3), 330–350 (2003). http://dx.doi.org/10.1287/mnsc.49.3.330.12737

5. R.E. Keller, W. Banzhaf, The evolution of genetic code in genetic programming, in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1077–1082 (1999)
6. L.C. Briand, Y. Labiche, A UML-based approach to system testing. Softw. Syst. Model. **1**(1), 10–42 (2002). http://dx.doi.org/10.1007/s10270-002-0004-8
7. J.L.M. Pasaje, M.G. Harbour, J.M. Drake, MAST real-time view: a graphic UML tool for modeling object-oriented real-time systems, in *Proceeding of: IEEE 22nd Real-Time Systems Symposium* (RTSS 2001) (2001). http://dx.doi.org/10.1109/REAL.2001.990618
8. H. Gomaa, *Designing Concurrent, Distributed, and Real-Time Applications with UML* (Addison-Wesley, Boston, 2000)
9. R. Jigorea, S. Manolache, P. Eles, Z. Peng, Modelling of real-time embedded systems in an object-oriented design environment with UML, in *Proceedings. Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pp. 210–213 (2000). http://dx.doi.org/10.1109/ISORC.2000.839532
10. J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, (University of Michigan Press, Ann Arbor) (reprinted, MIT Press, Cambridge, 1992)
11. J. Koza, F.H. Bennett III, D. Andre, M.A. Keane, Evolutionary design of analog electrical circuits using genetic programming, in *Adaptive Computing in Design and Manufacture*, ed. by I.C. Parmee (1998). http://dx.doi.org/10.1007/978-3-540-85857-7_8
12. S. Deniziak, A. Górski, Hardware/software co-synthesis of distributed embedded systems using genetic programming, in *Lecture Notes in Computer Science* (Springer, Berlin, 2008), pp. 83–93. http://dx.doi.org/10.1007/978-3-540-85857-7_8
13. J.R. Koza, R. Poli, Genetic programming, in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Chapter 5, ed. by E. Burke, G. Kendal (Springer, 2005). http://dx.doi.org/10.1007/0-387-28356-0_5
14. J. Alcaraz, C. Maroto, A robust genetic algorithm for resource allocation in project scheduling. Ann. Oper. Res. **102**, 83–109 (2001). http://dx.doi.org/10.1023/A:1010949931021
15. S. Hartmann, An competitive genetic algorithm for resource-constrained project scheduling. Nav. Res. Logist. **45**(7), 733–750 (1998). http://dx.doi.org/10.1002/(SICI)1520-6750(199810)45:7%3C733::AID-NAV5%3E3.3.CO;2-7
16. H. Zhang, H. Xu, W. Peng, A genetic algorithm for solving RCPSP, in *International Symposium on Computer Science and Computational Technology* (2008)
17. S. Hartmann, A self-adapting genetic algorithm for project scheduling under resource constraints. Wiley Period. Inc. Nav. Res. Logist. **49**, 433448 (2002)
18. X. Li, L. Kang, W. Tan, Optimized research of resource constrained project scheduling problem based on genetic algorithms. Lect. Notes Comput. Sci. **4683**, 177–186 (2007). http://dx.doi.org/10.1007/978-3-540-74581-5_19
19. H. Zoulfaghari, J. Nematian, N. Mahmoudi, M. Khodabandeh, A new genetic algorithm for the RCPSP in large scale. Int. J. Appl. Evol. Comput. **4**(2), 29–40 (2013). http://dx.doi.org/10.4018/jaec.2013040103
20. S. Hartmann, D. Briskorn, A survey of variants and extensions of the resource-constrained project scheduling problem, Eur. J. Oper. Res.: EJOR **207**(1)(16.11), 1–15 (Elsevier, Amsterdam, 2010). http://dx.doi.org/10.1016/j.ejor.2009.11.005
21. M. Al-Fawzan, M. Haouari, A bi-objective model for robust resourceconstrained project scheduling. Int. J. Prod. Econ. **96**, 175–187 (2005)
22. K.M. Calhoun, R.F. Deckro, J.T. Moore, J.W. Chrissis, J.C.V. Hove, Planning and re-planning in project and production scheduling. Omega Int. J. Manag. Sci. **30**(3), 155170 (2002)
23. K. Sapiecha, L. Ciopiński, S. Deniziak, An application of developmental genetic programming for automatic creation of supervisors of multi-task real-time object-oriented systems, in Proceedings. Federated Conference on Computer Science and Information Systems (FedCSIS, Warsaw, 2014). http://dx.doi.org/10.15439/2014F208
24. R.V. Binder, *Testing Object-Oriented Systems—Models, Patterns, and Tools* (Addison-Wesley, Reading, 1999)
25. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, Berlin, 1996)

Direct Shooting Method for Optimal Control of the Highly Nonlinear Differential-Algebraic Systems

Paweł Drąg and Krystyn Styczeń

Abstract In the paper the optimal control of highly nonlinear differential-algebraic systems (DAEs) is discussed. The direct shooting method is seen as an efficient tool for control of the complex real-life technological processes, where dynamics and conservation laws are presented. To stabilize the optimization algorithm, the multiple shooting method was proposed. The multiple shooting approach introduces new decision variables and constraints to the problem, but it can preserve the stability of the process, the continuity of the differential state trajectories and enables parallel computation of the mathematical model. The conditions for the frequency of shots, to establish the well-conditioned optimization problem, are considered. The proposed method was tested on the mathematical model of the fed-batch fermentor for penicillin production process, which is a highly nonlinear multistage differential-algebraic system. The numerical simulations were executed in MATLAB environment using Wroclaw Center for Networking and Supercomputing.

Keywords Optimal control · Differential-algebraic systems · Multiple shooting method · Nonlinear programming

1 Introduction

In recent years, many efforts have been devoted to the model-based dynamical optimization, especially in such fields like chemical and mechanical engineering, bioengineering and biotechnology. The main common feature of these mathematical models is the presence of dynamics and conservation laws.

The mathematical modeling of such processes, which operate under transient conditions, leads to systems with both differential and algebraic constraints. However,

P. Drąg (✉) · K. Styczeń
Department of Control Systems and Mechatronics, Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland
e-mail: pawel.drag@pwr.edu.pl

K. Styczeń
e-mail: krystyn.styczen@pwr.edu.pl

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_5

real-life industrial processes may be affected by external signals and actions influencing the dynamics. On the other hand, some intrinsic physical changes can cause discontinuities [5]. As a result, many of these processes can be described as sequences of different sets of differential-algebraic equations, especially, when the multistage systems are considered [10, 12]. In the article the dynamic optimization of both single and multistage DAE systems is considered.

The direct approaches to control DAE systems in open-loop manner transform the original infinite dimensional optimal control problem into a large-scale nonlinear programming (NLP) problem. This is done in two possible ways, either using control vector parametrization (CVP) or complete parametrization of both control and state [1, 4].

In complete parametrization approach, often known as simultaneous direct strategy, the controls and states are parametrized by means of collocation on finite elements. This full discretization results in a large-scale NLP problem, which can have thousands of variables. It has the advantage of avoiding the solution of the model for different values of decision variables. This method was extensively illustrated on control of the high-impact polystyrene reactor [13, 14].

On the other hand, there is the direct shooting method, which combines the advantages of the efficient DAE and NLP *solvers*, gives the possibility of parallel computations and improves the stability of the system [19]. Like in the simultaneous approach, direct shooting method enables us to use the sparse representations of the matrices.

Also, the authors want to discuss the advanced methods of control and optimization of the highly nonlinear systems with differential-algebraic constraints.

The article is constructed as follows. In Sect. 2 the formulation of dynamical systems using descriptor equations was discussed. The direct shooting formulation was presented in Sect. 3. Sequential dynamical approach for control of the differential-algebraic systems was presented in Sect. 4. Then results of the numerical simulation on the highly nonlinear fed-batch fermentor for penicillin production process was reported. The presented considerations were concluded in Sect. 6.

In the article was used the same notation as in [11].

2 Descriptor Equations and Dynamical Systems

The real-life technological systems with dynamics and conservation laws can be described in the general form

$$F\big(x(t), z(t), u(t)p, t\big) = 0, \tag{1}$$

where $x(t) \in \mathcal{R}^{n_x}$ is a differential state, $z(t) \in \mathcal{R}^{n_z}$ is an algebraic state, $u(t) \in \mathcal{R}^{n_u}$ the control function, $p \in \mathcal{R}^{n_p}$ is a vector of parameters constant in the time. Then the vector-valued nonlinear function $F : \mathcal{R}^{n_x} \times \mathcal{R}^{n_z} \times \mathcal{R}^{n_u} \times \mathcal{R} \to \mathcal{R}^{n_F}$ is considered.

The *fully-implicit* form presented in Eq. (1) is very general and can be difficult to analyze in all possible situations, in which definitely different properties can be observed.

On the other hand, when only dynamical features of the systems are under considerations, the description using ordinary differential equation is enough

$$\dot{x}(t) = G\big(x(t), u(t)p, t\big), \tag{2}$$

but in such manner some interesting relations between variables and their physical interpretations can be lost.

Complex technological processes presented in a descriptor form are more convenient to the analysis than presented like in Eq. (1), enable compact description of the systems with complicated structure, include both dynamics and conservation laws and allow advanced numerical methods for control and optimization. In wide range of applications, the process can be modeled in the conventional descriptor form

$$\mathcal{M}_{\mathcal{D},\mathcal{A}}\dot{x}(t) = \mathcal{F}\big(x(t), z(t), u(t)p, t\big). \tag{3}$$

It is known, that matrix $\mathcal{M}_{\mathcal{D},\mathcal{A}}$ is singular and can be seen as $\mathcal{M}_{\mathcal{D},\mathcal{A}} = \begin{bmatrix} \mathcal{M}_{\mathcal{D}} & 0 \\ 0 & \mathcal{M}_{\mathcal{A}} \end{bmatrix}$, where $\mathcal{M}_{\mathcal{D}}$ is the square submatrix of $\mathcal{M}_{\mathcal{D},\mathcal{A}}$ with the biggest size, such that $\det \mathcal{M}_{\mathcal{D}} \neq 0$. In this way Eq. (2) one can rewrite as

$$\begin{bmatrix} \mathcal{M}_{\mathcal{D}} & 0 \\ 0 & \mathcal{M}_{\mathcal{A}} \end{bmatrix} \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} \mathcal{F}_1\big(x(t), z(t), u(t)p, t\big) \\ \mathcal{F}_2\big(x(t), z(t), u(t)p, t\big) \end{bmatrix} \tag{4}$$

The appropriate sparse structure of the matrix $\mathcal{M}_{\mathcal{D},\mathcal{A}}$ enables us to distinguish in the considered system the differential and algebraic parts. This leads to the following system

$$\begin{aligned} \mathcal{B}(\cdot)\dot{x}(t) &= \mathcal{F}_1\big(x(t), z(t), u(t), p, t\big) \\ 0 &= \mathcal{F}_2\big(x(t), z(t), u(t), p, t\big), \end{aligned} \tag{5}$$

where the matrix $\mathcal{B}$ is nonsingular.

3 Multistage Systems and Direct Shooting Approach

The optimal control of the technological processes is aimed at the minimization of the performance cost function

$$\min_{(u(t),x(t),z(t),p)} \int_{t_0}^{t_f} L\big(x(t), z(t), u(t), p\big)dt + E\big(x(t_f)\big), \tag{6}$$

subject to a system of the index-one differential-algebraic equations (DAEs)

$$\begin{aligned} \mathcal{B}(\cdot)\dot{x}(t) &= \mathcal{F}_1\big(x(t), z(t), u(t), p\big) \\ 0 &= \mathcal{F}_2\big(x(t), z(t), u(t), p\big), \end{aligned} \tag{7}$$

which in many applications could have some highly nonlinear components.

The initial values of the differential and algebraic states and values for the system parameters are prescribed

$$x(t_0) = x_0, \tag{8}$$

$$p(t_0) = p_0. \tag{9}$$

In addition, the terminal constraints

$$r_1\big(x(t_f), p\big) = 0, \quad r_2\big(x(t_f), p\big) \leq 0, \tag{10}$$

as well as the state and control inequality constraints

$$h(x(t), z(t), u(t), p) \leq 0 \tag{11}$$

have to be satisfied.

This description is valid only for single-state processes. There is a quite different situation, when the multistage technological systems are considered, because each stage can be described by different sets of the nonlinear differential-algebraic equations.

Let us assume, that there are N stages in the complex industrial process and there is an independent variable t, for example time or length of the chemical reactor.

For a suitable partition of the time horizon $[t_0, t_f]$ into N subintervals $[t_i, t_{i+1}]$ with

$$t_0 < t_1 < \cdots < t_N = t_f, \tag{12}$$

the control function $u(t)$ is discretized in an appropriate way. The most frequently encountered representations of the control function are a piecewise constant, a piecewise linear or a polynomial approximation of the control [20]. If the control function is parametrized as a piecewise constant vector function, then

$$u(t) = u^l \tag{13}$$

for $t \in [t_{l-1}, t_l]$, $\quad l = 1, \ldots, N$.

By the multiple shooting method, the DAE model is parametrized in some sense too. The solution of the DAE system is decoupled on the N intervals $[t_l, t_{l+1}]$. In this way the initial values s_x^l and s_z^l of the differential and algebraic states at times t_i are introduced as the additional optimization variables. The trajectories $x(t)$ and

$z(t)$ are obtained as a set of trajectories $x^l(t)$ and $z^l(t)$ on each interval $[t_{l-1}, t_l]$. The mentioned trajectories $x^l(t)$ and $z^l(t)$ are the solutions of an initial value problem

$$\begin{aligned} \mathcal{B}^l(\cdot)\dot{x}(t) &= \mathcal{F}_1^l\big(x^l(t), z^l(t), u^l(t), p\big) \\ 0 &= \mathcal{F}_2^l\big(x^l(t), z^l(t), u^l(t), p\big) + \\ &\quad + \alpha^l(t_l) g^l\big(s_x^l, s_z^l, u^l, p\big) \\ t &\in [t_{l-1}, t_l], \qquad l = 1, \ldots, N. \end{aligned} \tag{14}$$

The relaxation parameter $\alpha^l(t_l)$ was introduced to allow an efficient DAE solution for the initial values of state trajectories s_x^l, s_z^l and controls u^l, that may temporarily violate the consistency conditions. In this way, the trajectories $x^l(t)$ and $z^l(t)$ on the interval $[t_{l-1}, t_l]$ are the functions of the initial values, controls and parameters s_x^l, s_z^l, u^l, p.

The integral part of the performance cost function is evaluated on each interval independently

$$\begin{aligned} &\textstyle\int_{t_0}^{t_1} L^1\big(x^1(t), z^1(t), u^1(t), p\big)dt + \cdots + \\ &+ \textstyle\int_{t_{N-1}}^{t_N} L^N\big(x^N(t), z^N(t), u^N(t), p\big)dt + \\ &\qquad + E\big(x(t_N)\big) = \\ &= \textstyle\sum_{l=1}^{N} \int_{t_{l-1}}^{t_l} L^l\big(x^l(t), z^l(t), u^l(t), p\big)dt + \\ &\qquad + E\big(x(t_N)\big). \end{aligned} \tag{15}$$

The multiple shooting method is often known as the *parallel shooting method*. It means, that DAE system can be solved parallel for each time interval $[t_{l-1}, t_l]$ [3].

The parametrization of the optimal control problem of the multistage DAE systems using the multiple shooting approach and a piecewise constant control representation leads to the following nonlinear programming problem

$$\begin{aligned} &\textstyle\sum_{l=1}^{N} \int_{t_{l-1}}^{t_l} L^l\big(x^l(t), z^l(t), u^l(t), p\big)dt + \\ &+ E\big(x(t_N)\big) = \Phi(s_x^l, s_z^l, u^l, p) = \Phi(\chi) \to \min \end{aligned} \tag{16}$$

subject to the continuity conditions

$$s_x^l = x^{l-1}(t_{l-1}), \qquad l = 2, \ldots, N, \tag{17}$$

the consistency conditions

$$0 = g^l(s_x^l, s_z^l, u^l, p), \qquad l = 1, \ldots, N, \tag{18}$$

the control and path constraints imposed pointwise at the multiple shooting nodes

$$h^l(s_x^l, s_z^l, u^l, p) \leq 0, \qquad l = 1, \ldots, N, \tag{19}$$

the terminal constraints

$$r_1(s_x^l, s_z^l, p) = 0, \quad r_2(s_x^l, s_z^l, p) \leq 0, \tag{20}$$

the lower and upper bounds on the decision variables

$$\chi_L \leq \chi \leq \chi_U, \tag{21}$$

$$\chi = [s_x^1, \ldots, s_x^N, s_z^1, \ldots, s_z^N, u^1, \ldots, u^N, p]^T, \tag{22}$$

$$\chi_L = [s_{x,L}^1, \ldots, s_{x,L}^N, s_{z,L}^1, \ldots, s_{z,L}^N, u_L^1, \ldots, u_L^N, p_L]^T, \tag{23}$$

$$\chi_U = [s_{x,U}^1, \ldots, s_{x,U}^N, s_{z,U}^1, \ldots, s_{z,U}^N, u_U^1, \ldots, u_U^N, p_U]^T \tag{24}$$

and to the DAE system in each interval

$$\begin{aligned} \mathcal{B}^l(\cdot)\dot{x}(t) &= \mathcal{F}_1^l\big(x^l(t), z^l(t), u^l(t), p\big) \\ 0 &= \mathcal{F}_2^l\big(x^l(t), z^l(t), u^l(t), p\big) + \\ &\quad + \alpha^l(t_l) g^l(s_x^l, s_z^l, u^l, p), \\ t &\in [t_{l-1}, t_l], \qquad l = 1, \ldots, N. \end{aligned} \tag{25}$$

Remark 1 It is assumed, that matrix $\mathcal{B}^l$ in each stage has a constant rank.

The application of the *multiple shooting method* enables us to control and optimize the multistage technological processes. It is a one of ways, to control of the unstable processes too. The stabilization of the unstable dynamical modes can be carried out by adjustment the frequency of the shots to the dynamics properties of the system.

Additional shots incorporate additional equality constraints into the nonlinear programming problem. The equality constraints, which provide continuity of the differential state trajectories, have often a simple form and the Karsuh-Kuhn-Tucker (KKT) system becomes more sparse.

The second type of the constraints represents technological constraints on the state variables. The mentioned method can preserve fulfillment of such constraints only at the beginning of each interval, and at the end of each interval by the active equality constraints on the continuity of the state trajectories [21].

The starting point for solving differential-algebraic equations are the consistent initial conditions. They are difficult to obtain especially, when the multistage systems are considered. Addition of the *dumping parameter* enables us to solve the DAE systems efficiently without other methods, e.g. penalty terms in the objective function.

The challenge is the frequency selection for the multiple shooting method. The point are chosen by the structure of the system, nature of the process or experience of the researchers. To obtain the appropriate shooting frequency, the following remarks and theorem can be helpful.

Remark 2 The approximately solution of the multiscale dynamical systems, known as *singularly perturbed problem*

$$\begin{aligned} \dot{x}(t) &= \mathcal{F}_1\big(x(t), z(t), u(t), p\big) \\ \varepsilon \dot{z}(t) &= \mathcal{F}_2\big(x(t), z(t), u(t), p\big) \end{aligned} \tag{26}$$

can be obtained by solving the following DAE system

$$\begin{aligned} \dot{x}(t) &= \mathcal{F}_1\big(x(t), z(t), u(t), p\big) \\ 0 &= \mathcal{F}_2\big(x(t), z(t), u(t), p\big) \end{aligned} \tag{27}$$

with consistent initial conditions.

It means, that more number of shots should not be dependent on a single state variables, which oscillates around small values.

In highly nonlinear dynamical processes a fast transition phase can be observed. The transition is continuous and it is an especially interesting area, where a larger number of shots can be required.

Theorem 1 *If the state trajectories are monotonic in the time interval* $[t_l, t_{l+1}]$*, then the number of shots can be used to specify the Lipschitz constant in each interval* l

$$\dot{x}(t_l) \leq \frac{|x^{l+1} - x^l|}{|t_{l+1} - t_l|} \leq K^l, \tag{28}$$

where x^l *and* x^{l+1} *are the values of the shots at the beginning and at the end of the time interval, respectively.*

In this way, one can preserve the state trajectories to come into instability regions. The interval, when the state trajectories are monotonic, are often known from the process specification.

On the other hand, it means, that the bigger number of the shots do not improve the stability of the solution. If the difference in the state trajectory $|x^{l+1} - x^l|$ is relatively small, but the shooting frequency is high, so $|t_{l+1} - t_l|$ is small, then the high fluctuations can be observed.

4 Sequential Dynamical Approach

The sequential dynamical approach is a part of the trend "discretize then optimize". This approach has the advantage of directly finding good approximate solutions, which are feasible to the pointwise state constraints. From literature it is known, that

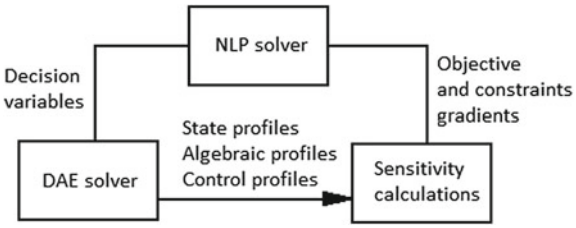


Fig. 1 Sequential dynamic optimization strategy [4]

this formulation requires an accurate level of discretization of the both control and state profiles.

As it was mentioned, the time domain was partitioned into smaller time elements and the DAE models are integrated separately in each element. The control variables are represented as a piecewise polynomial, hence optimization is performed with respect to the polynomials coefficients. The sensitivities in each element are obtained for both the control variables and the initial conditions of the states, which are the decision variables.

The equality constraints are added to the nonlinear program in order to link the elements and ensure that the states remain continuous over time. The inequality constraints for the states and the control can be imposed directly at the grid points, although the constraints on the state profiles may be violated between the grid points.

There is a sketch of sequential dynamic optimization strategy on the Fig. 1. This approach consists of three main elements. There are the NLP *solver*, the DAE *solver* and the sensitivity calculations. All of this parts, especially from parallel calculations point of view, were described and commented in this section.

4.1 NLP Solver

Consider the formulas (16)–(25) rewritten in the form

$$\min_{\chi} \Phi(\chi) \tag{29}$$

subject to

$$c_{eq}(\chi) = 0, \quad c_{in}(\chi) \leq 0. \tag{30}$$

An optimal solution of (29) and (30) satisfies the first order necessary conditions

$$\begin{aligned} \nabla_{\chi} \mathcal{L}(\chi^*, \lambda^*, s^*) &= 0, \\ s^* &\geq 0, \\ \lambda_{eq}^* &\geq 0, \\ {\lambda_{eq}^*}^T s^* &= 0, \end{aligned} \tag{31}$$

where s is a vector of slack variables, $\lambda = [\lambda_{eq}^T, \lambda_{in}^T]^T$ and

$$\mathcal{L}(\chi, \lambda, s) = \Phi(\chi) + \lambda^T \begin{bmatrix} c_{eq}(\chi) \\ c_{in}(\chi) + s \end{bmatrix}. \tag{32}$$

Here $\mathcal{L}$ denotes the Lagrange function for the problems (29) and (30), and λ_{eq}, λ_{in} are the Lagrange multipliers associated with the equality and inequality constraints respectively.

Sequential Quadratic Programming methods [8, 18] can be derived from the application of the Newton's formula to the problem of determining a stationary point of $\mathcal{L}$, which satisfies all the constraints. This results in a sequence of iterates $\{(\chi_k, \lambda_k, s_k)\}$, in which the next iterate is determined by solving at the kth iteration the system

$$\nabla^2_{\chi\chi}\mathcal{L}(\chi_k, \lambda_k, s_{k+1}) \begin{bmatrix} d_k \\ \delta_k \end{bmatrix} = -\nabla_\chi \mathcal{L}(\chi_k, \lambda_k, s_{k+1}) \tag{33}$$

$$\begin{aligned} s_{k+1} &\geq 0, \\ \lambda_{eq,k} + \delta_{eq,k} &\geq 0, \\ (\lambda_{eq,k} + \delta_{eq,k})^T s_{k+1} &= 0 \end{aligned} \tag{34}$$

to obtain (d_k, δ_k, s_{k+1}) and to determine

$$\chi_{k+1} = \chi_k + d_k, \tag{35}$$

$$\lambda_{k+1} = \lambda_k + \delta_k. \tag{36}$$

To limit the step size, the iteration is usually formulated as an equivalent QP subproblem

$$\min_{d_k} \nabla_\chi \Phi(\chi_k)^T d_k + \frac{1}{2} d_k^T B_{\mathcal{L}}(\chi_k, \lambda_k) d_k \tag{37}$$

subject to

$$\nabla_\chi c_{eq}(\chi_k)^T d_k + c_{eq}(\chi_k) = 0, \tag{38}$$

$$\nabla_\chi c_{in}(\chi_k)^T d_k + c_{in}(\chi_k) \leq 0. \tag{39}$$

The solution d_k determines a search direction, which can be used to determine the next iterate χ_{k+1}. In Eq. (37), $B_{\mathcal{L}}$ is either taken to be the Hessian of the Lagrange function $\mathcal{L}$

$$B_{\mathcal{L}}(\chi, \lambda) = \nabla^2_{\chi\chi}\mathcal{L} = \nabla^2_{\chi\chi}\Phi(\chi) \sum_i \lambda_i \nabla^2_{\chi\chi} c_i(\chi, \lambda), \tag{40}$$

or $B_{\mathcal{L}}$ is defined as a suitable approximation of $\nabla^2_{\chi\chi}\mathcal{L}$. The use of an approximate Hessian for $B_{\mathcal{L}}$ enables us to avoid the calculation of the second derivatives of the performance cost function and the constraints. In this way, it allows to reduce the

computational complexity, but it reduces the rate of the convergence near the solution too.

The proposed approach enables us to use the full machinery of NLP *solvers*, especially SQP codes for medium and large-scale problems are desirable [6, 17]. For nonlinear programming problems with a few hundreds of variables *fmincon solver* can be treated as quite suitable. The mentioned *solver* is coupled to the MATLAB environment and applies line search and BFGS updates of the Hessian.

4.2 Sensitivity Calculations

The place, where appropriate using of parallel computing can improve the time performance, is connected with calculations of sensitivities. Firstly, SQP algorithm needs the gradients of both the objective and the constraints functions. Estimation of the objective function and constraint functions can be given analytically or obtained using the finite difference derivative approximation [9].

Consider a subroutine, that estimates the gradient of the objective function and the constraint functions. This calculation involves the computing of function values at points near the current solution χ_k.

In this situation there is a possibility to compute successive elements of the gradient vector parallel. It is worthy to note, that these elements are independent and computing of each element needs the objective or constraint function evaluation [16].

Let us say, that the considered model is very complicated and the calculation of each objective function is time-consuming. Always one has to decide, if it is better to use parallel computing for the gradient estimation or parallel evaluation of the objective function.

One of the results of the multiple shooting approach is an increase in the size of the NLP problem. There are additional variables and constraints, which were introduced for each segment. The number of NLP variables and constraints for a multiple shooting application is $n = (n_x + n_z)(N - 1)$, where n_x is the number of dynamic variables x, n_z is the number of algebraic variables z, and $(N - 1)$ is the number of segments. It is important to note, that the Jacobian matrix, which is needed to compute the Newton search direction, is sparse. Only $(N - 1)(n_x + n_z)^2$ elements in this matrix are nonzero of a possible $[(N - 1)(n_x + n_z)]^2$. Thus, the percentage of nonzeros is proportional to $\frac{1}{N-1}$ indicating that the matrix get sparser as the number of intervals grows.

4.3 DAE Solver

The range of applications of this method depends on the applied DAE *solver*. The multiple shooting method allows to use parallel solving systems, hence the calculation

of the objective function can be faster, especially, when the function is complicated. One of the most known algorithm for solving index-1 DAE systems is the *backward differentiation formula*, known as the implicit Euler method [7].

5 Case Study: Optimal Control of a Fed-Batch Fermentor for Penicillin Production

This problem considers a fed-batch reactor for the production of penicillin [2]. We consider here the free terminal time version where the objective is to maximize the amount of penicillin using the feed rate as the control variable. The mathematical statement of the free terminal time problem is as follows.

Find $u(t)$ and t_f over $t \in [t_0, t_f]$ to maximize

$$J = x_2(t_f) \cdot x_4(t_f) \tag{41}$$

subject to differential-algebraic system

$$\dot{x}_1(t) = z_1 x_1 - u\left(\frac{x_1}{500x_4}\right), \tag{42}$$

$$\dot{x}_2(t) = z_2 x_1 - 0.01x_2 - u\left(\frac{x_2}{500x_4}\right), \tag{43}$$

$$\dot{x}_3(t) = -z_1\frac{x_1}{0.47} - z_2\frac{x_1}{1.2} - x_1\frac{0.029x_3}{0.0001 + x_3} + \frac{u}{x_4}\left(1 - \frac{x_3}{500}\right), \tag{44}$$

$$\dot{x}_4(t) = \frac{u}{500}, \tag{45}$$

$$0 = z_1 - 0.11\left(\frac{x_3}{0.006x_1 + x_3}\right), \tag{46}$$

$$0 = z_2 - 0.0055\left(\frac{x_3}{0.0001 + x_3(1 + 10x_3)}\right), \tag{47}$$

where x_1, x_2 and x_3 are the biomass, penicillin and substrate concentration (g/L), and x_4 is the volume (L). The initial conditions are

$$x(t_0) = [1.5 \quad 0 \quad 0 \quad 7]^T. \tag{48}$$

There are several path constraints for state variables

$$0 \le x_1 \le 40, \tag{49}$$

$$0 \le x_2 \le 25, \tag{50}$$

$$0 \leq x_3 \leq 10. \tag{51}$$

The upper and lower bounds on the only control variable (feed rate of substrate) are

$$0 \leq u \leq 50. \tag{52}$$

The control problem of a fed-batch fermentor for penicillin production was solved with the line-search SQP algorithm combined with multiple shooting method.

At first, the overall time domain was divided into 100 equidistance intervals. It results in 100 differential-algebraic submodels, each of them consists of 4 differential equations and 2 algebraic equations. Initial conditions only for the first stage are known. So, there are 396 decision variables connected with initial values for differential variables and 200 variables, which represents pointwise values of algebraic states. The last decision variables was the duration time of the process.

In the optimization process it was assumed piecewise constant control function u.

The inconsistent and away from the solution initial values for decision variables were as follows

$$\chi_{1,x_{1,2}}, \ldots, \chi_{99,x_{1,100}} = 1.5, \tag{53}$$

$$\chi_{100,x_{2,2}}, \ldots, \chi_{198,x_{2,100}} = 0.0, \tag{54}$$

$$\chi_{199,x_{3,2}}, \ldots, \chi_{297,x_{3,20}} = 0.0, \tag{55}$$

$$\chi_{298,x_{4,2}}, \ldots, \chi_{396,x_{4,20}} = 7.0, \tag{56}$$

$$\chi_{397,h_{1,1}}, \ldots, \chi_{496,h_{1,20}} = 10.0, \tag{57}$$

$$\chi_{497,h_{2,1}}, \ldots, \chi_{596,h_{2,20}} = 10.0, \tag{58}$$

$$\chi_{597,u_1}, \ldots, \chi_{696,u_{20}} = 10.0, \tag{59}$$

$$\chi_{697,t_f} = 110.0[h]. \tag{60}$$

To provide consistent initial values for DAE models and to preserve continuity on the differential state trajectories, 596 pointwise equality algebraic constraints were introduced.

The presented stages are based on technological constraints and assumptions, which reflect a desired performance of the process. To proceed the reactions more effectively, new equal length substages were introduced. They are aimed at fulfill the constraints on the control variables and both the differential and algebraic state trajectories. Partitioning the process into the substages has a positive impact on the stability of the solution.

The solution, with the accuracy 10^{-6} and the final value of the objective function is 87.80 $[g]$. The duration of the penicillin production process is 132.54 h. There are

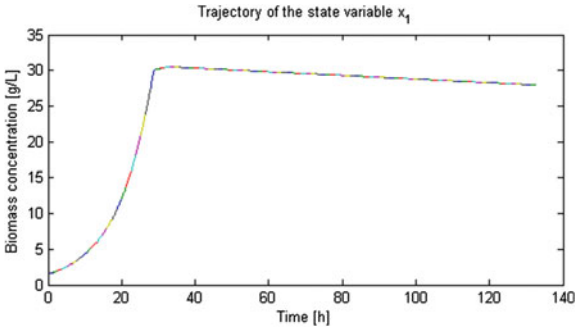


Fig. 2 Trajectory of the biomass concentration (g/L)

the optimal trajectories of the biomass, penicillin concentrations and feed substrate concentration in the Figs. 2, 3 and 4. Trajectory of the volume was presented in the Fig. 5.

Optimal control trajectory can be observed in the Fig. 6. Results presented in this researches are comparable with the solution presented in [2], but fluctuations of the control function are much smaller and do not reach the extreme values.

In the presented approach the multiple shooting method was used. It enables us to solve the dynamic optimization problem with some processors working parallel. The computation time, depending of the number of used processors, were presented in the Table 1. The performance time can be greatly reduced in the algorithms designed for parallel working.

The shooting method preseved the high fluctuations in the state trajectories, because

$$\frac{\max_i |x_{i+1} - x_i|}{|t_{i+1} - t_i|} = \frac{40}{\approx 1.3} \leq 30.77. \tag{61}$$

The shooting frequency about $|t_{i+1} - t_i| \leq 50$ can have a negative impact on the stability of the obtained solution.

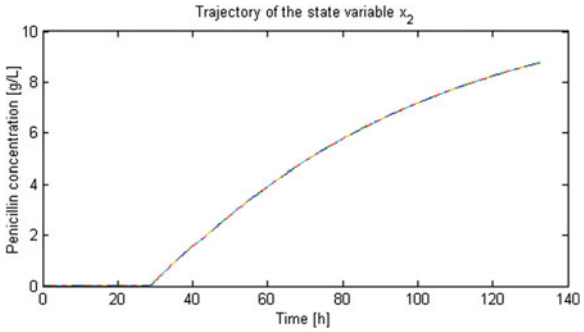


Fig. 3 Trajectory of the penicillin concentration (g/L)

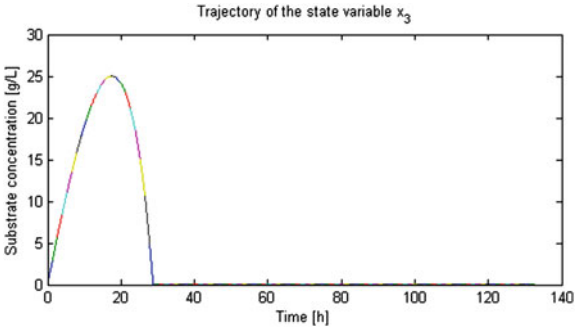


Fig. 4 Trajectory of the feed substrate concentration (g/L)

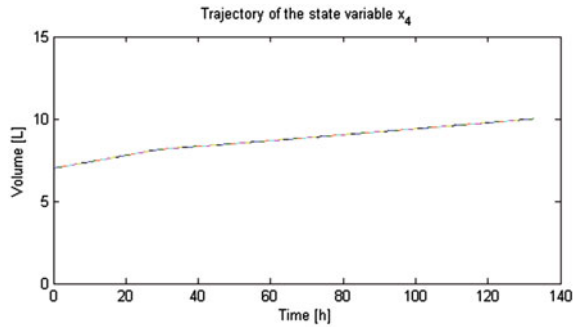


Fig. 5 Trajectory of the volume (L)

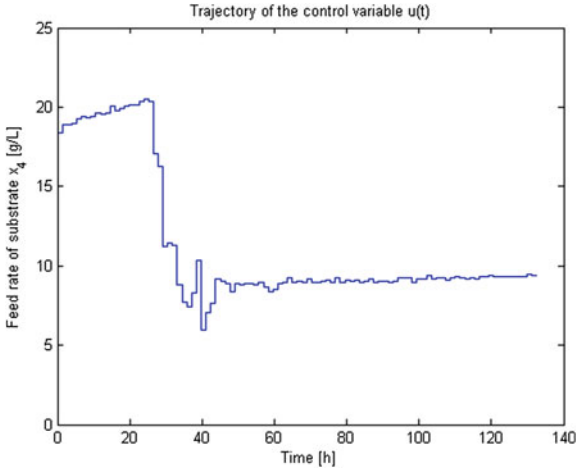


Fig. 6 The optimal control trajectory $u(t)$

Table 1 Computation time for different number of parallel processors

# of processors	Computation time (s)
1	32 936
2	25 000
4	14 707

6 Conclusion

In the article, the optimal control problem of highly nonlinear differential-algebraic systems (DAEs) was presented and discussed. To solve the control problem, a large-scale SQP optimization algorithm with the multiple shooting method was proposed. Because the multiple shooting method introduces a large number of the new decision variables, parallelization method can be successfully used to perform the optimization procedure effectively. The numerical simulations of control problem of model the penicillin production process were performed with the parallel working processors, which enables us to reduce the computation time significantly.

Next research focuses on the effective taking into account a large number of the nonlinear equality constraints in dynamic optimization problems and the new aspects of the constraints aggregation.

Acknowledgments The project was supported by the grant of National Science Centre Poland DEC-2012/07/B/ST7/01216.

References

1. E. Balsa-Canto, V.S. Vassiliadis, J.R. Banga, Dynamic optimization of single and multi-stage systems using a hybrid stochastic-deterministic method. Ind. Eng. Chem. Res **44**, 1514–1523 (2005)
2. J.R. Banga, E. Balsa-Canto, C.G. Moles, A.A. Alonso, Dynamic optimization of bioprocesses: efficient and robust numerical strategies. J. Biotechnol. **117**, 407–419 (2005)
3. J.T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd edn. (SIAM, Philadelphia, 2010)
4. L.T. Biegler, *Nonlinear Programming*, *Concepts, Algorithms and Applications to Chemical Processes* (SIAM, Philadelphia, 2010)
5. L.T. Biegler, S. Campbell, V. Mehrmann, DAEs, Control, and Optimization, in *Control and Optimization with Differential-Algebraic Constraints*, ed. by L.T. Biegler, S. Campbell, V. Mehrmann (SIAM, Philadelphia, 2012)
6. L.T. Biegler, I.E. Grossmann, Retrospective on optimization. Comput. Chem. Eng. **28**, 1169–1192 (2004)
7. K.E. Brenan, S.L. Campbell, L.R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations* (SIAM, Philadelphia, 1996)
8. M. Cannon, Efficient nonlinear model predictive control algorithms. Annu. Rev. Control **28**, 229–237 (2004)
9. M. Caracotsios, W.E. Stewart, Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations. Comput. Chem. Eng. **9**, 359–365 (1985)

10. E.F. Carrasco, J.R. Banga, Dynamic optimization of batch reactors using adaptive stochastic algorithms. Ind. Eng. Chem. Res. **36**, 2252–2261 (1997)
11. M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, F. Allgöwer, Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. J. Process Control **12**, 577–585 (2002)
12. P. Drąg, K. Styczeń, A two-step approach for optimal control of kinetic batch reactor with electroneutrality condition. Przegląd Elektrotechniczny **6**(88), 176–180 (2012)
13. A. Flores-Tlacuahuac, L.T. Biegler, E. Saldivar-Guerra, Dynamic optimization of hips open-loop unstable polymerization reactors. Ind. Eng. Chem. Res. **44**, 2659–2674 (2005)
14. A. Flores-Tlacuahuac, S.T. Moreno, L.T. Biegler, Global optimization of highly nonlinear dynamic systems. Ind. Eng. Chem. Res. **47**, 2643–2655 (2008)
15. I.E. Grossmann, L.T. Biegler, Part II. Future perspective on optimization. Comput. Chem. Eng. **28**, 1193–1218 (2004)
16. A. Hartwich, K. Stockmann, C. Terboven, S. Feuerriegel, W. Marquardt, Parallel sensitivity analysis for efficient large-scale dynamic optimization. Optim. Eng. **12**, 489–508 (2011)
17. D.B. Leineweber, I. Bauer, H.G. Bock, J.P. Schlöder, An efficient multiple shooting based reduced SQP strategy for large scale dynamic process optimization. Part 1: Theoretical aspects. Comput. Chem. Eng. **27**, 157–166 (2003)
18. J. Nocedal, S.J. Wright, *Numerical Optimization*, 2nd edn. (Springer, New York, 2006)
19. K. Styczeń, P. Drąg. A modified multipoint shooting feasible-SQP method for optimal control of DAE systems. Proceedings of the Federated Conference on Computer Science and Information Systems, Szczecin, Poland, pp. 477–484, 18–21 September 2011
20. V.S. Vassiliadis, R.W.H. Sargent, C.C. Pantelides, Solution of a class of multistage dynamic optimization problems. 1. Problems without path constraints. Ind. Eng. Chem. Res. **33**, 2111–2122 (1994)
21. V.S. Vassiliadis, R.W.H. Sargent, C.C. Pantelides, Solution of a Class of Multistage Dynamic Optimization Problems. 2. Problems with Path Constraints. Ind. Eng. Chem. Res. **33**, 2123–2122 (1994)

A Review on the Direct and Indirect Methods for Solving Optimal Control Problems with Differential-Algebraic Constraints

Paweł Drąg, Krystyn Styczeń, Marlena Kwiatkowska and Andrzej Szczurek

Abstract In the article the main features of direct and indirect approaches for solving optimal control problems were presented. The mentioned methods can be effectively applied in the Model Predictive Control of the complex technological systems in electrical, chemical and aerospace engineering, often described by nonlinear differential-algebraic equations. Among the direct and indirect methods for solving optimal control problems one can mention Euler-Lagrange equations, direct optimization methods and indirect gradients methods.

Keywords Optimal control · Direct and indirect methods · Differential-algebraic systems · Multiple shooting method

1 Introduction

The efficient designing and trouble-free controlling of technological systems requires the solution of optimal control problem in a reasonable short computation time. The real-life technological processes often impose themselves necessary constraints on computation time. From this manner, the question about the effective computational methods is crucial. This is important especially, when the dynamics of the system is

P. Drąg (✉) · K. Styczeń
Department of Control Systems and Mechatronics, Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland
e-mail: pawel.drag@pwr.edu.pl

K. Styczeń
e-mail: krystyn.styczen@pwr.edu.pl

M. Kwiatkowska · A. Szczurek
Laboratory of Sensor Technique and Indoor Air Quality Studies, Faculty of Environmental Engineering, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 55-007 Wrocław, Poland
e-mail: marlena.kwiatkowska@pwr.edu.pl

A. Szczurek
e-mail: andrzej.szczurek@pwr.edu.pl

S. Fidanova (ed.), *Recent Advances in Computational Optimization*, Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_6

highly nonlinear. The nonlinearities always incorporate the possibility of the multiple local minima and high sensitivities to small changes in parameter values. Depending on how the problem was formulated, it may have different number of possible solutions and require very different computational efforts.

The development of both the software and the hardware has enabled us to use the optimal control methods in complex industrial processes [18]. Nowadays, the main trends for the optimal control problems are:

(a) approach based on the Euler-Lagrange equations, which enables us to obtain a solution in the infinite dimensional space [13],

(b) advanced finite-dimensional optimization methods, which widely use nonlinear programming algorithms; these algorithms include different variants of sequential quadratic programming (SQP),

(c) minimization of the Hamiltonian function combined with the gradient methods [22].

The paper focuses on the review of direct and indirect methods for solving optimal control problems with differential-algebraic constraints. In the Sect. 2 indirect approach for solving the optimal control problem, which results in differential-algebraic system, was presented. The main features of direct and indirect methods was given in Sect. 3. The other various direct methods, extended with multiple shooting approach, were discussed in Sects. 4 and 5. Indirect gradient methods, which concerns minimization of the Hamiltonian function, were presented in the Sect. 6. The application areas of presented algorithms were given and the considerations were concluded in Sect. 8.

In the article was used the same notation as in [8] and [22].

2 Euler-Lagrange Equations

Let us consider the optimal control problem of the dynamical system

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

where $x(t) \in \mathcal{R}^{n_x}$ is a differential state, $u(t) \in \mathcal{R}^{n_u}$ is a control function, $t \in \mathcal{R}$ is an independent variable and $f : \mathcal{R}^{n_x} \times \mathcal{R}^{n_u} \times \mathcal{R} \to \mathcal{R}^{n_x}$ is a vector-valued function. Let consider also the constraints on the control function

$$u(t) \in \mathcal{U}. \tag{2}$$

The objective is to minimize the cost function of the following form

$$\min_{u(t)} \int_{t_0}^{t_f} L\big(x(t), u(t), t\big)dt + E\big(x(t_f)\big). \tag{3}$$

For a such defined task, let us define the Hamiltonian function $\mathcal{H}$ as

$$\mathcal{H}(x, u, \lambda, \mu) = L(x, u) + \lambda^T f(x, u) + \mu^T c(u), \tag{4}$$

where $\lambda = \lambda(t)$ is a co-state variable, $\mu = \mu(t)$ is a multiplier for inequality constraints on the control function and $c\big(u(t)\big) \leq 0$ is equivalent to the constraints $u(t) \in \mathcal{U}$.

The first order conditions for the minimization of the functional (3) subject to the constraints have the following form

$$\dot{x} = \nabla_\lambda \mathcal{H}, \tag{5}$$

$$\dot{\lambda} = -\nabla_x \mathcal{H}, \tag{6}$$

$$\lambda(t_f) = -\nabla_x E\big(x(t_f)\big), \tag{7}$$

$$0 = \nabla_u \mathcal{H}. \tag{8}$$

If the initial values of the co-state variables are known, then the optimal trajectories of the state and co-state, together with an optimal trajectory of the control variable can be obtained by simulating the system in a time interval $t \in [t_0, t_f]$. While the initial values of the co-state variables are implicitly determined by the final conditions, a value of $\lambda(0)$ can not be generally known analytically.

The necessary optimality conditions lead to the differential-algebraic equations (DAEs) [5]. In such systems the differential variables (x, λ) and algebraic variable u can be distinguished. Because the DAE system is taken into account, one can expect that the algebraic equations determine the algebraic variables. At this point, the optimality condition $0 = \nabla_u \mathcal{H}^T$ defines a control variable, provided that the matrix $\nabla^2_{uu}\mathcal{H}$ is nonsingular [2].

However, if the matrix $\nabla^2_{uu}\mathcal{H}$ is singular, then the control function u is not uniquely determined by the optimality condition. In such situation, singular arcs are considered.

3 Direct and Indirect Methods

The well-known classification of ways for solving optimal control problems is divided into direct and indirect methods. Direct methods construct a sequence of points

$$z_1, z_2, \ldots, z^\star, \tag{9}$$

that causes the objective function values $F(z)$ are getting smaller

$$F(z_1) > F(z_2) > \cdots > F(z^\star). \tag{10}$$

The purpose of indirect methods is to find the solutions of the equation, resulting from the application of the necessary optimality conditions $F'(z) = 0$.

Direct methods require only a comparison of the value of the objective function. In contrast, the indirect method first examine the slope $F'(z)$ and then decide whether it is sufficiently close to zero. This means, that the indirect methods seek solutions of the necessary conditions, while the direct method are searching for the minimum of the objective function or the Lagrange function.

In his book [2] J.T. Betts gave three main difficulties associated with the use of the indirect methods.

1. The user should calculate $\nabla_x \mathcal{H}$ and $\nabla_u \mathcal{H}$, which are necessary to perform further calculations. However, the calculation of these terms requires from the user a basic knowledge of optimal control theory. Even if the user will be able to provide analytical expressions for these terms, the use of this methodology can be very difficult for complex objects.

2. Every system can be described by using other equations, which means that necessary optimality conditions introduce other equations for each system.

3. If the considered system contains inequality constraints, the sequence of the active constraints should be known a priori. When the number of active constraints is unknown, the switching points are unknown also.

4. Indirect methods are not robust for noise.

5. The user should determine the initial values of the adjoint variables λ, which do not have a physical interpretation.

4 Direct Approach with Multiple Shooting Method

As a result of parametrization both the control function and state variables, direct methods can transform the infinite dimensional optimization problem into the finite-dimensional problem of nonlinear programming (NLP).

Let us consider the following optimal control problem of DAE system

$$\min_{u(t)} \int_{t_0}^{t_f} L\big(x(t), z(t), u(t), p\big) + E\big(x(t_f)\big) \tag{11}$$

subject to the differential-algebraic constraints

$$B(\cdot)\dot{x}(t) = f(x(t), z(t), u(t), p), \tag{12}$$

$$0 = g(x(t), z(t), u(t), p), \tag{13}$$

where $x(t) \in \mathcal{R}^{n_x}$ and $z(t) \in \mathcal{R}^{n_z}$ are the differential and algebraic variables, respectively. The variable $u(t) \in \mathcal{R}^{n_u}$ denotes the control function. The other constant in time parameters were determined by $p \in \mathcal{R}^{n_p}$. It was assumed, that the matrix

$B\big(x(t), z(t), u(t), p\big)$ is invertible. The initial values of differential variables and the parameters are as follows

$$x(t_0) = x_0, \tag{14}$$

$$p = p_0. \tag{15}$$

In the optimal control problem the conditions at the final time are taken into account

$$r_1\big(x(t_f), p\big) = 0, \tag{16}$$

$$r_2\big(x(t_f), p\big) \geq 0. \tag{17}$$

The control algorithms for real-life industrial systems should take into account the limitations of various types. The first noteworthy constraints on the control variables $u(t)$ are

$$h_1(u(t), t) \geq 0, \quad t_0 \leq t \leq t_f. \tag{18}$$

The second type are the constraints on the state variables $x(t)$

$$h_2(x(t), t) \geq 0, \quad t_0 \leq t \leq t_f. \tag{19}$$

In the technological process, there may be inequality constraints binding the state variables and the control variable

$$h\big(x(t), z(t), u(t), p\big) \geq 0. \tag{20}$$

Effective solution of the optimal control problem can be achieved by direct approach extended by multiple shooting method.

4.1 Parametrization of the Optimal Control Problem

Parametrization of an infinite dimensional optimization task is carried out in two steps.

In the first step, the time horizon $[t_0, t_f]$ is divided into a certain number of N subintervals of the form $[t_i, t_{i+1}]$, where

$$t_0 < t_1 < \cdots < t_N = t_f. \tag{21}$$

In the prepared subintervals, the control function $u(t)$ is discretized. For simplicity, the control function is often parametrized as a piecewise constant

$$u(t) = u_i, \qquad t \in [t_i, t_{i+1}]. \tag{22}$$

In the literature the different options of control function parametrization were proposed: a piecewise continuous linear function, piecewise linear and quadratic function with discontinuities [27].

In the next step, the system of differential-algebraic equations is parametrized by multiple shooting method. In this way, complex DAE system can be solved by introducing N subintervals $[t_i, t_{i+1}]$ with initial values for the differential variables s_i^x and the initial values of algebraic variables s_i^z.

In each subinterval $[t_i, t_{i+1}]$ the trajectories $x_i(t)$ and $z_i(t)$ are calculated as the solution of the initial value problem

$$B(\cdot)\dot{x}(t) = f\big(x_i(t), z_i(t), u_i(t), p\big), \tag{23}$$

$$0 = g\big(x_i(t), z_i(t), u_i(t), p\big) - \alpha_i(t) g(s_i^x, s_i^z, u_i, p), \tag{24}$$

$$x_i(t_i) = s_i^x. \tag{25}$$

The extension in the algebraic part (Eq. 24) has been introduced. Thus, consistency condition, significant from the point of solving the system by the DAE solver, was always satisfied. The parameter α_i in the literature is known as the scalar damping factor. There are also other possibilities for DAE relaxation.

In each interval $[t_i, t_{i+1}]$ the trajectories $x_i(t)$ and $z_i(t)$ are the functions of the initial values s_i^x, s_i^z, the control function u_i and parameters of the system p.

The integration of the cost function can be performed independently on each subinterval

$$L_i(s_i^x, s_i^z, u_i, p) = \int_{t_i}^{t_{i+1}} L\big(x_i(t), z_i(t), u_i, p\big)dt. \tag{26}$$

4.2 The Structure of Nonlinear Programming Problem

Parametrization of the optimal control problem, the use of the multiple shooting method and representation of the control function as a piecewise constants function leads to a nonlinear optimization problem

$$\min_{u,s,p} \sum_{i=0}^{N-1} L_i(s_i^x, s_i^z, u_i, p) + E(s_N^x, p). \tag{27}$$

The constraints on the initial values and parameters of the systems are presented as follows

$$s_0^x = x_0, \tag{28}$$

$$p = p_0, \tag{29}$$

the continuity conditions

$$s_{i+1}^{x} = x_i(t_{i+1}), \qquad i = 0, 1, \ldots, N-1 \tag{30}$$

and the consistency conditions

$$0 = g(s_i^x, s_i^z, u_i, p, t), \qquad i = 0, 1, \ldots, N. \tag{31}$$

The constraints on the control variable, along with the other constraints are taken into account only at the mesh points

$$h(s_i^x, s_i^z, u_i, p) \geq 0, \qquad i = 0, 1, \ldots, N, \tag{32}$$

similarly, as the constraints on the final state

$$r_1(s_N^x, p) = 0, \tag{33}$$

$$r_2(s_N^x, p) \geq 0. \tag{34}$$

The separation of both the variables and the appropriate functions can be effectively used in the implementation of the optimization algorithms.

5 Sequential Quadratic Programming with Multiple Shooting Method

One of the commonly used algorithms for solving nonlinear optimization problems is the Sequential Quadratic Programming (SQP) [21]. This algorithm can effectively utilize the structure of the problem, which is a result of the application of the multiple shooting method.

One of the most important advantage of SQP algorithms is their fast local convergence—quadratic or superlinear.

In general, the task of nonlinear programming can be stated as follows

$$\min_{w} F(w), \tag{35}$$

subject to

$$G(w) = 0, \tag{36}$$

$$H(w) \geq 0. \tag{37}$$

The vector w contains all the decision variables referring to the differential state variables, algebraic state variables, controls and parameters of the model

$$w = (s_0^x, s_0^z, u_0, s_0^x, s_0^z, u_1, \ldots, s_N^x, s_N^z, u_N, p). \tag{38}$$

The dynamic equations of the system (23)–(25) take the form of the discrete equations and are attached to the equality constraints $G(w) = 0$.

Starting at the point w^0, the SQP algorithm solves the optimization problem (35)–(37) in subsequent iterations

$$w^{k+1} = w^k + \beta^k \Delta w^k, \qquad k = 0, 1, \ldots. \tag{39}$$

The parameter $\beta^k \in [0, 1]$ is called as the relaxation factor. The search direction Δw^k is the solution of the quadratic programming subproblem

$$\min_{\Delta w \in \Omega^k} \nabla F(w^k)^T \Delta w + \frac{1}{2} \Delta w^T B^k \Delta w \tag{40}$$

subject to

$$G(w^k) + \nabla G(w^k)^T \Delta w = 0, \tag{41}$$

$$H(w^k) + \nabla H(w^k)^T \Delta w \geq 0. \tag{42}$$

The matrix B^k denotes the Hessian approximation of the Lagrangian function

$$B(w, \lambda, \mu) = \nabla^2_{ww} \mathcal{L} = \nabla^2_{ww} F(w) - \sum \tilde{\lambda}^T \nabla^2_{ww} G(w) - \sum \tilde{\mu}^T \nabla^2_{ww} H(w), \tag{43}$$

where $\tilde{\lambda}$ and $\tilde{\mu}$ are the Lagrange multipliers.

Parametrization of both the state variables and the control function means, that the resulting nonlinear optimization problem has a specific structure. The Lagrange function is partially separable and it can be written in the following form

$$\mathcal{L}(w, \tilde{\lambda}, \tilde{\mu}) = \sum_{i=0}^{N-1} \mathcal{L}_i(w_i, \tilde{\lambda}, \tilde{\mu}), \tag{44}$$

where $w_i = (s_i^x, s_i^z, u_i, p)$ are the elements of the vector w, which have an impact on the process in the *i-th* time interval $[t_i, t_{i+1}]$. It should be noted, that the parameter p can be considered as piecewise continuous constant control function. However, this is possible only, if the function $u(t)$ was parametrized in the same way.

The consequence of this approach is a block-diagonal structure of the Hessian matrix. In a similar manner, by the use of the multiple shooting method, the matrices $\nabla G(w)^T$ and $\nabla H(w)^T$ has a sparse block structure. To effectively use the block-diagonal structure of the matrices, many different methods has been proposed.

One of the most important and difficult step in solving nonlinear programming problem is to obtain the matrix B, which influence the convergence rate of the solution

process. In practical application, there are the following approaches for calculating the Hessian matrix or its approximation.

1. *Numerical calculation of the exact Hessian matrix;*
this approach is recommended when the size of the problem is not large and complete calculation of the Hessian does not require large amounts of computations. The use the exact Hessian in the calculation give a very fast convergence to a local solution (in the best case quadratic).
2. *Update the Hessian matrix using higher-order scheme (eg. Broyden or BFGS method);*
3. *Approximation of the Hessian matrix, when the objective function is a function of the least-squares-type* $F(w) = \frac{1}{2}\|C(w)\|_2^2$;
if the residues $C(w)$ of the objective function is sufficiently small, a good Hessian approximation is given by $\Delta_w C \Delta_w C^T$. This approximation is available using only the gradient [21].

Solution of the DAE system and the calculation of the corresponding derivatives can be generated in parallel on different subintervals by the effective DAE solvers. This possibility is given by the use of the multiple shooting method.

The real-life industrial processes, especially in chemical engineering, are the sources of large-scale nonlinear optimization problems. Limited computing resources associated with the power of currently available computers, programming techniques, and the time that one can spent on the calculation, causing the development of new methods that are applied to known algorithms. The new methods are as follows:

- the starting point, which is close to a solution, prevents convergence to another point than the requested optimum,
- when the solution was obtained with expected accuracy, then the calculation can be premature terminated,
- if the starting point is feasible, then the number of required iterations can be considerably reduced,
- *warm starting*—the algorithm starts at a point, which is already regarded as an approximate solution of the problem, for example: known solution of the similar problem.

The premature termination of the calculation is often used together with the "warm starting". Moreover, calculation of the starting point near the solution can significantly improve the optimization process. Then, the optimal solution can be achieved already after a few iterations [7].

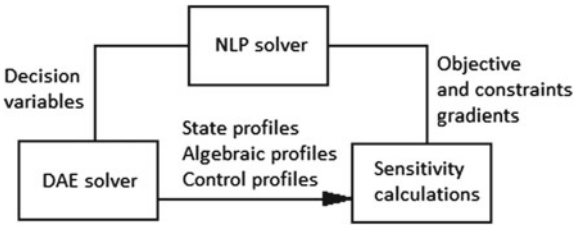


Fig. 1 The scheme of the sequential approach

5.1 The Sequential Approach

The scheme of the sequential approach in dynamic optimization is shown in Fig. 1, where one full optimization cycle was presented. In each cycle, the decision variables w are determined by nonlinear programming algorithm, for example Sequential Quadratic Programming [15, 26].

When the initial values of decision variables are known, then the DAE system can be solved in each time interval $[t_i, t_{i+1}]$. As a result of this step, the profiles of differential and algebraic state trajectories are obtained.

The last component calculates the gradients of the objective and the constraints functions, for the decision variables w. The values of both the objective function and gradients are passed to the nonlinear programming solver. This enables us to update the decision variables w. The described cycle is repeated as long, as the NLP solver reaches the result with the expected accuracy.

The majority of the calculations is related with the solving the DAE systems to obtain gradient vectors of objective and constraints functions. For small and medium-sized problems, with few hundred variables, the appropriate optimization algorithm is *fmincon* solver, attached to the MATLAB environment [12].

5.2 The Simultaneous Approach

In the simultaneous approach, the NLP solver takes into account more variable, than in the sequential case. This is due to the fact, that the dynamic equations and their associated discrete state variables are included as equality constraints in the quadratic programming subproblem. They do not form the Hessian matrix, which remains within a block diagonal structure, but allows for efficient use of sparse factorization methods. The computational complexity of quadratic programming subproblem depends linearly on the number of shots N.

The problem of unstable dynamics is overcome by imposing restrictions on the decision variables by lower and upper bounds.

The main disadvantage of the simultaneous approach is, that the calculation can not be prematurely terminated. The equations describing the model are fulfilled only

after the last iteration. This means, that as long as the algorithm does not achieve a solution, acceptable solutions are not available.

Simultaneous approach transform the optimal control problem into large-scale optimization problem with thousands decision variables and constraints. In practical application, for solving problems rewritten in a such way, interior point nonlinear programming codes are commonly used. The most popular codes are IPOPT [30] and KNITRO [6].

6 Indirect Gradient Approach

The indirect methods for solving optimal control problems base on indirect multiple shooting and indirect collocation approaches. Indirect multiple shooting method is originally known for solving purely continuous optimal control problems. The initialization for indirect multiple shooting method and indirect.

The third class of methods for solving optimal control problems are the combined indirect gradient methods. Passenberg et al. [22] proposed the min-$\mathcal{H}$ algorithm, which can be applied for solving both continuous and hybrid dynamic optimization problems.

Gradient methods are more intuitive to initialize than indirect multiple shooting and indirect collocation approach, because control values can be guessed initially instead of adjoint values.

Starting from the initial conditions, in indirect gradient algorithms, the dynamical system is integrated forward in time until the final time is reached. Then, the adjoint differential equations are integrated backward in time $[t_0, t_f]$. The backward integration is initialized with the relevant optimality conditions at final time. To perform the integrations and the initialization of the adjoint variables, a control function of the time has to be initially guessed. These unknowns are the decision variables, which are iteratively varied until an optimal solution is found. The unknown control function is a physically intuitive variable, which simplified the initialization of the control function. Then, the values of the decision variables are updated after each pair of forward-backward integrations.

It was assumed that the Hamiltonian is convex with respect to the control

$$\nabla^2_{uu}\mathcal{H}(x(t), \lambda(t), u(t)) > 0. \tag{45}$$

This implies, that the Hamiltonian minimization condition is satisfied for any values x and λ if the optimal control is chosen

$$\nabla_u\mathcal{H}(x(t), \lambda(t), u^\star(t)) = 0. \tag{46}$$

In the current iteration of the method, the optimal control $u^\star(t)$ is approached by the Newton step, if the gradient is not zero for the control $u(t)$. Linearization of the current gradient leads to the update formula of the control $u(t)$

$$\delta u(t) = -(\nabla^2_{uu}\mathcal{H}(x(t), \lambda(t), u(t)))^{-1} \tag{47}$$
$$(\nabla^2_{ux}\mathcal{H}(x(t), \lambda(t), u(t))\delta x(t) +$$
$$\nabla_u f^T(x(t), u(t))\delta\lambda(t) + \epsilon\nabla_u\mathcal{H}(x(t), \lambda(t), u(t))).$$

The step size ϵ with $0 < \epsilon \leq 1$ determines, how fast the optimal control $u^\star(t)$ satisfying $\nabla_u\mathcal{H} = 0$ is approached.

The update of the control

$$\delta u(t) = -\epsilon\Delta u(t) + \delta u^\star(t) \tag{48}$$

such that $\Delta u(t) = u(t) - u^\star(t)$ and the predicted variation is

$$\delta u^\star(t) = -(\nabla^2_{uu}\mathcal{H}(x(t), \lambda(t), u^\star(t)))^{-1} \tag{49}$$
$$(\nabla^2_{ux}\mathcal{H}(x(t), \lambda(t), u^\star(t))\delta x(t) +$$
$$\nabla_u f^T(x(t), u^\star(t))\delta\lambda(t))$$

To compute the update $\delta u(t)$ it is required to find the unknown values $\delta x(t)$ and $\delta\lambda(t)$ needed for $\delta u^\star(t)$. The values $\delta x(t)$ and $\delta\lambda(t)$ where obtained by solving the linear TPBVP of the forms

$$\delta\dot{x}(t) = A(t)\delta x(t) - B(t)\delta\lambda(t) - d(t) \tag{50}$$

$$\delta\dot{\lambda}(t) = -C(t)\delta x(t) - F(t)\delta\lambda(t) - \gamma(t), \tag{51}$$

where the matrices A, B, C, F and vectors d, γ result from the linearization of the differential equations.

The solution of the TPBVP can be derived with a single pair of backward differential integrations.

7 Applications

In the literature related to the controlling the large-scale, chemical industrial processes, more important, than the choice between using indirect or direct approach, is rather a choice between sequential and simultaneous approaches.

Direct methods combined with the multiple shooting approach are widely used in the control of batch processes [3].

Batch processes involve defined sequence of operations processes and can be single or multistage. These operations are the function of the process and the product. Each stage of the process (heating, cooling, pressure changing, adding and removing substances) makes several physical or chemical changes in the being processed

material. These stages run for a specified period of time, until the manufacture of the final product [20, 24, 25].

Batch processes are often used in places, where product characteristics requirements are very high, i.e. pharmaceuticals production [29]. Other examples include: fermentation [23], crystallization [31] and polymer preparation processes [1], utilized in the wide range of industry branches: petrochemical, paper and food ones [17, 19, 25]. The batch processes link to the high pressure tubular and kinetics reactors [3, 10, 27, 28], distillation columns [3] and other reactions, that take place in the presence of the catalyst [9, 16].

Batch process control stands important and difficult aspect of product manufacturing. Factors that can affect the batch processes can be different. In dynamic optimization problems of batch processes off-line and online problems can be distinguished [3].

Batch process control is necessary from an economical point of view. The costs arising from inadequate production of product batches are usually very large. Furthermore, even small changes in technological parameters may affect the final products quantity and quality [25].

The second wide range of applications is an aerospace industry. An important issue is the control of the airplane, to achieve the proper height or to determine the trajectory of the shuttle, which returns from a space mission [2].

Application of the described methods is highly connected with the application of the appropriate software packages. To the most popular are the Sequential Quadratic Programming codes, e.g. fmincon [12], filterSQP [11], MUSCOD-II [18], SNOPT [14] and SOCS [2].

8 Summary

The main trends in solving optimal control problems were presented in the article. The starting point were the Euler-Lagrange equations. On this basis, one can designate the necessary conditions for optimality, and as a result obtain a system of differential-algebraic equations. Because of the obtaining this type of equations requires knowledge of advanced mathematical methods, the new, more intuitive, direct methods have been developed. The direct methods transform the optimal control problem into a large-scale nonlinear programming problem, which can be solved sequentially or simultaneously. The multiple shooting method allows to control nonlinear and ill-conditioned dynamic systems, described by ordinary differential equations (ODE) and differential-algebraic equations (DAE). Large-scale optimization problem can be solved effectively by nonlinear programming algorithms [21].

Algorithms based on the direct approach are widely used in industry, especially in the chemical industry and aerospace engineering.

The new trends in solving optimal control problems are the min-$\mathcal{H}$ algorithms, which jet have not been widely used for control and optimization of the complex industrial processes.

Acknowledgments This project was supported by the grant Młoda Kadra B30036/I6 at Wrocław University of Technology.

References

1. P. Arora, R. Jain, K. Mathur, A. Sharma, A. Gupta, Synthesis of polymethyl methacrylate (PMMA) by batch emulsion polymerization. Afr. J. Pure Appl. Chem. **4**, 152–157 (2010)
2. J.T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd edn. (SIAM, Philadelphia, 2010)
3. L.T. Biegler, *Nonlinear Programming. Concepts, Algorithms, and Applications to Chemical Processes* (SIAM, Philadelphia, 2010)
4. L.T. Biegler, S.L. Campbell, V. Mehrmann, *Control and Optimization with Differential-Algebraic Constraints* (SIAM, Philadelphia, 2012)
5. K.E. Brennan, S.L. Campbell, L.R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations* (SIAM, Philadelphia, 1996)
6. R.H. Byrd, M.E. Hribar, J. Nocedal, An interior point algorithm for large-scale nonlinear programming. SIAM J. Optim. **9**, 877–900 (1999)
7. M. Cannon, Efficient nonlinear model predictive control algorithms. Annu. Rev. Control **28**, 229–237 (2004)
8. M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, F. Allgöwer, Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equation. J. Process Control **12**, 577–585 (2002)
9. P. Drąg, K. Styczeń, A two-step approach for optimal control of kinetic batch reactor with electroneutrality condition. Prz. Elektrotechiczny **6**, 176–180 (2012)
10. P. Drąg, K. Styczeń, Parallel simultaneous approach for optimal control of DAE systems, Proceedings of the federated conference on computer science and information systems (FedCSIS), 545–551, 2012
11. R. Fletcher, S. Leyffer, Nonlinear programming without a penalty function. Math. Program. **91**, 239–269 (2002)
12. fmincon SQP Algorithm, MATLAB User's Guide, MathWorks, 2012
13. I.M. Gelfand, S.W. Fomin, *Rachunek Wariacyjny* (Państwowe Wydawnictwo Naukowe, Warszawa, 1975)
14. P.E. Gill, W. Murray, A. Saunders, SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**, 99–131 (2005)
15. A. Hartwich, K. Stockmann, C. Terboven, S. Feuerriegel, W. Marquardt, Parallel sensitivity analysis for efficient large-scale dynamic optimization. Optim. Eng. **12**, 489–508 (2011)
16. Y.J. Huang, G.V. Reklaitis, V. Venkatasubramanian, Model decomposition based method for solving general dynamic optimization problems. Comput. Chem. Eng. **26**, 863–873 (2002)
17. V. Kumar, P. Dhall, S. Naithani, A. Kumar, R. Kumar, Biological approach for the treatment of pulp and paper industry effluent in sequence batch reactor. J. Bioremediation Biodegrad. **5**, 218 (2014)
18. D.B. Leineweber, I. Bauer, H.G. Bock, J.P. Schlöder, An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: Theoretical aspects. Comput. Chem. Eng. **27**, 157–166 (2003)
19. A. Malakahmada, A. Hasani, M. Eisakhani, M.H. Isa, Sequencing batch reactor (SBR) for the removal of Hg^{2+} and Cd^{2+} from synthetic petrochemical factory wastewater. J. Hazard. Mater. **191**, 118–125 (2011)
20. C.T. Maravelias, A decomposition framework for the scheduling of single- and multi-stage processes. Comput. Chem. Eng. **30**, 407–420 (2006)
21. J. Nocedal, S.J. Wright, *Numerical Optimization*, 2nd edn. (Springer, New York, 2006)

22. B. Passenberg, M. Leibold, O. Stursberg, M. Buss, A globally convergent, locally optimal min-H algorithm for hybrid optimal control. SIAM J. Control Optim. **52**, 718746 (2014)
23. B.C. Saha, Commodity Chemicals Production by Fermentation: An Overview, in *Fermentation Biotechnology, ACS Symposium Series*, ed. by B.C. Saha (American Chemical Society, Washington, 2003), pp. 3–17
24. P.E. Sawyer, Control, *Handbook of Batch Process Design* (Springer, Berlin, 1997), pp. 219–252
25. P.G. Smith, *Introduction to Food Process Engineering* (Springer, Berlin, 2011)
26. K. Styczeń, P. Drąg, A modified multipoint shooting feasible-SQP method for optimal control of DAE systems. In Proceedings of Federated Conference on Computer Science and Information Systems (FedCSIS), 477–484, 2011
27. V.S. Vassiliadis, R.W.H. Sargent, C.C. Pantelides, Solution of class of multistage dynamic optimization problems. 1. problems without path constraints. Ind. Eng. Chem. Res. **33**, 2111–2122 (1994)
28. V.S. Vassiliadis, R.W.H. Sargent, C.C. Pantelides, Solution of class of multistage dynamic optimization problems. 2. problems with path constraints. Ind. Eng. Chem. Res. **33**, 2123–2133 (1994)
29. Y. Wang, Y. Yang, D. Zhou, F. Gao, Active fault-tolerant control of nonlinear batch processes with sensor faults. Ind. Eng. Chem. Res. **46**, 9158–9169 (2007)
30. A. Wätcher, L.T. Biegler, On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**, 25–57 (2006)
31. R.C. Zumstein, T. Gambrel, R.W. Rousseau, factors affecting the purity of l-isoleucine recov ered by batch crystallization, in: A.S. Myerson, K. Toyokura, Crystallization as a Separations Process, ACS Symposium Series **438**, 85–99, 2009

InterCriteria Analysis of ACO and GA Hybrid Algorithms

Olympia Roeva, Stefka Fidanova and Marcin Paprzycki

Abstract In this paper, the recently proposed approach for multicriteria decision making—InterCriteria Analysis (ICA)—is presented. The approach is based on the apparatus of the index matrices and the intuitionistic fuzzy sets. The idea of InterCriteria Analysis is applied to establish the relations and dependencies of considered parameters based on different criteria referred to various metaheuristic algorithms. A hybrid scheme using Genetic Algorithm (GA) and Ant Colony Optimization (ACO) is used for parameter identification of *E. coli* MC4110 fed-batch cultivation process model. In the hybrid GA-ACO, the GA is used to find feasible solutions to the considered optimization problem. Further ACO exploits the information gathered by GA. This process obtains a solution, which is at least as good as—but usually better than—the best solution devised by GA. Moreover, a comparison with both the conventional GA and ACO identification results is presented. Based on ICA the obtained results are examined and conclusions about existing relations and dependencies between model parameters of the *E. coli* process and algorithms parameters and outcomes, such as number of individuals, number of generations, value of the objective function and computational time, are discussed.

Keywords InterCriteria analysis · Metaheuristics · Hybrid algorithm · Ant colony optimization · Genetic algorithm · *E. coli* cultivation process

O. Roeva
Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Science, Sofia, Bulgaria
e-mail: olympia@biomed.bas.bg

S. Fidanova (✉)
Institute of Information and Communication Technology, Bulgarian Academy of Science, Sofia, Bulgaria
e-mail: stefka@parallel.bas.bg

M. Paprzycki
Systems Research Institute, Polish Academy of Sciences, Warsaw and Management Academy, Warsaw, Poland
e-mail: marcin.paprzycki@ibspan.waw.pl

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_7

1 Introduction

To solve different optimization problems we can apply various techniques and approaches, namely exact algorithms (Branch-and-Bound, Dynamic Programming, local search techniques) [14, 20, 39], heuristics [27, 35], and metaheuristics (Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization, Simulated Annealing, Tabu Search, etc.) [15, 17, 23]. Today, the use of metaheuristics has received more and more attention. These methods offer good solutions, even global optima, within reasonable computing time [38]. An even more efficient behavior and higher flexibility when dealing with real-world and large-scale problems, can be achieved through a combination of a metaheuristic with other optimization techniques, the so-called hybrid metaheuristic [15, 22, 29, 30, 36, 37, 40].

The main goal of the hybrid algorithms is to exploit the advantages of different optimization strategies, avoiding their disadvantages. Choosing an adequate combination of metaheuristic techniques we can achieve a better algorithm performance in solving hard optimization problems. Developing such effective hybrid algorithm requires expertise from different areas of optimization. There are many hybridization techniques that have shown to be successful for different applications.

In this paper, we investigate a hybrid metaheuristic method that combines Genetic Algorithms (GA) and Ant Colony Optimization (ACO), named GA-ACO. There are some applications of ACO-GA hybrid for several optimization problems. In [25, 26] a hybrid metaheuristic ACO-GA for the problem of sports competition scheduling is presented. In the proposed algorithm first, GA generates activity lists thus provides the initial population for ACO. Next, ACO is executed. In the next step GA, based on the crossover and mutation operations, generates new population. continuous engineering optimization. Authors in [18] presented hybrid algorithm in that ACO and GA search alternately and cooperatively in the solution space. Test examples show that hybrid algorithm can be more efficient and robust than the traditional population based heuristic methods. In [2] the problem of medical data classification is discussed. Authors propose a hybrid GA-ACO and show the usefulness of the proposed approach on a number of benchmark real-world medical datasets. For solving NP-hard combinatorial optimization problems in [1] a novel hybrid algorithm combining the search capabilities of the ACO and GA is introduced. As a result a faster and better search algorithm capabilities is achieved.

Provoked by the promising results obtained from the use of hybrid GA-ACO algorithms, we propose a hybrid algorithm, i.e. collaborative combination between ACO and GA for model parameters optimization of *E. coli* cultivation process. The effectiveness of GA and ACO have already been demonstrated for model parameter optimization considering fed-batch cultivation processes [32]. Moreover, parameter identification of cellular dynamics models has especially become a research field of great interest. Robust and efficient methods for parameter identification are of key importance.

On the other hand, the recently proposed approach for multicriteria decision making—InterCriteria Analysis (ICA)—is applied for additional exploring of the

used metaheuristic techniques. In here discussed case the *E. coli* model parameter estimates, number of individuals (chromosomes and ants), number of algorithm generations, corresponding algorithm accuracy and computational time are considered as user criteria. The ICA is applied with the aim to more profoundly understand the nature of the criteria involved and discover on this basis existing correlations between the criteria themselves. The theory of ICA has been presented in details in [4], and in [9–12] it was further discussed and developed.

The paper is organized as follows. The problem formulation is given in Sect. 2. The proposed hybrid GA-ACO technique is described in Sect. 3. The background of the ICA is presented in Sect. 4. The numerical results and a discussion are presented in Sect. 5. Conclusion remarks are done in Sect. 6.

2 Problem Formulation

2.1 E. coli Fed-batch Fermentation Model

The mathematical model of the fed-batch cultivation process of *E. coli* is presented by the following non-linear differential equation system [33]:

$$\frac{dX}{dt} = \mu X - \frac{F_{in}}{V} X \tag{1}$$

$$\frac{dS}{dt} = -q_S X + \frac{F_{in}}{V}(S_{in} - S) \tag{2}$$

$$\frac{dV}{dt} = F_{in} \tag{3}$$

where

$$\mu = \mu_{max} \frac{S}{k_S + S} \tag{4}$$

$$q_S = \frac{1}{Y_{S/X}} \mu \tag{5}$$

X is the biomass concentration, [g/l];
S is the substrate concentration, [g/l];
F_{in} is the feeding rate, [l/h];
V is the bioreactor volume, [l];
S_{in} is the substrate concentration in the feeding solution, [g/l];
μ and q_S are the specific rate functions, [1/h];
μ_{max} is the maximum value of the specific growth rate, [1/h];
k_S is the saturation constant, [g/l];
$Y_{S/X}$ is the yield coefficient, [−].

Fig. 1 Bioreactor and FIA measurement system

For the model parameters identification, experimental data of an *E. coli* MC4110 fed-batch cultivation process are used. The experiments are performed in the Institute of Technical Chemistry, University of Hannover, Germany. The detailed description of the cultivation condition and experimental data could be found in [3, 31].

The fed-batch process starts at time $t = 6.68$ h, after batch phase. The initial liquid volume is 1350 ml. Before inoculation a glucose concentration of 2.5 g/l was established in the medium. Glucose in feeding solution is 100 g/l. The temperature was controlled at 35 °C, the pH at 6.9. The stirrer speed was set to 900 rpm and was increased to 1800 rpm, so that the dissolved oxygen concentration was never below 30 %. The aeration rate was kept at 275 l/h and the carbon dioxide was measured in the exhaust gas. The process is stopped at time $t = 11.54$ h.

The bioreactor, as well as FIA measurement system is shown on Fig. 1. The feed rate profile and the dynamics of the measured substrate concentration are presented, respectively on Figs. 2 and 3.

For the considered non-linear mathematical model of *E. coli* fed-batch cultivation process Eqs. (1)–(5) the parameters that should be identified are:

- maximum specific growth rate (μ_{max}),
- saturation constant (k_S),
- yield coefficient ($Y_{S/X}$).

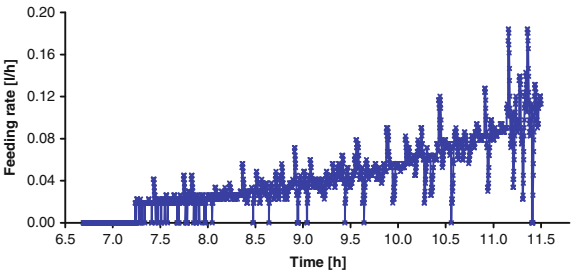


Fig. 2 Feed rate profile

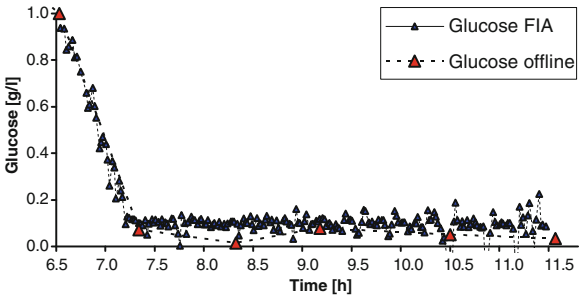


Fig. 3 Measured substrate concentration

The following upper and lower bounds of the model parameters are considered [32]:

$$0 < \mu_{max} < 0.7,$$

$$0 < k_S < 1,$$

$$0 < 1/Y_{S/X} < 30.$$

In the model identification procedures measurements of main process variables (biomass and glucose concentration) are used. For on-line glucose determination a FIA system has been employed. For biomass, off-line analysis are performed [3].

2.2 Optimization Criterion

The objective consists of adjusting the parameters (μ_{max}, k_S and $Y_{S/X}$) of the non-linear mathematical model function Eqs. (1)–(5) to best fit a data set. The objective function is presented as a minimization of a distance measure J between experimental and model predicted values of the main state variables (biomass X and substrate S):

$$J = \sum_{i=1}^{m} \left(X_{\text{exp}}(i) - X_{\text{mod}}(i)\right)^2 + \\ + \sum_{i=1}^{m} \left(S_{\text{exp}}(i) - S_{\text{mod}}(i)\right)^2 \rightarrow \min \tag{6}$$

where m is the number of experimental data; X_{exp} and S_{exp} are the known experimental data for biomass and substrate; X_{mod} and S_{mod} are the model predictions for biomass and substrate with a given set of parameters (μ_{max}, k_S and $Y_{S/X}$).

3 Methodology

3.1 Genetic Algorithm

GA is a metaheuristic technique based on an analogy with the genetic structure and behaviour of chromosomes within a population of individuals using the following foundations [21]:

- chromosomes in a population compete for resources and mates;
- those chromosomes most successful in each competition will produce more offspring than those chromosomes that perform poorly;
- genes from good chromosomes propagate throughout the population so that two good parents will sometimes produce offspring that are better than either parent;
- thus each successive generation will become more suited to their environment.

The structure of the GA, shown by the pseudocode is presented in Fig. 4.

GA mainly operating on binary strings and using a recombination operator with mutation. GA support a population of chromosomes, $Pop(t) = x_1^t, \ldots, x_n^t$ for generation t. Each chromosome introduces a potential solution to the problem and is implemented as some data structure S. Each solution is evaluated according its "fitness". Fitness of a chromosome is assigned proportionally to the value of the objective function of the chromosomes. Then, a new population (generation $t + 1$) is formed by selecting better chromosomes (selection step).

Roulette wheel, developed by Holland [28] is most used selection method. The probability, P_i, for each chromosome to be selected is defined by:

$$P[\text{Individual } i \text{ is chosen}] = \frac{F_i}{\sum_{j=1}^{PopSize} F_j}, \tag{7}$$

where F_i equals the fitness of chromosome i and $PopSize$ is the population size.

Fig. 4 Pseudocode for GA

Genetic Algorithm
$i = 0$
Initial population $Pop(0)$
Evaluate $Pop(0)$
while (not done) **do** (test for termination criterion)
    $i = i + 1$
    Select $Pop(i)$ from $Pop(i - 1)$
    Recombine $Pop(i)$
    Mutate $Pop(i)$
    Evaluate $Pop(i)$
end while
Final solution

Selected members of the new population have been subjected to transformations by means of "genetic" operators to form new solution. There are unary transformations m_i (mutation type), which create new chromosomes by a small change in a single chromosome ($m_i : S \to S$), and higher order transformations c_j (crossover type), which create new chromosomes by combining parts from several chromosomes ($c_j : S \times \cdots \times S \to S$). The combined effect of selection, crossover and mutation gives so-called reproductive scheme growth equation (the schema theorem) [24]:

$$\xi(S, t+1) \geq$$

$$\xi(S, t) \cdot eval(S, t) / \bar{F}(t) \left[1 - p_c \cdot \frac{\delta(S)}{m-1} - o(S) \cdot p_m\right].$$

Good schemata receive an exponentially increasing number of reproductive trials in successive generations.

3.2 Ant Colony Optimization

The ACO is a stochastic optimization method that mimics the social behavior of real ants colonies, which try to find shortest rout to feeding sources and back. Real ants lay down quantities of pheromone (chemical substance) marking the path that they follow. An isolated ant moves essentially at random but an ant encountering a previously laid pheromone will detect it and decide to follow it with high probability and reinforce it with a further quantity of pheromone. The repetition of the above mechanism represents the auto-catalytic behavior of a real ant colony, where the more the ants follow a trail, the more attractive that trail becomes. The idea comes from observing the exploitation of resources of food among ants, in which ants have collectively been able to find the shortest path between to the food.

The ACO is implemented as a team of intelligent agents, which simulate the ants behavior, walking around the graph representing the problem to solve. The requirements of the ACO algorithm are as follows [16, 19]:

- The problem needs to be represented appropriately, which would allow the ants to incrementally update the solutions through the use of a probabilistic transition rules, based on the amount of pheromone in the trail and other problem specific knowledge.
- A problem-dependent heuristic function, that measures the quality of components that can be added to the current partial solution.
- A rule set for pheromone updating, which specifies how to modify the pheromone value.
- A probabilistic transition rule based on the value of the heuristic function and the pheromone value, that is used to iteratively construct a solution.

Fig. 5 Pseudocode for ACO

```
Ant Colony Optimization
Initialize number of ants;
Initialize the ACO parameters;
while not end-condition do
    for k = 0 to number of ants
        ant k choses start node;
        while solution is not constructed do
            ant k selects higher probability node;
        end while
    end for
    Update-pheromone-trails;
end while
```

The structure of the ACO algorithm, shown by the pseudocode is presented in Fig. 5.

The transition probability $p_{i,j}$, to choose the node j when the current node is i, is based on the heuristic information $\eta_{i,j}$ and the pheromone trail level $\tau_{i,j}$ of the move, where $i, j = 1, \ldots, n$.

$$p_{i,j} = \frac{\tau_{i,j}^a \eta_{i,j}^b}{\sum_{k \in Unused} \tau_{i,k}^a \eta_{i,k}^b}, \tag{8}$$

where $Unused$ is the set of unused nodes of the graph.

The higher the value of the pheromone and the heuristic information, the more profitable it is to select this move and resume the search. In the beginning, the initial pheromone level is set to a small positive constant value τ_0; later, the ants update this value after completing the construction stage. The ACO algorithms adopt different criteria to update the pheromone level.

The pheromone trail update rule is given by:

$$\tau_{i,j} \leftarrow \rho \tau_{i,j} + \Delta \tau_{i,j}, \tag{9}$$

where ρ models evaporation in the nature and $\Delta \tau_{i,j}$ is a new added pheromone which is proportional to the quality of the solution. Better solutions will receive more pheromone than others and will be more desirable in a next iteration.

3.3 Hybrid GA-ACO Algorithm

We proposed to combine two metaheuristics, namely GA [24, 28] and ACO [19]. GA is a population-based method where initial population is randomly generated. Thus generated initial solutions, further is genetically evaluated. ACO algorithm is a population-based too. The difference with GA is that ACO do not need initial

Fig. 6 Pseudocode for Hybrid GA-ACO

```
GA-ACO hybrid algorithm
i = 0
Initial population Pop(0)
Evaluate Pop(0)
while not end-condition do
        i = i + 1
        Select Pop(i) from Pop(i − 1)
        Recombine Pop(i)
        Mutate Pop(i)
        Evaluate Pop(i)
end while
Final GA solution for ACO
Initialize number of ants;
Initialize the ACO parameters;
while not end-condition do
        for k = 0 to number of ants
            ant k choses start node;
            while solution is not constructed do
                    ant k selects higher probability node;
            end while
        end for
        Update-pheromone-trails;
end while
Final solution
```

population. ACO is a constructive method and we manage the ants to look for good solutions by parameter called pheromone. At the beginning the initial pheromone is the same for the elements of the all potential solutions. After every iteration the pheromone is updated. The elements of better solutions receive more pheromone then others and become more desirable in a next iterations. In our hybrid algorithm the solutions achieved by GA are like solutions achieved by ACO from some previous iteration and we update the initial pheromone according them. After that we continue with ACO algorithm.

The pseudocode of the proposed GA-ACO algorithm is shown in Fig. 6.

4 InterCriteria Analysis

4.1 Short Remarks on Intuitionistic Fuzzy Pairs and Index Matrices

The Intuitionistic Fuzzy Pairs (IFPs) is an object in the form of an ordered pair

$$\langle a, b \rangle,$$

where $a, b \in [0, 1]$ and $a + b \leq 1$.

IFPs are used as an evaluation of some object or process, and the components (a and b) are interpreted, respectively, as degrees of membership and non-membership to a given set, or degrees of validity and non-validity, or degree of correctness and non-correctness, etc. [5].

Let us have two IFPs $x = \langle a, b\rangle$ and $y = \langle c, d\rangle$.

In [5] the following relations are defined:

$$\begin{array}{l} x < y \text{ iff } a < c \text{ and } b > d \\ x \leq y \text{ iff } a \leq c \text{ and } b \geq d \\ x = y \text{ iff } a = c \text{ and } b = d \\ x \geq y \text{ iff } a \geq c \text{ and } b \leq d \\ x > y \text{ iff } a > c \text{ and } b < d \end{array}$$

The concept of Index Matrix (IM) was introduced in [6] and discusssed in more details in [7, 8].

The basic definitions and properties related to IMs are follows [7]:

Let I be a fixed set of indices and $\mathcal{R}$ be the set of all real numbers. By IM with index sets K and L ($K, L \subset I$), we mean the object,

$$[K, L, \{a_{k_i,l_j}\}] \equiv \begin{array}{c|cccc} & l_1 & l_2 & \ldots & l_n \\ \hline k_1 & a_{k_1,l_1} & a_{k_1,l_2} & \ldots & a_{k_1,l_n} \\ k_2 & a_{k_2,l_1} & a_{k_2,l_2} & \ldots & a_{k_2,l_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_m & a_{k_m,l_1} & a_{k_m,l_2} & \ldots & a_{k_m,l_n} \end{array},$$

where

$$K = \{k_1, k_2, ..., k_m\},\ L = \{l_1, l_2, ..., l_n\},$$

and for $1 \leq i \leq m$, and $1 \leq j \leq n : a_{k_i,l_j} \in \mathcal{R}$.

On the basis of the above definition, in [8] the new object—the Intuitionistic Fuzzy IM (IFIM)—was introduced in the form

$$[K, L, \{\langle \mu_{k_i,l_j}, \nu_{k_i,l_j}\rangle\}]$$

$$\equiv \begin{array}{c|ccccc} & l_1 & & l_2 & \ldots & l_n \\ \hline k_1 & \langle \mu_{k_1,l_1}, \nu_{k_1,l_1}\rangle & \langle \mu_{k_1,l_2}, \nu_{k_1,l_2}\rangle & \ldots & \langle \mu_{k_1,l_n}, \nu_{k_1,l_n}\rangle \\ k_2 & \langle \mu_{k_2,l_1}, \nu_{k_2,l_1}\rangle & \langle \mu_{k_2,l_2}, \nu_{k_2,l_2}\rangle & \ldots & \langle \mu_{k_2,l_n}, \nu_{k_2,l_n}\rangle \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_m & \langle \mu_{k_m,l_1}, \nu_{k_m,l_1}\rangle & \langle \mu_{k_m,l_2}, \nu_{k_m,l_2}\rangle & \ldots & \langle \mu_{k_m,l_n}, \nu_{k_m,l_n}\rangle \end{array},$$

where for every $1 \leq i \leq m, 1 \leq j \leq n$: $0 \leq \mu_{k_i,l_j}, \nu_{k_i,l_j}, \mu_{k_i,l_j} + \nu_{k_i,l_j} \leq 1$, i.e., $\langle \mu_{k_i,l_j}, \nu_{k_i,l_j}\rangle$ is an IFP.

Let us have an IM

$$A = \begin{array}{c|cccccccc} & O_1 & \dots & O_k & \dots & O_l & \dots & O_n \\ \hline C_1 & a_{C_1,O_1} & \dots & a_{C_1,O_k} & \dots & a_{C_1,O_l} & \dots & a_{C_1,O_n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ C_i & a_{C_i,O_1} & \dots & a_{C_i,O_k} & \dots & a_{C_i,O_l} & \dots & a_{C_i,O_n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ C_j & a_{C_j,O_1} & \dots & a_{C_j,O_k} & \dots & a_{C_j,O_l} & \dots & a_{C_j,O_n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ C_m & a_{C_m,O_1} & \dots & a_{C_m,O_k} & \dots & a_{C_m,O_l} & \dots & a_{C_m,O_n} \end{array}, \tag{10}$$

where for every p, q, $(1 \leq p \leq m, 1 \leq q \leq n)$:

- C_p is a criterion, taking part in the evaluation,
- O_q is an object, being evaluated.
- a_{C_p,O_q} is a real number or another object, that is comparable about relation R with the other a-objects, so that for each i, j, k: $R(a_{C_k,O_i}, a_{C_k,O_j})$ is defined. Let $\overline{R}$ be the dual relation of R in the sense that if R is satisfied, then $\overline{R}$ is not satisfied and vice versa. For example, if "R" is the relation "$<$", then $\overline{R}$ is the relation "$>$", and vice versa.

Let $S^{\mu}_{k,l}$ be the number of cases is which $R(a_{C_k,O_i}, a_{C_k,O_j})$ and $R(a_{C_l,O_i}, a_{C_l,O_j})$ are simultaneously satisfied. Let $S^{\nu}_{k,l}$ be the number of cases is which $R(a_{C_k,O_i}, a_{C_k,O_j})$ and $\overline{R}(a_{C_l,O_i}, a_{C_l,O_j})$ are simultaneously satisfied.

Obviously,

$$S^{\mu}_{k,l} + S^{\nu}_{k,l} \leq \frac{n(n-1)}{2}.$$

Now, for every k, l, such that $1 \leq k < l \leq m$ and for $n \geq 2$, we define

$$\mu_{C_k,C_l} = 2\frac{S^{\mu}_{k,l}}{n(n-1)}, \quad \nu_{C_k,C_l} = 2\frac{S^{\nu}_{k,l}}{n(n-1)}. \tag{11}$$

Therefore, $\langle \mu_{C_k,C_l}, \nu_{C_k,C_l} \rangle$ is an IFP. Now, we can construct the IM

$$\begin{array}{c|ccc} & C_1 & \dots & C_m \\ \hline C_1 & \langle \mu_{C_1,C_1}, \nu_{C_1,C_1} \rangle & \dots & \langle \mu_{C_1,C_m}, \nu_{C_1,C_m} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ C_m & \langle \mu_{C_m,C_1}, \nu_{C_m,C_1} \rangle & \dots & \langle \mu_{C_m,C_m}, \nu_{C_m,C_m} \rangle \end{array}, \tag{12}$$

that determine the degrees of correspondence between criteria $C_1, \dots, C_m$.

5 Numerical Results and Discussion

5.1 Model Parameters Identification of E. coli Fed-batch Fermentation Process

The theoretical background of the GA and ACO is presented in details [32]. For the considered here model parameter identification, we used real-value coded GA instead binary encoding. The type of the basic operators in GA are as follows:

- encoding—real-value,
- fitness function—linear ranking,
- selection function—roulette wheel selection,
- crossover function—extended intermediate recombination,
- mutation function—real-value mutation,
- reinsertion—fitness-based.

In the applied here ACO algorithm the problem is represented by graph and the artificial ants try to construct shortest path under some conditions. In our case the graph of the problem is represented by tripartite graph. There are not arcs inside a level and there are arcs between levels. Every level corresponds to one of the model parameters we identify (μ_{max}, k_S and $Y_{S/X}$).

To set to the optimal settings the parameters of the GA and ACO, several pre-tests, according considered here optimization problem, are performed.

The optimal settings of the GA and ACO parameters are summarized in Tables 1 and 2.

Computer specification to run all identification procedures are Intel Core i5-2329 3.0 GHz, 8 GB Memory, Windows 7 (64bit) operating system and Matlab 7.5 environment.

We perform 30 independent runs of the hybrid GA-ACO. The hybrid algorithm starts with population of 20 chromosomes. We use 40 generation to find solution. We take the achieved best GA solution to update ACO initial pheromone. Further ACO is used to obtain the best model parameters vector using 30 ants for 100 generations.

Table 1 Parameters of GA

Parameter	Value
ggap	0.97
xovr	0.7
mutr	0.05
Maximum generations (maxgen)	40
Number of individuals (nind)	20
Number of variables	3
Inserted rate	100 %

Table 2 Parameters of ACO algorithm

Parameter	Value
Number of ants (nind)	20
Initial pheromone	0.5
Evaporation	0.1
Maximum generations (maxgen)	100

Table 3 Results from model parameters identification procedures

Value	Algorithm	Algorithm performance	
		T, [s]	J
Best	GA	67.5172	4.4396
	ACO	67.3456	4.9190
	GA-ACO	38.7812	4.3803
	ACO-GA	35.5212	4.4903
Worst	GA	66.5968	4.6920
	ACO	66.6280	6.6774
	GA-ACO	41.4495	4.6949
	ACO-GA	35.3498	4.6865
Average	GA	67.1370	4.5341
	ACO	69.5379	5.5903
	GA-ACO	39.4620	4.5706
	ACO-GA	36.1313	4.5765

For comparison of hybrid performance pure GA and pure ACO are run (30 times) with parameters shown in Tables 1 and 2.

The main numerical results, from parameter identification, are summarized in Table 3. The obtained average values of the model parameters (μ_{max}, k_S and $Y_{S/X}$) are summarized in Table 4.

Table 4 Parameters' estimations of the *E. coli* fed-batch cultivation process model

Value	Algorithm	Model parameters		
		μ_{max}	k_S	$1/Y_{S/X}$
Average	GA	0.4857	0.0115	2.0215
	ACO	0.5154	0.0151	2.0220
	GA-ACO	0.4946	0.0123	2.0204
	ACO-GA	0.4976	0.0135	2.0221

As it can be seen from Table 3 the hybrid GA-ACO achieves similar to pure GA and pure ACO algorithm values of the objective function. In the same time, the running time of the proposed hybrid algorithm is about two times less. The pure ACO algorithm starts with equal initial pheromone for all problem elements. In the case of hybrid GA-ACO we use the best found solution by the GA to update the ACO pheromone. Thus our ACO algorithm uses the GA "experience" and starts from "better" pheromone. This strategy helps to the ants to find good solutions using less computational resources like time and memory. In result our hybrid algorithm uses more than three times less memory than pure ACO and pure GA algorithms.

Moreover, in Table 3 we compare achieved in this work results with results in our previous work [34]. There we run the ACO algorithm for several iterations and thus we generate initial populations for GA algorithm. Thus the GA starts from population closer to the good (optimal) solution than the randomly generated population. We observe that ACO-GA and GA-ACO algorithms achieves very similar results for a similar running time. We run the ANOVA test to measure the relative difference between two algorithms. The two hybrid algorithms achieves statistically equivalent results, but the GA-ACO algorithm uses 30 % less memory. Thus we can conclude that hybrid GA-ACO algorithm performs better than ACO-GA hybrid algorithm.

On Fig. 7 the comparison of the dynamics of measured and modeled biomass concentration is shown. With a line we show the modeled biomass during the cultivation process and with stars we show the measured biomass concentration. We put only several stars because the two line are almost overlapped. On Fig. 8 the comparison between time profiles of measured and modeled substrate concentration

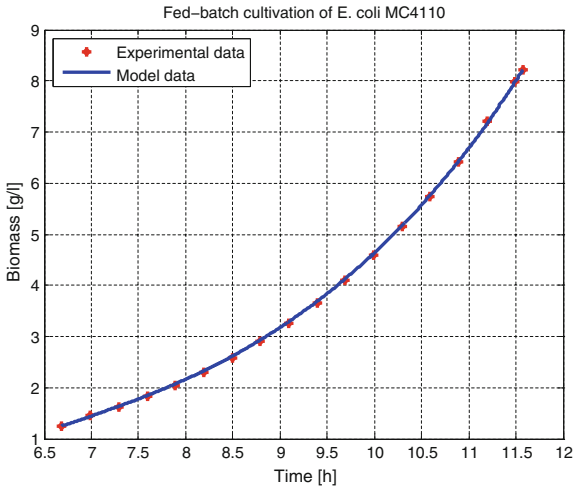


Fig. 7 Comparison between measured and modeled biomass concentration

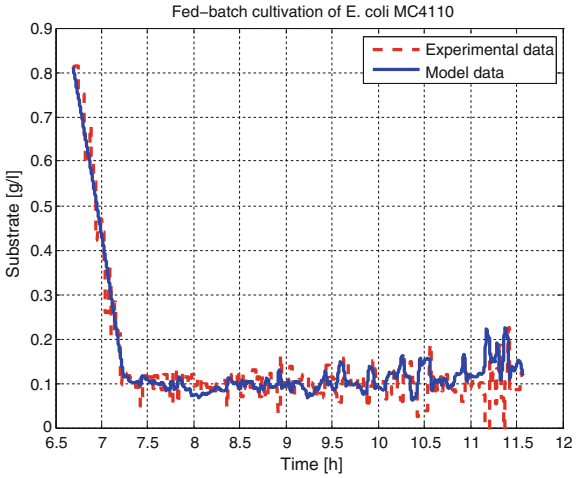


Fig. 8 Comparison between measured and modeled substrate concentration

during the cultivation process is shown. On the both figures we observe how close are the modeled and measured data. Thus we show the quality of our hybrid GA-ACO algorithm.

5.2 InterCriteria Analysis

Based on the obtained results from the identification procedures the following IM is defined:

$$A = \begin{array}{c|cccc} & \text{GA} & \text{ACO} & \text{GA-ACO} & \text{ACO-GA} \\ \hline T_{ave} & 67.1370 & 69.5379 & 39.4620 & 36.1313 \\ J_{ave} & 4.5341 & 5.5903 & 4.5706 & 4.5765 \\ J_{best} & 4.4396 & 4.9190 & 4.3803 & 4.4903 \\ J_{worst} & 4.6920 & 6.6774 & 4.6949 & 4.6865 \\ \mu_{max_{ave}} & 0.4857 & 0.5154 & 0.4946 & 0.4976 \\ k_{S_{ave}} & 0.0115 & 0.0151 & 0.0123 & 0.0135 \\ 1/Y_{S/X_{ave}} & 2.0215 & 2.0220 & 2.0204 & 2.0221 \\ \text{nind} & 100 & 20 & 30 & 10 \\ \text{maxgen} & 100 & 20 & 30 & 10 \end{array} \tag{13}$$

In the IM A (13) the average values for computation time (T_{ave}) and for the three model parameters estimations ($\mu_{max_{ave}}$, k_{Save} and $1/Y_{S/X_{ave}}$), in case of GA, ACO, GA-ACO and ACO-GA, are presented. Moreover, population number (individuals and/or ants) (nind), algorithm generations (maxgen) and the average, best and worst value of the objective function (J_{ave}, J_{best}, J_{worst}) Eq. (6) are considered.

Resulting IMs that determine the degrees of "agreement" (μ) and "disagreement" (ν) between criteria are follows:

μ	T_{ave}	J_{ave}	J_{best}	J_{worst}	$\mu_{max_{ave}}$	k_{Save}	$1/Y_{S/X_{ave}}$	nind	maxgen
T_{ave}	**1**	0.5	0.67	0.83	0.5	0.5	0.5	0.67	0.67
J_{ave}	0.5	**1**	0.83	0.67	1	1	0.67	0.17	0.33
J_{best}	0.67	0.83	**1**	0.5	0.83	0.83	0.83	0.33	0.5
J_{worst}	0.83	0.67	0.5	**1**	0.67	0.67	0.33	0.5	0.5
$\mu_{max_{ave}}$	0.5	1	0.83	0.67	**1**	1	0.67	0.17	0.33
k_{Save}	0.5	1	0.83	0.67	1	**1**	0.67	0.17	0.33
$1/Y_{S/X_{ave}}$	0.5	0.67	0.83	0.33	0.67	0.67	**1**	0.17	0.33
nind	0.67	0.17	0.33	0.5	0.17	0.17	0.17	**1**	0.5
maxgen	0.67	0.33	0.5	0.5	0.33	0.33	0.33	0.5	**1**

(14)

ν	T_{ave}	J_{ave}	J_{best}	J_{worst}	$\mu_{max_{ave}}$	k_{Save}	$1/Y_{S/X_{ave}}$	nind	maxgen
T_{ave}	**0**	0.5	0.33	0.17	0.5	0.5	0.5	0.33	0
J_{ave}	0.5	**0**	0.17	0.33	0	0	0.33	0.33	0.33
J_{best}	0.33	0.17	**0**	0.5	0.17	0.17	0.17	0.67	0.17
J_{worst}	0.17	0.33	0.5	**0**	0.33	0.33	0.67	0.5	0.17
$\mu_{max_{ave}}$	0.5	0	0.17	0.33	**0**	0	0.33	0.33	0.33
k_{Save}	0.5	0	0.17	0.33	0	**0**	0.33	0.33	0.33
$1/Y_{S/X_{ave}}$	0.5	0.33	0.17	0.67	0.33	0.33	**0**	0.33	0.33
nind	0.33	0.83	0.67	0.5	0.83	0.83	0.83	**0**	0.17
maxgen	0	0.33	0.17	0.17	0.33	0.33	0.33	0.17	**0**

(15)

As expected, every criteria perfectly correlates with itself, so the value μ is always 1, and $\nu = 0$. Also, the two matrices are obviously symmetrical according to the main diagonal. Observing obtained values of "agreement" (μ, Eq. (14)) and "disagreement" (ν, Eq. (15)), we can group the pairs of defined criteria in the following 6 groups.

- $\mu = 1$ and $\nu = 0$
  $\mu_{max_{ave}} - J_{ave}$, $k_{Save} - J_{ave}$, $k_{Save} - \mu_{max_{ave}}$
- $\mu = 0.83$ and $\nu = 0.17$
  $J_{worst} - T_{ave}$, $J_{best} - J_{ave}$, $\mu_{max_{ave}} - J_{best}$, $k_{Save} - J_{best}$, $1/Y_{S/X_{ave}} - J_{best}$
- $\mu = 0.67$ and $0 <= \nu <= 0.33$
  $J_{worst} - J_{ave}$, $nind - T_{ave}$, maxgen $- T_{ave}$, $1/Y_{S/X_{ave}} - J_{ave}$, $J_{best} - T_{ave}$
- $\mu = 0.5$ and $0.17 <= \nu <= 0.5$
  $J_{ave} - T_{ave}$, $\mu_{max_{ave}} - T_{ave}$, $k_{Save} - T_{ave}$, $1/Y_{S/X_{ave}} - T_{ave}$, $J_{worst} - J_{best}$, nind $- J_{worst}$, maxgen $- J_{best}$, maxgen $- J_{worst}$, maxgen $-$ nind
- $\mu = 0.33$ and $0.33 <= \nu <= 0.67$
  maxgen $- J_{ave}$, nind $- J_{best}$, $1/Y_{S/X_{ave}} - J_{worst}$, $\mu_{max_{ave}}$ $-$ maxgen, k_{Save} $-$ maxgen, $1/Y_{S/X_{ave}}$ $-$ maxgen
- $\mu = 0.17$ and $\nu <= 0.83$
  nind $- J_{ave}$, $\mu_{max_{ave}}$ $-$ nind, k_{Save} $-$ nind, $1/Y_{S/X_{ave}}$ $-$ nind

The high value of "agreement" of the parameters values from the first two groups confirms the robustness of the proposed algorithms. At the same time the strong connection in pair $k_{Save} - \mu_{max_{ave}}$ derives from the physical meaning of these model parameters [13]. In most of the cases the sum of the values of μ and ν is 1. However, sometimes this sum is less than 1, therefore there is some uncertainty. This uncertainty could be explained with the stochastic nature of the applied algorithms. The values of μ in the 4th group show the correctness of the algorithms. The value of "agreement" between the average value of the objective function and computation time, or worst and best value of the objective function is not very high, but it exists. The last group shows that the dependence between number of population and achieved objective function value is less important than with running time. The running time depends on both—the number of population and the number of algorithm generations. Thus we can conclude that for achieving good results the balance between number of population and number of generations is very important. We observe that the worst value of the objective function depends much more of the running time than the best objective function value. If we run the algorithm for a short running time with high probability we will achieve bad solutions only, at the same time long running time can not guarantee achieving of good solutions.

On Fig. 9 with stars are shown the pairs $\langle \mu, \nu \rangle$. When there is not uncertainty, the stars are on the diagonal. In the case of uncertainty the stars are under the diagonal. The uncertainties are in 30 % of the cases and thus the robustness of our algorithms is confirmed.

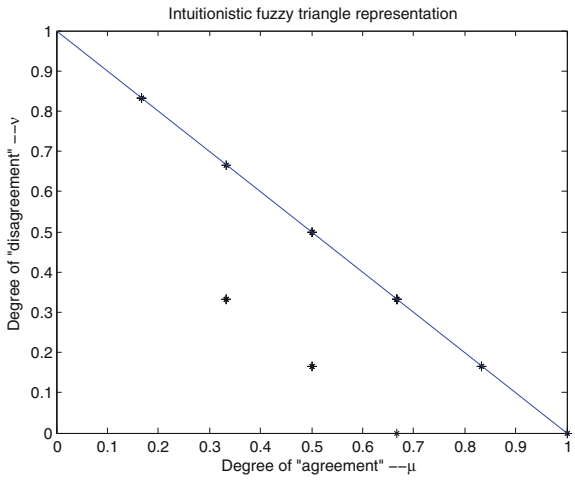


Fig. 9 Intuitionistic fuzzy *triangle* representation

6 Conclusion

In this paper we apply a new approach—InterCriteria analysis—for establishing relations and dependencies between different algorithm parameters and model parameters. We considered two hybrid algorithms—GA-ACO and ACO-GA—as well as pure GA and pure ACO algorithms.

First, algorithms are applied for parameter identification of nonlinear mathematical model of *E. coli* fed-batch cultivation process. We observe that our hybrid algorithms (GA-ACO and ACO-GA) achieve similar to pure GA and ACO algorithms solutions using less computational resources like time and memory. Both hybrid algorithms achieve statistically similar results for a similar running time, but GA-ACO algorithm uses 30 % less memory, which is important when we solve large problems.

Second, InterCriteria analysis is performed to determine the levels of dependence between *E. coli* process model parameters themselves. Then between algorithms outcomes as computational time and accuracy, as well as the number of used populations and maximum number of algorithms generations. Next we determine the levels of dependence between *E. coli* process model parameters and the considered algorithms' parameters. This analysis shows some relations and dependencies that result from the physical meaning of the model parameters—on the one hand, and from stochastic nature of the considered metaheuristics—on the other hand. Moreover, the results show the robustness of the proposed algorithms (both hybrid and pure techniques) and confirm their correctness.

Acknowledgments Work presented here is a part of the Poland-Bulgarian collaborative Grant "Parallel and distributed computing practices" and the Bulgarian National Scientific Fund under the grants DFNI-I02/20 "Efficient ParallelAlgorithms for Large Scale Computational Problems and DFNI-I02/5 InterCriteria Analysis". A New Approach to Decision Making.

References

1. A. Acan, A GA + ACO hybrid for faster and better search capability, in *Ant Algorithms: Proceedings of the Third International Workshop, ANTS 2002*. Lecture Notes in Computer Science (2002)
2. S. AlMuhaideb, M. El, B. Menai, A new hybrid metaheuristic for medical data classification. Int. J. Metaheuristics **3**(1), 59–80 (2014)
3. M. Arndt, B. Hitzmann, Feed forward/feedback control of glucose concentration during cultivation of *Escherichia coli*, in *8th IFAC International Canada, Conference on Computer Applications in Biotechnology*, Canada, 2001, pp. 425–429
4. K. Atanassov, D. Mavrov, V. Atanassova, InterCriteria decision making: a new approach for multicriteria decision making, based on index matrices and intuitionistic fuzzy sets. Issues IFSs GNs **11**, 1–8 (2014)
5. K. Atanassov, E. Szmidt, J. Kacprzyk, In intuitionistic fuzzy pairs. Notes Intuitionistic Fuzzy Sets **19**(3), 1–13 (2013)
6. K. Atanassov, Generalized index matrices. C. R. Acad. Bulg. Sci. **40**(11), 15–18 (1987)

7. K. Atanassov, On index matrices, part 1: standard cases. Adv. Stud. Contemp. Math. **20**(2), 291–302 (2010)
8. K. Atanassov, On index matrices, part 2: intuitionistic fuzzy case. Proc. Jangjeon Math. Soc. **13**(2), 121–126 (2010)
9. V. Atanassova, D. Mavrov, L. Doukovska, K. Atanassov, Discussion on the threshold values in the InterCriteria decision making approach. Notes IFS **20**(2), 94–99 (2014)
10. V. Atanassova, L. Doukovska, K. Atanassov, D. Mavrov, InterCriteria decision making approach to EU member states competitiveness analysis. BMSD 289–294 (2014)
11. V. Atanassova, L. Doukovska, D. Karastoyanov, F. Capkovic, InterCriteria decision making approach to EU member states competitiveness analysis: trend analysis. Proceedings of IEEE (2014), pp. 107–115
12. V. Atanassova, I. Vardeva, Sum- and average-based approach to criteria shortlisting in the InterCriteria analysis. Notes IFS **20**(4), 41–46 (2014)
13. G. Bastin, D. Dochain, *On-line Estimation and Adaptive Control of Bioreactors* (Elsevier Science Publisher, Amsterdam, 1991)
14. M. Battarra, A.A. Pessoa, A. Subramanian, E. Uchoa, Exact algorithms for the traveling salesman problem with draft limits. Eur. J. Oper. Res. **235**(1), 115–128 (2014)
15. C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003)
16. E. Bonabeau, M. Dorigo, G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems* (Oxford University Press, New York, 1999)
17. I. Boussaid, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics. Inf. Sci. **237**, 82–117 (2013)
18. A. Csebfalv, A hybrid metaheuristic method for continuous engineering optimization. Civ. Eng. **53**(2), 93–100 (2009)
19. M. Dorigo, T. Stutzle, *Ant Colony Optimization* (MIT Press, Cambridge, 2004)
20. I. Dumitrescu, T. Sttzle, Combinations of local search and exact algorithms, in *Applications of Evolutionary Computation*, ed. by G.R. Raidl et al. Lecture Notes in Computer Science, vol. 2611 (Springer 2003), pp. 211–223
21. Genetic Algorithms, http://www.doc.ic.ac.uk/nd/surprise_96/journal/vol1/hmw/article1.html. Accessed 14 Jan 2015
22. A. Georgieva, I. Jordanov, Hybrid metaheuristics for global optimization using low-discrepancy sequences of points. Comput. Oper. Res. **37**(3), 456–469 (2010)
23. F. Glover, G. Kochenberger (eds.), *Handbook of Metaheuristics*. International Series in Operations Research and Management Science, vol. 57 (Kluwer Academic Publishers, Boston, 2003)
24. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison Wesley Longman, London, 2006)
25. H. Guangdong, P. Ling, Q. Wang, A hybrid metaheuristic ACO-GA with an application in sports competition scheduling, in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, vol. 3, (2007), pp. 611–616
26. H. Guangdong, Q. Wang, in *Chapter 7, A Hybrid ACO-GA on Sports Competition Schedulingby, Ant 451 Colony Optimization—Methods and Applications*, ed. by A. Ostfeld, (InTech, 2011), pp. 89–100
27. N. Harvey, Use of heuristics: insights from forecasting research. Think. Reason. **13**(1), 5–24 (2007)
28. J.H. Holland, *Adaptation in Natural and Artificial Systems*, 2nd edn. (MIT Press, Cambridge, 1992)
29. M. Lukasiewycz, M. Gla, F. Reimann, J. Teich, Opt4J—a modular framework for metaheuristic optimization, in *Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2011)*, Dublin, Ireland, (2011), pp. 1723–1730
30. S. Masrom, S.Z.Z. Abidin, P.N. Hashimah, A.S. Abd, Rahman, Towards rapid development of user defined metaheuristics hybridisation. Int. J. Softw. Eng. Appl. **5**, 1–12 (2011)

31. O. Roeva, T. Pencheva, B. Hitzmann, St. Tzonkov, A genetic algorithms based approach for identification of escherichia coli fed-batch fermentation. Int. J. Bioautomation **1**, 30–41 (2004)
32. O. Roeva, S. Fidanova, Chapter 13. a comparison of genetic algorithms and ant colony optimization for modeling of E. coli cultivation process, in *Real-World Application of Genetic Algorithms*, ed. by O. Roeva (InTech, 2012), pp. 261–282
33. O. Roeva, *Improvement of Genetic Algorithm Performance for Identification of Cultivation Process Models*. Advanced Topics on Evolutionary Computing, Book Series: Artificial Intelligence Series—WSEAS (WSEAS Press, Bulgaria, 2008), pp. 34–39
34. O. Roeva, S. Fidanova, V. Atanassova, Hybrid ACO-GA for parameter identification of an E. coli cultivation process model, in *Large-Scale Scientific Computing*. Lecture Notes in Computer Science, vol. 8353 (Springer, Germany, 2014), pp. 288–295. ISSN 0302–9743
35. H. Smith, Use of the anchoring and adjustment heuristic by children. Curr. Psychol. J. Divers. Perspect. Divers. Psychol. Issues **18**(3), 294–300 (1999)
36. E.G. Talbi (ed.), *Hybrid Metaheuristics*. Studies in Computational Intelligence, vol. 434 (Springer, Berlin, 2013) (XXVI, 458 p. 109 illus)
37. E.G. Talbi, A taxonomy of hybrid metaheuristics. J. Heuristics **8**, 541–564 (2002)
38. J. Toutouh, Metaheuristics for Optimal Transfer of P2P Information in VANETs, MSc Ph.D. thesis, University of Luxembourg, 2010
39. G.J. Woeginger, *Exact Algorithms for NP-Hard Problems: A Survey*. Lecture Notes in Computer Science, vol. 2570 (Springer, Berlin, 2003), pp. 185–207
40. H. Yi, Q. Duan, T. Warren Liao, Three improved hybrid metaheuristic algorithms for engineering design optimization. Appl. Soft Comput. **13**(5), 2433–2444 (2013)

A Two-Stage Look-Ahead Heuristic for Packing Spheres into a Three-Dimensional Bin of Minimum Length

Hakim Akeb

Abstract In this work we propose a two-stage look-ahead heuristic for packing a given set of spheres into a three-dimensional bin of fixed height and depth but variable length. The problem consists to pack all the spheres into the bin of minimum length. This problem is also known under the name of three-dimensional strip packing problem. The computational results conducted on a set of benchmark instances taken from the literature indicates that the proposed method is effective since it improves most of the best known results by finding new upper bounds for the length of the bin.

1 Introduction

Packing spheres can be used to model many solid state systems. Indeed, the association of different-sized spheres can for example approximate a given solid form. Packing identical and non-identical spheres is for example used in the domain of stereotactic radio surgery radiation therapy (see for example the works of Gavriliouk [4], Sutou and Dai [15], and Wang [17]) where the target areas are delimited by spheres of different sizes. Random sphere packing is used in physics in order to approximate granular materials (see for example Li and Ji [9]).

The problem of packing spheres into an open-dimension bin, also known as the Three-Dimensional Strip Packing Problem (3DSPP) is NP-Hard [7] and can be stated as follows. Given a set S containing n spheres s_i, $1 \leq i \leq n$ where each sphere has radius r_i and is placed with its center at coordinates (x_i, y_i, z_i) in the Euclidean space. Let also $\mathbb{B}$ be a three-dimensional bin (rectangular cuboid or parallelepiped) of fixed height and depth (H, D) respectively but of variable length L. The objective is then to place the n spheres inside the parallelepiped of minimum length such that no sphere overlaps another sphere and no sphere exceeds the container boundaries. The method proposed in this work is based on the use of several tools including the *Maximum Hole Degree* (MHD) heuristic, a modified look-ahead strategy, and an interval search.

H. Akeb (✉)
ISC Paris Business School, 22 boulevard du Fort de Vaux, 75017 Paris, France
e-mail: hakeb@iscparis.com

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_8

2 Literature Review

Even if the 2D-Packing problems are very-well studied in the literature, the three-dimensional versions are less studied. In sphere packing problems the spheres may be of identical or different sizes (radii). The problem of packing non-identical spheres into a given three-dimensional (3D) container was for example considered by Li and Ji [9] where a dynamics-based collective method for random sphere packing was proposed as well as an application to the problem of packing spheres into a cylinder container. The authors studied also the stability of the method and the convergence of their algorithm. Sutou and Dai [15] used a global optimization approach (including Linear Programming relaxation and branch-and-bound) in order to pack unequal spheres inside a three-dimensional container. More precisely, the objective is to maximize the volume of the container (of fixed size) occupied by the placed spheres, this consists then to maximize the "density" of the packing that is equal to the sum of volumes of the placed spheres divided by the volume of the container. This is also called the *Knapsack* version of the problem, i.e., the objective is not to place all the objects but those maximizing the obtained profit. The profit used often corresponds to the volume of the corresponding objects placed. Stoyan et al. [14] developed a mathematical model in order to place different-sized spheres inside a parallelepiped of fixed length and width but with variable height. The objective is then to minimize the height of the container. The proposed method uses different tools including extreme points and neighborhood search. Solutions are given for a set containing eight instances (designed by the authors) where the number of spheres varies from 20 to 60. Farr [3] studied the problem of random close packing fractions of lognormal distributions of hard spheres. The author employed a one-dimensional algorithm for predicting close packing of spheres of lognormal distributions of sphere sizes.

For the case of identical spheres, M'Hallah et al. [11] proposed a Variable Neighborhood Search (VNS) coupled with a Non-Linear Programming (NLP) in order to place identical spheres into the smallest containing sphere. VNS consists here to move some spheres situated in the neighborhood of a given placed sphere, then a NLP procedure is called in order to remove overlapping between spheres. M'Hallah and Alkandari [12] applied the same principle (VNS and NLP) as in [11] but to solve the problem of packing unit spheres into the smallest cube. Soontrapa and Chen [13] considered the problem of packing identical spheres into a cube by using a random-search technique based on the Monte Carlo method. The problem concerns actually the development of a fuel catalyst layer. Vance [16] proved an asymptotic lower bound for the sphere density that improves previous known lower bounds by considering the Hurwitz lattice sphere packing density. Lochmann et al. [10] proposed a statistical analysis for packing random spheres with variable radius distribution.

Finally Birgin and Sobral [2] studied the problem of packing identical and non-identical spheres into different three-dimensional containers. The objective of the work is to minimize the dimension of the container. The method proposed by the authors is based on twice-differentiable models as well as on non-linear programming.

The problem to solve in this paper is the Three-Dimensional Strip Packing Problem (3DSPP) that consists to pack (without overlapping) a given set containing n spheres of known radii into a bin of fixed depth D and height H but unlimited length L. The goal is then to minimize the value of L. The proposed method is essentially based on the look-ahead strategy that explores here a new idea based on the use of a two-level search.

3 Problem Formulation

The spheres are to be placed inside a three-dimensional bin denoted by $\mathbb{B}$ that has six faces $\mathbb{F} = \{\text{left, top, right, bottom, back, front}\}$. Moreover, the bin is placed such that its bottom-left-back corner corresponds to the origin $O(0, 0, 0)$ of the axes in the Euclidean space as shown in Fig. 1. The length L, the height H, and the depth D of the container are associated with the $\overrightarrow{Ox}$, $\overrightarrow{Oy}$, and $\overrightarrow{Oz}$ axes respectively. In addition, each sphere $s_i \in S$ has radius r_i and its center's coordinates are (x_i, y_i, z_i). The 3DSPP can then be formulated as follows:

$$\min \; L \tag{1}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \geq (r_i + r_j)^2 \quad \text{for } 1 \leq i < j \leq n \tag{2}$$

$$x_i \geq r_i \qquad \forall i \in [1, \ldots, n] \tag{3}$$

$$x_i \leq L - r_i \qquad \forall i \in [1, \ldots, n] \tag{4}$$

$$y_i \geq r_i \qquad \forall i \in [1, \ldots, n] \tag{5}$$

$$y_i \leq H - r_i \qquad \forall i \in [1, \ldots, n] \tag{6}$$

$$z_i \geq r_i \qquad \forall i \in [1, \ldots, n] \tag{7}$$

$$z_i \leq D - r_i \qquad \forall i \in [1, \ldots, n] \tag{8}$$

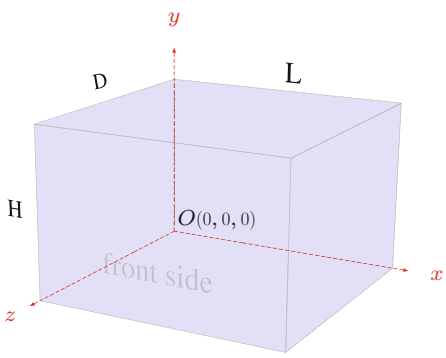


Fig. 1 The three-dimensional bin container placed with its bottom-left-back corner at the origin of the axes in the Euclidean space

Equation 1 indicates the objective (value) to minimize (the length L of the bin). Equation 2 is the non-overlapping constraint that verifies that any pair of distinct spheres $(s_i, s_j) \in S^2$ do not overlap each other. Equations 3–8 mean that each sphere must not exceed the boundaries of the container.

The distance between the edges of two distinct spheres s_i and s_j, denoted by $d_{i,j}$, is defined as follows:

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} - r_i - r_j \quad \text{for } i \neq j. \tag{9}$$

4 The 3DMHD Heuristic for Packing Spheres into a Three-Dimensional Bin

In this section, a greedy heuristic, denoted by 3DMHD (Three-Dimensional Maximum Hole Degree), for packing spheres into a three-dimensional bin is given. This corresponds to the adaptation of the Maximum Hole Degree (MHD) heuristic [5], designed for packing circles (two-dimensional case), to the three-dimensional case.

With the MHD heuristic, a simple way to pack the spheres inside the container consists for example to place the first sphere $s_1 \in S$ (the set of spheres to pack) at the bottom-left-back corner, i.e., at coordinates (r_1, r_1, r_1). After that, at each step i, $(1 < i \leq n)$ a new sphere is chosen and is placed at the *best* position (that has the maximum hole degree). So at each step we have to compute the possible positions for the spheres that are not yet placed by using the spheres already placed and the boundaries of the bin.

More precisely, let:

- S_{in} define the set of spheres already placed inside the container.
- S_{out} is the complementary set containing the spheres that are not yet placed (outside). Note that $S_{\text{in}} \cup S_{\text{out}} = S$.
- P denotes the set of possible positions (called *corner positions*) for the spheres of set S_{out}.

Figure 2 shows an example where two spheres s_1 and s_2 (the two greatest ones) are already placed inside the container $\mathbb{B}$, we have then $S_{\text{in}} = \{s_1, s_2\}$. The figure also indicates six possible corner positions for packing another sphere $s_3 \in S_{\text{out}}$. These positions are denoted by $\{p_3^1, \ldots, p_3^6\}$. Each position p_3^k is computed by using three elements, an element may be a sphere already placed or one of the six faces of the parallelepiped. These three elements correspond to set $\mathscr{T}(p_3^k)$ associated to this position. For example, position p_3^1 is computed by using sphere s_1, the left-edge and the back-face of the container, then $\mathscr{T}(p_3^1) = \{s_1, \text{left}, \text{back}\}$. Similarly, $\mathscr{T}(p_3^2) = \{s_1, s_2, \text{left}\}$.

Generally, let position $p_{i+1}^k \in P$, associated to a sphere of radius r_{i+1}^k, be one of the possible corner positions for the next sphere s_{i+1} to place. Then, the 3DMHD value for position p_{i+1}^k is defined and computed as follows:

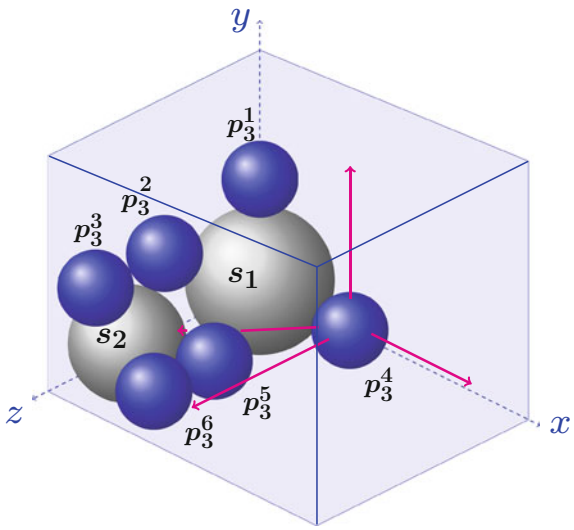


Fig. 2 The 3DMHD heuristic for packing spheres into a three-dimensional bin

$$\lambda(p_{i+1}^k) = \max_{j \in S_{\text{in}} \cup \mathbb{F} \setminus \mathscr{T}(p_{i+1}^k)} 1 - \frac{d_{i+1,j}^k}{r_{i+1}^k} \tag{10}$$

Equation 10 indicates that the hole degree $\lambda(p_{i+1}^k)$ is computed for each position in the set of positions P (associated with set S_{out}) for the next sphere to place. This value uses the distance $d_{i+1,j}^k$ between the edge of position p_{i+1}^k (that is similar to a sphere) and the nearest object j in the set $S_{\text{in}} \cup \mathbb{F} \setminus \mathscr{T}(p_{i+1}^k)$ that contains the spheres already placed, the six faces ($\mathbb{F}$) of the container but excluding set $\mathscr{T}(p_{i+1}^k)$. The distance is divided by the radius r_{i+1}^k of the sphere corresponding to position p_{i+1}^k. Note that if a given position touches more than three objects, then $\lambda = 1$, meaning that this positions has a high probability to be chosen for placing the next sphere.

For example, Fig. 2 indicates the distance between position p_3^4 and four other objects: sphere s_2, the front face, the top face, and finally the right face of the container.

So at each step, the 3DMHD heuristic places the next sphere at position $p^* \in P$ that corresponds to the maximum value of $\lambda(p_{i+1}^k)$ as indicated in Eq. 11.

$$p^* = \arg\max_{p_{i+1}^k} \lambda(p_{i+1}^k) \tag{11}$$

Algorithm 1 explains how the 3DMHD heuristic proceeds in order to place a set of spheres inside the container $\mathbb{B}$ of dimensions $(L \times H \times D)$. Procedure 3DMHD receives a partial solution $\{S_{\text{in}}, S_{\text{out}}, P\}$ indicating the spheres already packed into

Require: Set S_{in} containing spheres already placed, S_{out} containing the remaining spheres to place, set P indicating the possible positions for spheres in S_{out} and the current length L of the container.
Ensure: TRUE if all the spheres are packed into the container, FALSE otherwise.

1: $i \leftarrow |S_{\text{in}}|$;
2: **while** $(P \neq \emptyset)$ **do**
3: Compute/update the 3DMHD value for each corner position $p \in P$;
4: Place the next sphere s_{i+1} at position p^* that has the maximum hole degree as shown in Eq. 11.
5: Move sphere s_{i+1} from S_{out} to S_{in};
6: Remove from set P the positions that overlap the new inserted sphere;
7: Compute new positions by using the new inserted sphere and the other objects already placed;
8: $i \leftarrow i + 1$;
9: **end while**
10: **if** $(i = n)$ **then**
11: Set $L \leftarrow \max(x_i + r_i)$;
12: Update the best known length if L is smaller than this value;
13: **return** TRUE;
14: **else**
15: **return** FALSE;
16: **end if**

Algorithm 1: The 3DMHD greedy heuristic

the container, the remaining spheres, and the set of corner positions for spheres in S_{out} respectively. The current length L of the container is also transmitted to the procedure. The heuristic's output is a boolean value indicating whether yes or no all the spheres were successfully packed into the container. So procedure 3DMHD is able to start with any partial solution where the number of spheres already packed is greater than or equal to zero.

At line 1 in Algorithm 1 counter i indicating the number of spheres already packed is set to the number of spheres inside S_{in}. After that, in the **while** loop, the 3DMHD value is computed for each position $p \in P$ (line 3), this is done by using the formula of Eq. 10. At line 4, the best position p^* is chosen in order to place the next sphere s_{i+1}. After that, the new sphere moves from set S_{out} to set S_{in} (line 5) and the set of positions P is updated by removing those overlapping the new inserted sphere (line 6) and by computing new positions by using the new inserted sphere (line 7). Counter i is then incremented at line 8. The **while** loop ends when the set of positions P becomes empty meaning that no additional sphere can be packed. Then two cases can be distinguished: if $i = n$ then all the n spheres were successfully packed into the container. In this case the procedure computes at line 11 the exact value for L which is equal to $\max(x_i + r_i)$, i.e., using the most right placed sphere $s_i \in S_{\text{in}}$. If the obtained value L is smaller than the best known length then this value is updated (line 12) and the procedure returns TRUE (line 13). If $i < n$ then a feasible packing was not obtained and the procedure returns FALSE (line 15), this means that the current length L of the container has to be changed.

Note that one can test several values for the length L of the bin in order to try to compute a feasible solution with the 3DMHD heuristic (not necessarily a binary search but other more efficient strategies). This can be done for example by decreasing the length from an upper bound to a lower bound. Indeed, this strategy may escape from local optima (see Sect. 5).

4.1 A Multi-Level Look-Ahead strategy for the 3DSPP

This section describes a look-ahead algorithm designed for the three-dimensional strip packing problem.

Look-ahead (LA) strategies (see for example [1, 5, 8]) are often used in order to improve the results obtained by different algorithms. Its objective is to evaluate the future behavior of a decision (choice) made at a given step of the problem solving process. For example, in a greedy algorithm, the *best* decision among all the possible decisions is made at step i in order to move to the next step $i + 1$. The look-ahead strategy tries then several (or all) choices at step i and see what will be obtained when executing the greedy algorithm few steps ahead of until the end (this is often executed on a copy of the partial solution). After that, the decision actually made at step i is the one that had the best behavior or led to the best outcome.

In packing problems, the look-ahead strategy often uses a parameter called *density* of a solution. The density of a solution S_{in}, denoted by $density(S_{\text{in}})$ is equal to the sum of the volumes of spheres in S_{in} divided by the volume of the container as indicated in Eq. 12. The look-ahead strategy selects then the decision that will obtain the highest density.

$$density(S_{\text{in}}) = \frac{4 \times \pi}{3 \times L \times H \times D} \times \sum_{i=1}^{|S_{\text{in}}|} (r_i^3) \tag{12}$$

The algorithm that implements the look-ahead strategy, denoted by LA-3DMHD, is described in Algorithm 2. It receives as input parameters a partial solution $\{S_{\text{in}}, S_{\text{out}}, P\}$ where $|S_{\text{in}}|$ spheres are already packed, set S_{out} denotes the spheres that remain to pack and P contains the corner positions for spheres of set S_{out}. The algorithm receives also the current length (L) of the container. Algorithm LA-3DMHD returns TRUE if it succeeds to compute a feasible solution, FALSE otherwise.

Instruction at line 1 of Algorithm 2 sets the counter i indicating the number of spheres already packed. At line 2, a boolean value (found) is set to FALSE (this indicator is set to TRUE if a feasible solution is obtained).

The difference between the look-ahead strategy and the 3DMHD heuristic (described in Algorithm 1) is that the look-ahead tries (evaluates) several positions at each step of the packing process while the greedy heuristic 3DMHD selects, at each step, only one position (the best one) in order to pack the next sphere. Moreover, the look-ahead used here contains two levels, i.e., it places the two next spheres and continues the placement of the remaining spheres by using the greedy heuristic 3DMHD

Require: Sets S_{in}, S_{out}, P, and the current length L of the container.
Ensure: TRUE if all the spheres are packed into the container, FALSE otherwise.

1: $i \leftarrow |S_{\text{in}}|$;
2: found $\leftarrow$ FALSE;
3: **while** ($P \neq \emptyset$ and found=FALSE) **do**
4: Sort the positions of set P in decreasing order of their hole degree (λ) value;
5: **for all** of the first $\psi_1 \times |P|$ positions $p \in P$ **do**
6: Let $S'_{\text{in}} \leftarrow S_{\text{in}}$, $S'_{\text{out}} \leftarrow S_{\text{out}}$ and $P' \leftarrow P$;
7: Insert the next sphere s'_{i+1} into S'_{in} at position p and update sets S'_{in}, S'_{out}, and P';
8: $density^* \leftarrow 0$;
9: Sort the positions of set P' in decreasing order of their hole degree (λ) value;
10: **for all** of the first $\psi_2 \times |P'|$ positions $p' \in P'$ **do**
11: Let $S''_{\text{in}} \leftarrow S'_{\text{in}}$, $S''_{\text{out}} \leftarrow S'_{\text{out}}$ and $P'' \leftarrow P'$;
12: Insert the next sphere s''_{i+2} into S''_{in} at position p' and update sets S''_{in}, S''_{out}, and P'';
13: found $\leftarrow$ 3DMHD(S''_{in}, S''_{out}, P'', L);
14: **if** (found=TRUE) **then**
15: Set L equal to the length computed by 3DMHD;
16: **return** TRUE;
17: **else**
18: **if** ($density(S''_{\text{in}}) > density^*$) **then**
19: $density^* \leftarrow density(S''_{\text{in}})$;
20: **end if**
21: **end if**
22: **end for**
23: Assign to position $p \in P$ the density $density^*$ obtained after calling 3DMHD;
24: **end for**
25: Let $p^* \in P$ be the position that has obtained the highest density $density^*$;
26: Place the next sphere s_{i+1} at position p^* and move sphere s_{i+1} from S_{out} to S_{in};
27: Remove from set P the positions that overlap the new inserted sphere;
28: Compute new positions by using the new inserted sphere;
29: $i \leftarrow i + 1$;
30: **end while**
31: **if** ($i = n$) **then**
32: Set $L \leftarrow \max(x_i + r_i)$ where x_i and r_i are the $x-$coordinate and the radius of sphere $s_i \in S_{\text{in}}$;
33: Update the best known length if L is smaller than this value;
34: **return** TRUE;
35: **else**
36: **return** FALSE;
37: **end if**

Algorithm 2: LA-3DMHD

(Algorithm 1). This is implemented by using two nested **for** loops that begin at lines 5 and 10 respectively. In addition, the first **for** loop considers only the best $\psi_1 \times |P|$ positions with $0 < \psi_1 \leq 1$ and P is the set of corner positions in the first level. In the second **for** loop the algorithm considers only the best $\psi_2 \times |P'|$ with $0 < \psi_2 \leq 1$ and P' is the set of corner positions in the second level. So if for example $\psi_1 = 0.5$, then only the half best positions in the list of positions are considered in the first level of the look-ahead strategy, and if $\psi_1 = 1$, then this means that all the positions will

be considered. Using a value of ψ_1 and ψ_2 lower than 1 will of course decrease the computation time of the algorithm.

More precisely, the positions in set P are sorted in decreasing order of their hole degree value (λ). This is done at line 4. In the first **for** loop, the algorithm expends the current solution $\{S_{\text{in}}, S_{\text{out}}, P\}$ by choosing at each time a position $p \in P$ by creating a copy of the current solution denoted by $\{S'_{\text{in}}, S'_{\text{out}}, P'\}$ (line 6) and inserts the next sphere s'_{i+1} at that position (line 7). At line 8, a variable called *density** is set to 0. This parameter is used in order to store the best density obtained in the second level of the look-ahead. The corner positions of set P' are after that sorted in decreasing order of their λ value (line 9). The second **for** loop starts at line 10, after placing sphere s'_{i+1}. Like in the first level, only a proportion $\psi_2 \times |P'|$ of the best corner positions are taken into account in set P'. Then for each selected position $p' \in P'$, the procedure creates a copy, denoted by $\{S''_{\text{in}}, S''_{\text{out}}, P''\}$, for the current partial solution $\{S'_{\text{in}}, S'_{\text{out}}, P'\}$ (line 11). After that, the next sphere s''_{i+2} is placed at position p' (line 12). Then, the partial solution is evaluated by calling the 3DMHD heuristic (Algorithm 1) at line 13 in order to try to pack the remaining $n - i - 2$ spheres. If 3DMHD succeeded to pack all the remaining spheres, then it returns TRUE (line 14), the current length of the container is then set to the length computed by 3DMHD (line 15). The algorithm then exits at line 16 since it has succeeded to pack all the spheres (it returns TRUE). Otherwise (found=FALSE), this means that 3DMHD did not succeed to place all the remaining spheres, then the density of the obtained solution *density*(S''_{in}) is assigned to the best known density *density** if a better value is obtained (line 19). The second **for** loop ends when all the selected positions $p' \in P'$ are evaluated and the best obtained density (*density**) is assigned to position $p \in P$ that is currently considered in the first **for** loop.

At the output of the two **for** loops, the next sphere s_{i+1} is placed at position p^* (line 26) that has obtained the best density after calling 3DMHD. The set P of positions is then updated at line 27 by removing those that overlap the new inserted sphere and new positions are computed at line 28. The number of placed spheres (i) is incremented at line 29.

Instructions of the **while** loop (lines 3–30) are executed until a feasible solution is obtained (found=TRUE) or the set of positions P becomes empty. So if $i = n$ (line 31), this means that a feasible solution is reached, then the true length of the container is computed at line 32 and the best known length is updated if a better one is obtained (line 33). The algorithm returns TRUE (line 34). If ($i < n$), then this means that algorithm LA-3DMHD did not succeed to compute a feasible solution and returns FALSE (line 36).

Finally, Algorithm 2 can for example be called by an interval-search procedure that modifies the value of the length L of the container at each call as described in Sect. 4.2.

Require: Instance S containing n spheres, the height H, and the depth D of the three-dimensional bin $\mathbb{B}$;
Ensure: The best length L^* obtained and the corresponding density $density^*$;

1: Set $L_{\min} \leftarrow \max\left(\frac{4\times\pi\times\sum_{i=1}^{n}(r_i^3)}{3\times H\times D}, 2\times r_{\max}\right)$ be the lower bound of the interval search;
2: Set $L_{\max} \leftarrow 3 \times L_{\min}$;
3: Set $\Delta L \leftarrow 0.01$;
4: $L \leftarrow L_{\max}$;
5: $L^* \leftarrow L$;
6: $density^* \leftarrow 0$;
7: **while** $(L \geq L_{\min})$ **do**
8: $S_{\text{in}} \leftarrow \emptyset$;
9: $S_{\text{out}} \leftarrow S$;
10: Create set P of positions corresponding to the placement of each sphere $s_i \in S$ of radius r_i at position (r_i, r_i, r_i) in the bin of dimensions $L \times H \times D$;
11: found $\leftarrow$ LA-3DMHD(S_{in}, S_{out}, P, L);
12: **if** (found = TRUE) **then**
13: Update L if a lower value was obtained by LA-3DMHD;
14: $L^* \leftarrow L$;
15: Update the best density $density^*$;
16: **end if**
17: $L \leftarrow L - \Delta L$;
18: **end while**

Algorithm 3: (LA2)

4.2 Computing the Best Packing by Using Interval Search

This section presents the interval search algorithm, denoted by LA2 and described in Algorithm 3, used in order to compute the best feasible packing. The search principle consists to decrease the value of the bin length L from an upper bound $L_{\max}$ by a given step ΔL until matching the lower bound $L_{\min}$. The search may also stop if the computation time limit is reached.

Algorithm 3 (LA2) explains how the heuristic proceeds in order to compute the best packing of the n spheres into the three-dimensional bin of minimum length. Procedure LA2 receives as input parameters the instance $S = \{s_1, \ldots, s_n\}$ containing n spheres of radii $r_1, \ldots, r_n$ respectively as well as the height H and the depth D of the three-dimensional bin $\mathbb{B}$. The output of the algorithm is the best length found L^* and the corresponding density ($density^*$) that is equal to the sum of the volumes of the spheres divided by the volume of the bin ($L^* \times H \times D$).

The continuous lower bound for the length of the container is used as the minimum value ($L_{\min}$) of the interval search (line 1). Note that if this value is lower than the diameter of the greatest sphere, then this diameter ($2 \times r_{\max}$) is used as the lower bound. The upper bound $L_{\max}$ of the interval search is set equal to $3 \times L_{\min}$. The step ΔL with which the length is decreased at each step is defined at line 3, this value is set to 0.01. The length of the container is then set equal to the upper bound $L \leftarrow L_{\max}$ (line 4) and the best length L^* is set equal to L at line 5. The next instruction serves

to initialize the value of the best known density ($density^*$) associated with the best length L^* (line 6).

After that, at each step in the **while** loop (lines 7–18) a starting configuration is created where the set S_{in} of spheres already packed is set equal to the empty set (line 8) and the set of the remaining spheres to pack (S_{out}) is set equal to the instance S. List P of positions for spheres in set S_{out} is then computed (line 10) so that each position is placed at (r_i, r_i, r_i). This is a novel method because most of the greedy heuristics start by placing one or several objects, here only the list of positions is computed and no object is placed.

Algorithm LA-3DMHD is then called at line 11 in order to try to compute a feasible solution (packing the n spheres into the bin of dimensions $L \times H \times D$). If procedure LA-3DMHD succeeded to pack the n spheres (found=TRUE) then the value of L is updated if a lower value was computed by LA-3DMHD (line 13) and the best length L^* is set equal to L (line 14). The best density $density^*$, corresponding to L^* is then updated at line 15. The value of the length L is after that deceased (line 17), even if a feasible solution was not obtained by procedure LA-3DMHD. Indeed, this method is, to our opinion, preferable to a basic dichotomous search where the dimensions of the container are increased when a feasible solution was not obtained. This is not always a good strategy because, in our case for example, if a feasible solution is not obtained by using a given value of the length L, it may be obtained by using a lower value $L - \Delta L$. In fact, decreasing the value of the length L is a good strategy to escape from local optima in order to increase the solution quality.

Algorithm LA2 stops when the value of L becomes lower than the lower bound L_{min} or when the computation time limit is reached.

5 Computational Results

In order to evaluate the performance of the proposed algorithm LA2 (Algorithm 3), two sets of instances were considered:

- Six instances, denoted by SYS, proposed by Stoyan et al. [14]. The number of spheres varies from 25 to 60. All the spheres have different radii in each instance. These instances are then strongly heterogeneous.
- The second set contains twelve instances, denoted by KBG1, . . . , KBG12, proposed by Kubach et al. [6]. Here, the number of spheres is equal to 30 for the first six instances and 50 for the six last ones. Moreover, instances KBG1–KBG3 and KBG7–KBG9 are strongly heterogeneous since all the radii are different. The other six instances KBG4–KBG6 and KBG10–KBG12 are weakly heterogeneous because there are only $n/10$ different radii in each instance, each radius is duplicated 10 times.

The algorithms and procedures are coded in C++ language and executed under Linux environment on a computer with a 2.4 GHz processor. The results are

compared to those obtained by the B1.6 algorithm [6] that is mainly based on a look-ahead strategy and starting configurations, the results taken from [6] were also obtained on a 2.4 GHz processor. Algorithm B1.6 is in fact the adaptation of algorithm B1.5 [5] for packing circles inside a rectangular container to the three-dimensional case. Algorithm B16 however tries more starting configurations than B1.5 does. In addition, B1.6 uses a parameter denoted by τ ($0 < \tau \leq 1$) that serves to indicate the proportion of corner positions evaluated at each step of the look-ahead process. The authors in [6] tried two values: $\tau = 0.8$ and $\tau = 1$. The first case means that only 80 % of corner positions are evaluated by the look-ahead while the second case means that all positions are evaluated. So in fact, algorithm B1.6 is executed twice (60 min for each value of τ). It is to note that the proposed algorithm LA2 is executed only once during 60 min on each instance.

In algorithm LA2, the number of positions evaluated by the look-ahead is set to 50 % in the two levels ($\psi_1 = \psi_2 = 0.5$). So at each time the corner positions are sorted in decreasing order of their hole degree λ and only the first half ones are evaluated. The objective is of course to save computation time.

The results obtained by the proposed algorithm LA2 are given in two distinct tables. Table 1 gives the results obtained by different algorithms on the first set of instances (SYS) that contains six strongly heterogeneous instances. Column 1 indicates the instance's name and column 2 its size. The two next columns indicate the height H and the depth D of the container. Column 5 (SYS) indicates the results (best length) obtained by the SYS method [14] on instances SYS1–SYS6. Columns 6 and 7 contain the best results (the best length L and the corresponding density *"Dens."* respectively) obtained by algorithm B1.6 on the six instances (SYS) when parameter τ is set equal to 0.8 (80 % of positions are evaluated by the look-ahead). The next two columns display the same results as the two previous columns but when parameter τ is set equal to 1 (all the positions are evaluated in the look-ahead). Columns 10–14 contain the results obtained by the proposed algorithm LA2 on the considered instances. Column 10 (L) gives the best length obtained and column 11 the corresponding density. Column 12 (t^*) indicates the time needed by algorithm LA2 for computing the best solution.

The two last columns of Table 1 indicate the percentage of improvement obtained by the proposed algorithm LA2 on algorithm B1.6. Column "Imp. 0.8" shows the improvement obtained when considering B1.6 with $\tau = 0.8$ and the last column "Imp. 1" is the percentage of improvement when B1.6 with $\tau = 1$ is considered. Note that the percentage of improvement is computed as follows: Imp. $= \frac{Density(\text{LA2}) - Density(\text{B1.6})}{Density(\text{B1.6})}$.

The results of Table 1 indicate that the proposed algorithm LA2 improves all the results obtained by the SYS method on the first six instances. The last row of the table indicates that algorithm LA2 improves B1.6 with 1.68 % in average (column "Imp. 0.8") when $\tau = 0.8$ for B1.6. The improvement in equal to 0.81 % when $\tau = 1.0$.

Figure 3 shows the improvement of the best solution obtained by the proposed algorithm LA2 on instance SYS1 ($n = 30$) spheres. The horizontal axis indicates the cumulative time while the vertical axis indicates the density of the solution (in %).

Table 1 Results obtained by the proposed method LA2 on the six instances SYS

				SYS	B1.6 $\tau = 0.8$ (1 h)		B1.6 $\tau = 1$ (1 h)		Algorithm LA2 (1 h)				
Inst.	n	H	D	L	L	*Dens.*	L	*Dens.*	L	*Dens.*	t^*	Imp. 0.8	Imp. 1.0
SYS1	25	5.5	6.9	9.8668	9.5397	53.160	9.2874	54.604	9.2234	**54.983**	1911	3.43	0.69
SYS2	35	6.5	7.9	9.6221	9.2608	55.077	9.1280	55.878	9.1138	**55.965**	2680	1.61	0.16
SYS3	40	5.5	6.9	9.4729	9.0540	53.554	8.9850	53.965	8.9316	**54.288**	2900	1.37	0.60
SYS4	45	8.5	9.9	11.0862	10.8932	53.771	10.8760	53.856	10.7653	**54.410**	3600	1.19	1.03
SYS5	50	8.5	9.9	11.6453	11.2170	54.975	11.3494	54.334	11.1948	**55.084**	2030	0.20	1.38
SYS6	60	8.5	9.9	12.8416	12.5339	54.346	12.3745	55.046	12.2519	**55.597**	3330	2.30	1.00
Av.						54.147		54.614		**55.055**		1.68	0.81

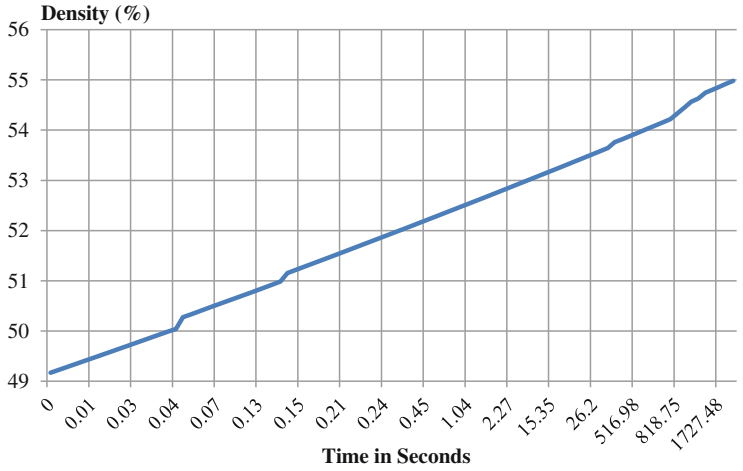


Fig. 3 Improvement of the solution quality obtained by algorithm LA2 on the first instance SYS1 ($n = 30$, *Density* $= 54.983\,\%$)

Note that the improvement is faster during the beginning of the execution but becomes slower after that. This is because the improvement becomes more and more difficult since the quality of the solution becomes better. For example the density moves from 49.171 to 53.655 % during the first 53 s of the execution and reaches the best density (54.983 %) after 1911 s.

Table 2 gives the results obtained by algorithms B1.6 and LA2 on the second set of instance (KBG) that contains twelve examples. The results obtained by algorithm SYS are not known for this set. Columns 1–4 indicates the name of the instance, the number of spheres (n), the height H and the depth D of the container. Column 5 gives the best density *Dens.* obtained by algorithm B1.6 when $\tau = 0.8$ and Column 6 contains the result obtained by the same algorithm with $\tau = 1.0$. Columns 7–11 shows the results obtained by the proposed algorithm LA2. Column 7 (L) gives the best length of the bin computed by LA2 and the next column *Dens.* contains the corresponding density. Column t^* corresponds to the computation time needed to compute the best solution that is given in columns 7 and 8. Columns 10 and 11 contains the percentage of improvement of LA2 when compared to compared B1.6. LA2 outperforms B1.6 when $\tau = 0.8$ in 5 cases, the two algorithms reached optimal solutions on instances KBG2, KBG4, and KBG10. B1.6 remains better than LA2 in 4 cases. The percentage of improvement obtained by LA2 is equal to 0.84 % when $\tau = 0.8$ in B1.6. For $\tau = 1$, LA2 improves 1.6 in six cases, the two algorithms obtains the oprimal solution on instances KBG2, KBG4, and KBG10. B1.6 ($\tau = 1$) remains better than LA2 in three cases. The corresponding percentage of improvement is equal to 0.53 %.

Figure 4 gives the solutions obtained by the proposed algorithm LA2 on the six instances SYS. The number of spheres (n) as well as the obtained density is indicated.

Table 2 Results obtained by the proposed method LA2 on instances KBG

				B1.6 $\tau = 0.8$ (1 h)	B1.6 $\tau = 1$ (1 h)	Algorithm LA2 (1 h)				
Inst.	n	H	D	*Dens.*	*Dens.*	L	*Dens.*	t^*	Imp. 0.8	Imp. 1
KBG1	30	10	10	53.772	54.096	11.2063	**54.494**	2400	1.34	0.74
KBG2	30	10	10	***30.071**	***30.071**	1.9900	***30.071**	2	0.00	0.00
KBG3	30	10	10	50.614	51.387	18.9231	**51.693**	3300	2.13	0.60
KBG4	30	10	10	***37.765**	***37.765**	1.9960	***37.765**	1	0.00	0.00
KBG5	30	10	10	**48.278**	**48.278**	1.9279	48.181	1930	−0.20	−0.20
KBG6	30	10	10	**48.966**	47.792	18.8807	48.847	3400	−0.24	2.21
KBG7	50	10	10	54.623	55.372	13.5075	**55.824**	2030	2.20	0.82
KBG8	50	10	10	44.924	45.060	2.6027	**46.639**	326	3.82	3.50
KBG9	50	10	10	52.210	**52.732**	29.7023	51.783	3420	−0.82	−1.80
KBG10	50	10	10	***51.866**	***51.866**	1.8100	***51.866**	9	0.00	0.00
KBG11	50	10	10	51.629	**52.708**	5.2640	52.658	420	1.99	−0.09
KBG12	50	10	10	**52.120**	51.757	22.2060	52.063	1000	−0.11	0.59
Av.				48.070	48.240		**48.488**		0.84	0.53

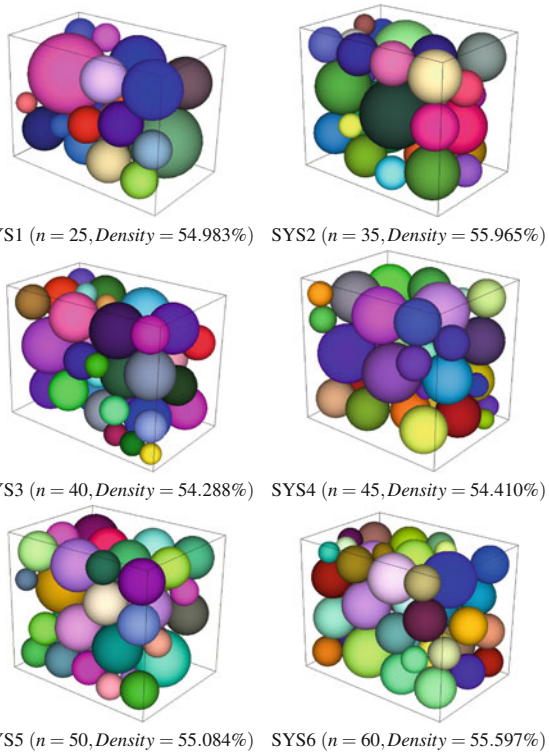


Fig. 4 Solutions obtained by algorithm LA2 on the six instances SYS

Figure 5 gives the solution obtained by algorithm LA2 on nine instances KBG where the solution cannot be proved to be optimal. The number of spheres (n) and the obtained density (in parentheses) is indicated.

Figure 6 gives the optimal solutions obtained by algorithm LA2 on the three instances KBG2, KBG4, and KBG10. The solution is optimal because the length of the bin is exactly equal to the diameter of the greatest sphere in the instance.

6 Conclusion

In this paper, a look-ahead heuristic was proposed in order to solve the problem of packing spheres into a three-dimensional bin of minimum length. Three main new ideas were used. The first one is that the method starts with an empty configuration instead of placing one or several pieces inside the container. The second idea consists to use an interval search that proceeds by decreasing the value of the length of the bin instead of using a dichotomous search, the objective is to escape from local optima. Finally the look-ahead procedure uses a double search (two levels) instead of one level. The obtained results on the two sets of instances demonstrated that the proposed

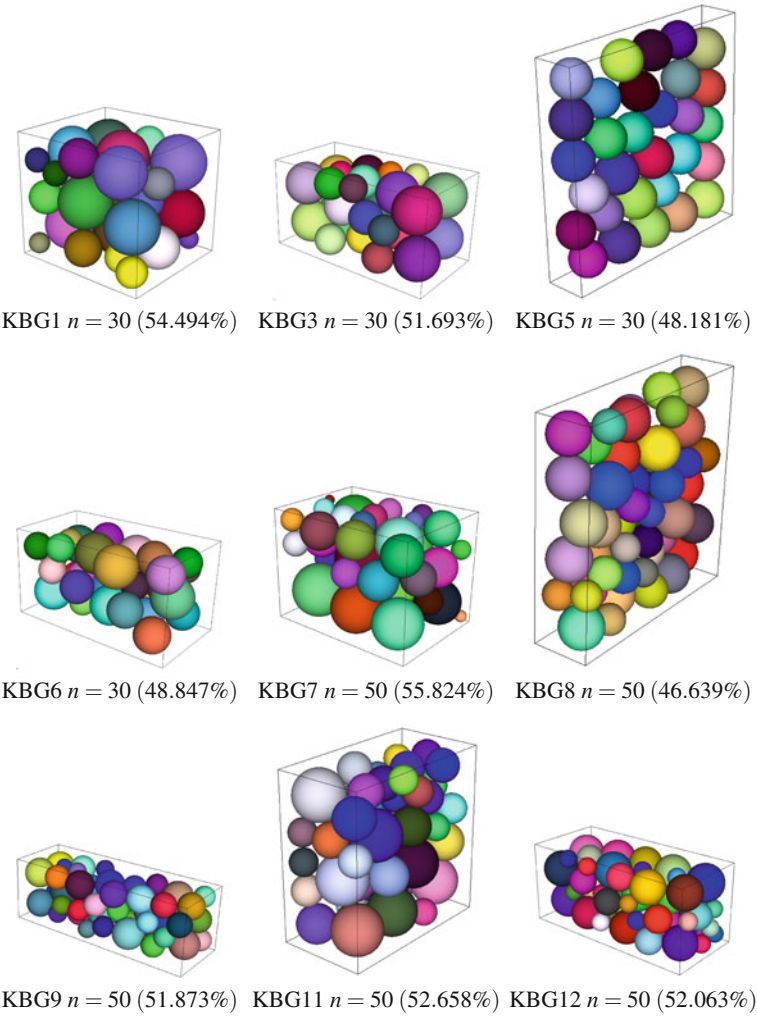


Fig. 5 Solutions obtained by algorithm LA2 on the twelve instances KBG

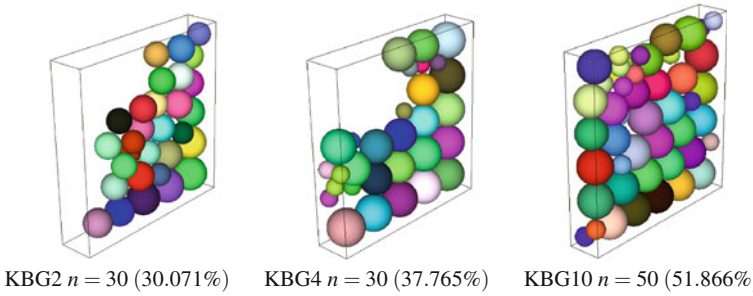


Fig. 6 Optimal solutions obtained by algorithm LA2 on instances KBG2, KBG4, and KBG10

method is effective since it succeeded to improve or reach almost all the best known results published in the literature. As a future work, it will be interesting to design a new heuristic for packing weakly heterogeneous spheres because it is well-known that the MHD heuristic was designed for packing strongly heterogeneous circles and spheres.

References

1. H. Akeb, M. Hifi, D. Lazure, An improved algorithm for the strip packing problem, in *Proceedings of the Federated Conference on Computer Science and Information Systems, FedCSIS* (Wroclaw, Poland, 2012), pp. 357–364. ISBN 978-83-60810-51-4
2. E.G. Birgin, F.N.C. Sobral, Minimizing the object dimensions in circle and sphere packing problems. Comput. Oper. Res. **35**, 2357–2375 (2008)
3. R.S. Farr, Random close packing fractions of lognormal distributions of hard spheres. Powder Technol. **245**, 28–34 (2013)
4. E.O. Gavriliouk, *Unequal Sphere Packing Problem in the Context of Stereotactic Radiosurgery* (Shaker Verlag, 2007), 96 p
5. W.Q. Huang, Y. Li, H. Akeb, C.M. Li, Greedy algorithms for packing unequal circles into a rectangular container. J. Oper. Res. Soc. **56**, 539–548 (2005)
6. T. Kubach, A. Bortfeldt, T. Tilli, H. Gehring, Greedy algorithms for packing unequal sphere into a cuboidal strip or a cuboid. Asia Pac. J. Oper. Res. **28**, 739–753 (2011)
7. L.K. Lenstra, A.H.G. Rinnooy Kan, Complexity of packing, covering, and partitioning problems, in *Packing and Covering in Combinatorics*, ed. by A. Schrijver (Mathematisch Centrum, Amsterdam, 1979), pp. 275–291
8. M. Lin, R. Chen, J.S. Liu, Lookahead strategies for sequential Monte Carlo. Stat. Sci. **28**, 69–94 (2013)
9. Y. Li, W. Ji, Stability and convergence analysis of a dynamics-based collective method for random sphere packing. J. Comput. Phys. **250**, 373–387 (2013)
10. K. Lochmann, L. Oger, D. Stoyan, Statistical analysis of random sphere packings with variable radius distribution. Solid State Sci. **8**, 1397–1413 (2006)
11. R. M'Hallah, A. Alkandari, N. Mladenović, Packing unit spheres into the smallest sphere using VNS and NLP. Comput. Oper. Res. **40**, 603–615 (2013)
12. R. M'Hallah, A. Alkandari, Packing unit spheres into a cube using VNS. Electron. Notes Discret. Math. **39**, 201–208 (2012)
13. K. Soontrapa, Y. Chen, Mono-sized sphere packing algorithm development using optimized Monte Carlo technique. Adv. Powder Technol. **24**(6), 955–961 (2013). doi:10.1016/j.apt.2013.01.007
14. Y. Stoyan, G. Yaskow, G. Scheithauer, Packing of various radii solid spheres into a parallelepiped. Cent. Eur. J. Oper. Res. **11**, 389–407 (2003)
15. A. Sutou, Y. Dai, Global optimization approach to unequal sphere packing problems in 3D. J. Optim. Theory Appl. **114**, 671–694 (2002)
16. S. Vance, Improved sphere packing lower bounds from Hurwitz lattices. Adv. Math. **227**(5), 2144–2156 (2011)
17. J. Wang, Packing of unequal spheres and automated radiosurgical treatment planning. J. Comb. Optim. **3**, 453–463 (1999)

Handling Lower Bound and Hill-Climbing Strategies for Sphere Packing Problems

Mhand Hifi and Labib Yousef

Abstract In this paper the 3-dimensional sphere packing problem is solved by using an iterative tree search-based heuristic. The goal of the problem is to determine a minimum length of the container that contains all available spheres/items without overlapping. Such a length is searched by applying a tree search that combines hill-climbing and bounding strategies. All branches of the tree are created following eligible positions associated to successive items to pack and the bounds are computed by applying a greedy procedure. Because the number of positions is large, the hill-climbing strategy is introduced in order to filter the search by choosing some best paths. The proposed algorithm is evaluated on benchmark instances taken from the literature and on new benchmark instances: the provided results are compared to those reached by recent methods available in the literature. The proposed method remains competitive and it yields new results.

1 Introduction

In this paper we investigate the use of the truncated tree search for solving the so-called *3-Dimensional Sphere Packing Problem* (noted 3DSPP). An instance of 3DSPP is defined by a set N of n spheres/items and an object/container $\mathscr{P}$ of fixed width W and height H and, unlimited length (noted L for the rest of the paper). Moreover, each $i \in N$ is characterized by its radius r_i and the aim of the problem is to optimize the length L of the object $\mathscr{P}$ such that all items of N are packed in the target object, without overlapping.

M. Hifi (✉) · L. Yousef
EPROAD - EA 4669, Université de Picardie Jules Verne, 7 rue du Moulin Neuf,
80000 Amiens, France
e-mail: hifi@u-picardie.fr

L. Yousef
e-mail: labib.yousef@edut.u-picardie.fr

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_9

Formally, 3DSPP can be stated as follows:

$$\text{Minimize} \quad L \tag{1}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \geq (r_i + r_j)^2, \quad \forall (i, j) \in N^2,\ i < j \tag{2}$$

$$r_i \leq x_i \leq L - r_i,\ \forall i \in N \tag{3}$$

$$r_i \leq y_i \leq H - r_i,\ \forall i \in N \tag{4}$$

$$r_i \leq z_i \leq W - r_i,\ \forall i \in N \tag{5}$$

$$\underline{L} \leq L \leq \overline{L} \tag{6}$$

$$(x_i, y_i, z_i) \in \mathbb{R}^3_+,\ \forall i \in N, \tag{7}$$

where the objective function (1) minimizes the length of the container $\mathscr{P}$ containing all the items, Eq. (2) ensures the non-overlap constraint of any pair of distinct items (i, j) of $N \times N$ and Eqs. (3)–(5) ensure that all items of N belong to the target container $\mathscr{P}$ of dimensions (L, W, H). Finally, Eq. (7) ensure that all items are placed in the container $\mathscr{P}$. Note that since the goal of the problem is to find the smallest length of the container containing all items of N, then it is easy to start any method trying to solve the above problem by a trivial solution value representing the sum of the spheres' area affected to $\underline{L}$ (Eq. (6)) and a quick solution value, reached by a greedy algorithm, can be affected to $\overline{L}$.

The rest of the paper is organized as follows. The literature review on the 3DSPP and some of its variants is given in Sect. 2. The presentation and the model used for the problem is described in Sect. 3.1. Section 3.2 describes the subset of eligible positions associated to each selected item. Section 3.3 details the tree search that uses a hill-climbing strategy for approximately solving an instance of 3DSPP. Indeed, given a current container, the tree search is applied in order to create a set of eligible nodes to evaluate and the hill-climbing strategy mimics the global search where all eligible positions are evaluated following an evaluation operator. In this case, only the best nodes may be chosen for further branchings and so, trying to improve the quality of the solutions. The previous search is iterated until reaching the minimum length of the target container. The performance of the proposed algorithm is evaluated in Sect. 4, where its obtained results are compared to those reached by recent algorithms published in the literature. Finally, Sect. 5 summarizes the contribution of the paper.

2 Background

Cutting and Packing (CP) is considered as a natural combinatorial optimization problems (cf., Wascher et al. [18]), where its problematics are admitted in several real-world applications, like logistics, manufacturing, production process, automated planning, etc. The 3DSPP belongs to the CP family and one of the more recent paper addressing an optimization of a set of unequal spheres, is due to Sutou and Dai [17]; that has been used for tackling an application of the automated radiosurgical treatment planning.

Few papers addressing the 3DSPP are available in the literature. Indeed, among these papers, we can cite Lochmann et al. [12] who proposed a statistical analysis for packing random spheres with variable radius distribution. Li and Ji [11] investigated the use of dynamics-based collective approach in order to study the stability and convergence of their approach when tackling the cylinder container packing. The random close packing fractions of lognormal distributions of hard spheres has been studied by Farr [3], where a one-directional approach was proposed to predict a close packing of spheres of lognormal distributions of sphere sizes.

Packing spheres into a container has been considered by Sutou and Dai [17] who addressed a global optimization method. Stoyan et al. [16] designed a mathematical model for packing spheres into an open container, where a neighborhood search was considered for providing extremum points and a series of approximate solutions. In M'Hallah et al. [13], VNS and a nonlinear programming solver were combined for solving the sphere packing problem into a container. Such an approach can be viewed as a straightforward of Hifi and M'Hallah's [5] approach, where the last method can be applied to any variant of the two and three dimensional packing problems. By using the same principle, Alkandari and M'Hallah [14] adapted the same principle as in [13] for solving the problem of packing identical spheres into a cube.

In Soontrapa and Chen [15], the problem of packing identical spheres into a smallest containing sphere has been also tackled: a random search by using Monte Carlo's method has been investigated. Birgin and Sobral [2] proposed twice-differentiable non-linear programming models for 2D and 3D (circles and spheres) packing where the container may be circular, rectangular, etc. Such a model has been solved by applying ALGENCAN solver in order to create multiple starts solutions. Hifi and Yousef [7] investigated a hybrid heuristic for solving the 3-dimensional sphere packing problem. That approach is based upon combining a greedy selection phase, a width-beam search and a dichotomous search. Later, a simple version of Hifi and Yousef's [7] approach has been presented in Akeb [1] and according to its experimental part, the reader can easily show the disadvantage of this approach when compared to the performance of a simple version of Hifi and Yousef's [7] approach. Finally, for the 3DSPP, a modified and improved version of Hifi and Yousef's [7] method has been proposed in Hifi and Yousef's [8], where the chosen nodes were selected following the width-beam search combined with hill-climbing. Extensive efficient models and methods for packing both circular and sphere problems were reviewed in Hifi and M'Hallah [4].

In this paper, we propose an extended version of the truncated tree search-based heuristic for the 3-dimensional sphere packing problem (cf., Hifi and Yousef [8]). The designed method is based on combining two main futures:

(i) A greedy search-based procedure, which is used for creating a series of partial solutions: each partial solution characterizes a subset of items already packed into the current container.
(ii) A hill-climbing strategy that is used in order to simulate a lower bound for the rest of non packed items.

The aforementioned steps are repeated on a tree search-based heuristic which mimics the well-known branch-and-bound procedure.

3 A Tree Search-Based Algorithm for 3DSPP

This section begins by exposing the problem representation (Sect. 3.1). The branches, characterizing the nodes of the tree, are generated following a basic procedure as described in Sect. 3.2. Finally, Sect. 3.3 exposes the principle of the proposed algorithm and its main steps.

3.1 3DSPP's Representation

Because of the huge number of possible positions when an item will be assigned to the target object, namely $\mathscr{P}$, we propose to generate a subset of eligible positions by using a Greedy Procedure (GP) (cf. Hifi and Yousef [7]). Besides to the approach used in [7], herein, the generated positions are used for creating eligible branches according to a selected node (as described in Hifi and Yousef [8]). GP is then used as an operator for evaluating the potential of a node (position) to investigate. Therefore, GP is used either for providing a complete solution or for estimating the gap between the node's upper bound and the best length reached up to now.

Generating eligible nodes for the developed tree needs the representation of the object and each item. We use the following representation:

- We assume that the bottom-left-depth corner of the target object $\mathscr{P}$ is positioned at the origine position $(0, 0, 0)$, where $\mathscr{P}$ is characterized by a finite set $\mathbb{F}$ of faces such that $\mathbb{F} = \{\text{left, top, right, bottom, depth, front}\}$ (Fig. 1 illustrates the representation of $\mathscr{P}$).

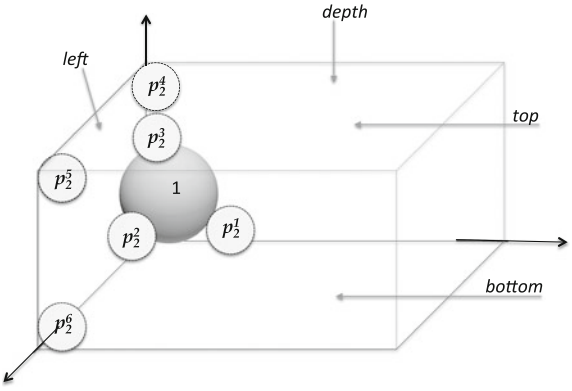


Fig. 1 Illustration of the target object $\mathscr{P}$ and the mechanism generating valid nodes

Table 1 The distance between both item i and a face f

f	$\delta_{i,f} \mid i \in N,\ f \in \mathbb{F}$
Left	$x_i - r_i$
Bottom	$y_i - r_i$
Depth	$z_i - r_i$
Right	$L - x_i - r_i$
Top	$H - y_i - r_i$
Front	$W - z_i - r_i$

- Each item $i \in N$ is centered at the position (x_i, y_i, z_i).
- A pair (i, j) of N are represented by their distance $\delta_{i,j}$:

$$\delta_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} - (r_i + r_j), \quad \forall (i, j) \in N^2, \quad (8)$$

where both i and j can be positioned when $\delta_{i,j} \geq 0$.

Furthermore, positioning $i \in N$ into the target object $\mathscr{P}$ needs to define a distance between i and the faces of $\mathbb{F}$. Indeed, for all faces of $\mathscr{P}$, Table 1 shows each corresponding distance to be satisfied if $i \in$ is assigned to an eligible position of $\mathscr{P}$.

3.2 Nodes of the Tree

Generating nodes of the tree is equivalent to create a subset of eligible positions in the target object $\mathscr{P}$. We do it by applying a Greedy Procedure (GP) that searches for the minimum distance between already packed items and the target object $\mathscr{P}$.

3.2.1 Favorable Nodes and Paths

In this section, we describe some positions that represent the favorable nodes for the developed tree. These nodes induce also the paths which serves to build the best paths for the developed tree. Indeed, let $i \in N$ be the next item chosen to be packed in the target object $\mathscr{P}$ and P_{I_i} denote the subset of its eligible positions. Assume that the first item of N is positioned at (r_1, r_1, r_1) and for $i \in N,\ i \geq 2$, suppose that:

- I_i is the set of items of N already positioned in the current object $\mathscr{P}$;
- $\overline{I_i}$ denotes the set of items of N which are not yet packed in $\mathscr{P}$ and,
- P_{I_i} represents the set of distinct eligible positions (nodes) for the next item i to pack given the set of packed items I_i.

It follows that a favorable point is associated to an eligible position $p_{i+1} \in P_{I_i}$ (for the item i), which can be determined according to 3 elements: $e_1,\ e_2$ and e_3, respectively. An element is either an item belonging to N already positioned or one

of the six faces of $\mathbb{F}$. Let $\mathscr{T}_{p_{i+1}}$ be the set of favorable nodes according to the three elements e_1, e_2 and e_3.

Then, a step of GP, for $i \geq 2$, $i \in N$, proceeds as follows: the distance $\Delta(p^k_{i+1})$ corresponding to the $(i+1)$th item to pack, when positioned at the eligible position $p^k_{i+1} \in P_{I_i}$, induces the distance:

$$\Delta(p^k_{i+1}) = \min_{j \in I_i \cup \mathbb{F} \setminus \mathscr{T}_{p^k_{i+1}}} \delta_{(i+1,j)}. \tag{9}$$

3.2.2 Comments

Note that GP starts by positioning the first item $i = 1$ at the bottom-left-depth position (i.e., (r_1, r_1, r_1)), while the remaining $n-1$ items are successively positioned depending on the value of Eq. (9). Of course, we always favor

(i) the first item according to the order initially fixed or
(ii) that realizes the smallest distance to the left side of $\mathscr{P}$.

Moreover, one can use GP as a greedy procedure for searching a feasible solution to 3DSPP. Indeed, by setting the starting length of the target object $\mathscr{P}$ to ∞ (i.e., $L_{\mathscr{P}} = \infty$) and by applying successively GP (by excluding the "right" face from $\mathbb{F}$) on the rest of the non packed items, GP terminates by building a feasible solution with length L_U. The provided length L_U can also be considered as an upper bound for starting the proposed algorithm.

3.3 A Tree Search-Based Algorithm

It is well-known that using a tree search requires:

1. Defining the nodes of the tree with their characteristics and,
2. The branching mechanism out of the nodes that serves to generate successors.

3.3.1 Valid Nodes

The set of *eligible positions* forms the *valid nodes* of the developed tree. As mentioned in Sect. 3.2, let η_i be the current selected node to develop; that is defined by a pair of subsets (i) the first subset I_i that contains all items assigned to the target object $\mathscr{P}$ and (ii) the complementary subset $\overline{I}_i$ that includes the unassigned items. A *partial solution* derives from I_i when all the items are positioned in $\mathscr{P}$ and a *complementary solution* may be constructed according to $\overline{I}_i$. The complementary solution can be reached by applying GP; that is used as a greedy solution procedure (as discussed in Sect. 3.2).

It follows that, for the selected node η_i,

1. Its successors correspond to eligible positions which are those created according to all the positions of the set P_{I_i};
2. Among all successors of P_{I_i}, only the best positions are chosen following an evaluation operator (as discussed in Sect. 3.4).

Hence, branching out of a valid node η_i is equivalent to create at most $|P_{I_i}|$ branches emanating out of the current node. Each resulting node corresponds to packing the subset of items I_i and assigning to the current item i a favorite eligible position. Moreover, each of these created nodes will be represented by a pair of two complementary subsets of items of N.

3.3.2 Branching

The assessment of the potential of each valid node (in leading to the better solution) is a critical step in enumerative search. It is equivalent to construct a lower bound to the optimal solution. Herein, it should be based on a simple and quick evaluation operator as it will be applied to each generated node. Yet, it has to be accurate by providing each node η with a potential value that is close to the objective function value of any leaf node emerging from η. Note that evaluation operators can be classified as *priority* or *total-cost*:

1. The *priority operator* is a local or greedy strategy that focuses on the next decision to be made.
2. The *total-cost operator* has a more global view: it estimates the potential of the current node η by estimating the objective function value of a complete solution that is built from the current partial solution of η.

Algorithm 1 describes the framework of the standard tree search applied for a minimization problem. A node corresponds to a partial feasible solution and the set *Open* of current nodes is initialized to the root node, namely η_0. Each node η taken from *Open* generates a set of offspring nodes, and stores them into a temporary list, namely B_η. If a node γ of B_η is a leaf (i.e., no further branching is possible out of γ), then its objective function value z_γ is computed and compared to $z^\star$ (the best objective value obtained up to now). If $z_\gamma < z^\star$, then the incumbent solution is set to the leaf node; $z^\star$ is then updated: $z^\star = z_\gamma$; and γ is removed from B_γ (the nodes emanating from the selected node γ). Further, all nodes belonging to B_η are transferred to *Open* for further branchings. This process is iterated until no further branching is possible, i.e., until $Open = \emptyset$.

Algorithm 1. A standard Tree Search-Based Algorithm (TSBA)

Input. An instance of a minimization problem.
Output. An optimal solution η^* with objective value z^*.

1: **Initialization Step.**
2: Set $Open = \{\eta_0\}$, where $Open$ is the set of nodes to be investigated and η_0 is the root node.
3: If an initial feasible solution $\eta^\star$ is available, set $z^\star$ to its objective function value; otherwise, set $z^\star = +\infty$.
4: **Iterative Step.**
5: **while** ($Open \neq \emptyset$) **do**
6: Choose a node $\eta \in Open$; branch out η; remove η from $Open$ and insert the created nodes into B_η.
7: **if** (a node γ of B_η is a leaf) **then**
8: compute its objective function value z_γ;
9: **if** $z_\gamma < z^\star$ **then**
10: update $z^\star$ and the incumbent solution $\eta^\star$;
11: **end if**
12: remove γ from B_η.
13: **end if**
14: Assess the potential of each node of B_γ using an evaluation operator.
15: Insert all nodes of B_γ into $Open$.
16: **end while**

3.4 Adaptation of TSBA to 3DSPP

In our study, we introduce a Hill-Climbing (HC) strategy that is used for avoiding exhaustive search by performing a partial enumeration of the solution space. As a result, a truncated TSBA which may be equivalent to the well-known beam search (Hifi et al. [6] and Yavuza [19])), where only a subset of paths are taken for further branchings and the other nodes are discarded. A each step of the search procedure, a node η is selected and after evaluating all its successors, only the best ω nodes are chosen for further branchings. Of course, as described above, each selected node is assessed via its evaluation function whose role is to provide a promising separation mechanism of the nodes.

The proposed HC considers a *hybrid* operator that can be viewed as an alternative to both *priority* and *total-cost* operators (cf. Sect. 3.3.2). Indeed, almost of the priority operator (that explores a local information) nor the total-cost operator (because estimating a "good" lower bound for 3DSPP remains difficult), we propose to combine these two operators as follows:

(i) Using an approximation for the *total-cost*: one can complete the local solution (the already positioned items of I_η) with a lower bound estimating the best solution that contains all items of $\overline{I_\eta}$.
(ii) Completing the *local solution*: one can use a quick greedy local search-based procedure, like the procedure GP (cf. Sect. 3.2) when the generated node is chosen for further branchings.

Algorithm 2. Truncated Tree Search-Based Heuristic for 3DSPP (TTSBH)

Input. A set of items I and a predefined length l_{best}.
Output. `feasible`..

1: **Initialization Step.**
2: Let ω and ε be two predefined values.
3: Set $Open = B_0$, where B_0 denotes the starting eligible nodes according to the first packed item $i = 1$.
4: Set $\ell = 1$ and $B_\ell = \emptyset$.
5: Set the variable `feasible` to `false` /* no feasible solution at hand */
6: **Iterative Step.**
7: **while** $\Big((Open \neq \emptyset)$ and (the runtime limit is not performed) and $(\ell < n)\Big)$ **do**
8: Choose η from *Open*;
9: Let $B_\eta = \{\gamma_1, \ldots, \gamma_{|P_{I_\eta}|}\}$ be the successors of η.
10: Evaluate the potential of each node γ belonging to B_η by computing $g(\gamma)$ and $h'(\gamma)$.
11: For each $\gamma \in B_\eta$ apply ISBH(γ, $L^\star$) and update $L^\star$ if necessary with the incumbent solution.
12: Filter B_η by keeping the ω best nodes realizing the smallest values of $L^\star/\big(g(\gamma) + h'(\gamma)\big)$.
13: Replace all the nodes of *Open* by those of B_η, reduce B_η to empty and increment ℓ.
14: **end while**

3.4.1 A Truncated Tree Search-Based Heuristic

Algorithm 2 describes an adaptation of truncated tree search-based heuristic (noted TTSBH). We recall that a node corresponds to a partial solution and the set *Open* of current nodes contains initially the starting nodes of the root node B_0 whereas B_η containing the offspring nodes is initialized to the empty set.

On the one hand, a selected node η taken from *Open* (step 7), whose evaluation is z_η, creates a subset of nodes $B_\eta = \{\gamma_1, \ldots, \gamma_{|P_{I_\eta}|}\}$, where each resulting node is evaluated according to its global-cost operator; that is,

$$z_\eta = g(\eta) + h(\eta).$$

On the other hand, because $|P_{I_\eta}|$ may be large, then only a subset of nodes is chosen for further branching. Indeed (line 9), if a node γ of B_η packs at most $n - 1$ items, then it remains in B_η whenever $z'(\gamma) < z^\star$, where

$$z'(\gamma) = g(\eta) + h'(\eta) \tag{10}$$

with $h'(\eta) = (1 + \varepsilon)h(\eta)$ and ε is considered as a small predefined value that is used for making a correction on the complementary lower bound $h(\eta)$.

Whenever Eq. (10) is not satisfied, then γ is removed from B_η. Further, since we try to intensify the search that permits to improve the quality of the solution, we apply GP on all retuned nodes (line 10). Then, $L^\star$ is updated whenever GP realizes a better length; in this case, its corresponding incumbent solution is also updated. The rest of the nodes belonging to B_η (line 11) are reordered in nondecreasing order of their estimated lower bounds $z'(\gamma)$ and only the best ω nodes are selected and transferred

to *Open* for further branchings. This process is iterated until no further branching is possible, i.e., until $Open = \emptyset$, or when the fixed runtime limit is performed.

Note also that, at lines 9 and 10, if a node γ of B_η is a leaf (i.e., no further branching is possible out of γ), then its objective function value z_γ is computed and compared to the best solution value $z^\star$ obtained up to now. If $z_\gamma < z^\star$, then the incumbent solution is set to a leaf node; $z^\star$ is then updated: $z^\star = z_\gamma$; and γ is removed from B_η.

3.4.2 Bounding the Search with an Iterative Process

It was already mentioned in Sect. 3.4 (see Step 10 of Algorithm 2) that TTSBH uses an extensive search. Indeed, we introduce an Iterative TTSBH (noted ITTSBH) in order to improve the quality of solutions reached. Herein, the principle of such a procedure is explained.

Let $(\overline{L}, W, H)$ be the current object $\mathscr{P}$ and η be the node chosen for branching. The length $\overline{L}$ denotes the best length reached by TSBH at a certain time. Now suppose that instead of packing the n items in $(\overline{L}, W, H)$, we propose to perform the search on a series of objects of dimensions (L_k, W, H), with $\underline{L} \leq L_k \leq \overline{L}$. In this case, one can use a starting interval $[\underline{L}, \overline{L}]$, where $\underline{L}$ denotes a lower bound for the 3DSPP and $\overline{L}$ its upper bound and so, for each target object (L_k, W, H), TSBH attempts to pack all the n items into (L_k, W, H), where $\underline{L} \leq L_k \leq \overline{L}$.

Algorithm 3 . An Iterative Truncated Tree Search-Based Heuristic: ITTSB

Input. A node η and the best length $L^\star$.
Output. `feasible`
/* with a new length $L^\star$ and the coordinates of all items of N when `feasible = true` */

1: **Initialization step**
2: Set $\underline{L} \leftarrow \frac{4\pi}{3 \times W \times H} \sum_{i \in N} (r_i^3)$ and $\overline{L} = L^\star$.
3: **Iterative step**
4: **while** (the runtime limit is not performed) **do**
5: **repeat**
6: $L' = (\overline{L} + \underline{L})/2$
7: Set `feasible` = TTBH(η)
8: **if** `feasible` **then**
9: `set` $\overline{L} = L'$, $L^\star = L'$ `and return` $L^\star$; $\underline{L} = L'$ `otherwise`
10: **end if**
11: **until** $(\overline{L} - \underline{L} \geq \alpha)$
12: **end while**

Algorithm 3 summarizes the principle of such a process which can be viewed as an intensification procedure for a given node η. Indeed, it begins (line 2) by defining the starting interval $[\underline{L}, \overline{L}]$, where $\overline{L}$ is setting equal to the best length $L^\star$ reached so far. The internal loop (lines 5–11) tries to pack all items in the target object by using TSBH. If a feasible solution is obtained, then the incumbent solution is stored with

its best length ($L^\star = \overline{L}$) and ITTBH stops and returns this solution. In the case that IBSP doesn't pack the n items, the process is iterated till the gap between both lower and upper bounds (of the current interval) becomes closest to a certain tolerance, namely α (that is fixed to 0.1 in our experimental study). Finally, the aforementioned process can also be stopped when the fixed runtime limit is exceeded.

4 Experimental Part

The performance of the proposed iterative truncated tree search-based heuristic that uses hill-climbing strategy (noted ITTSBH) was evaluated on three sets of benchmark instances: Set1, Set2 and Set3. Set1 contains six instances (SYS1, ..., SYS6) taken from Stoyan et al. [16], where the number of items varies from 25 to 60. These instances have been used as benchmarks in Stoyan et al. [16], Birgin and Sobral [2] and Kubach et al. [10]. Set2 contains six instances (KBTG1, KBTG2, KBTG3, KBTG7, KBTG8, and KBTG9) taken from Kubach et al. [10]. For each instance, both dimensions W and H of the object are fixed to 10 whereas the number of items is equal to 30 (resp. 50) for the first (resp. last) three instances. Further, these six instances have been used as benchmarks in Kubach et al. [10] where they represent the six instances with unequal spheres. Set3 contains 22 instances that are obtain ed by combining the six instances of Stoyan et al. [16].

The rest of this part is organized as follows. First, the behavior of ITTSBH is evaluated on the first set of instances Set1 (Sect. 4.2). The results reached by ITTSBH are thereafter compared to those reached by five heuristics: Stoyan et al.'s [16] method-based heuristic (noted SYS), Birgin and Sobral's [2] algorithm (noted BSA), both sequential and parallel heuristics proposed by Kubach et al. [9, 10] (noted KBTG_s and KBTG_p respectively) and Hifi and Yousef's [7] algorithm (noted HY) that uses a standard width-beam search. Second, for the instances of Set2 (Sect. 4.3), the results provided by TSBH are compared to those reached by Kubach et al. [10] and Hifi and Yousef [7]. Third and last, the instances of Set3 are tested by comparing the results reached by both HY and ITTSBH. Note also that the proposed algorithm was coded in C++ and tested on an Intel Core 2 Duo (2.53 Ghz and with 4 Gb of RAM) and the runtime limit was fixed to one hour.

4.1 Behavior of ITTSBH on the Instance SYS5

ITTSBH uses three parameters: ω, ε and the maximum runtime limit to fix, noted t_{max}. Our computational study was conducted by varying ω in the discrete interval $\{1, 2, \ldots\}$ with a maximum runtime limit $t_{max} = 3600$ s when considering Set1 and Set2 (which can be considered as a standard runtime limit considered by algorithms of the literature—cf., Sects. 4.2 and 4.3). Finally, a new set (noted Set3) containing

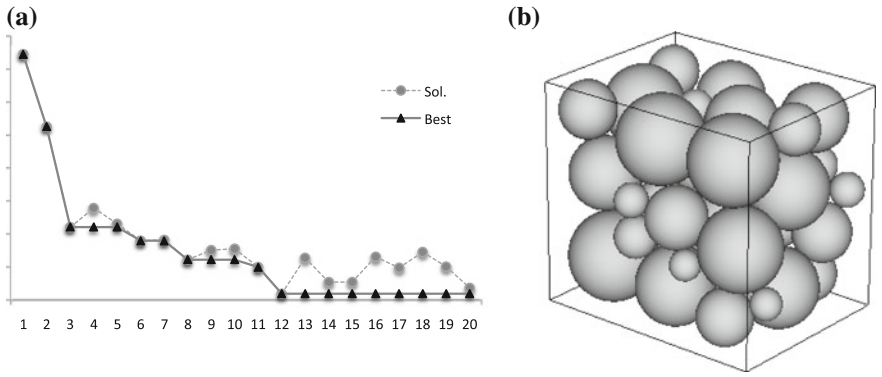


Fig. 2 Illustration of HY's behavior on SYS5 instance: **a** variation of solution's quality and **b** the best solution's structure reached by HY

22 new large-scale instances is tested in Sect. 4.4, where two different runtime limits are considered.

In order to show the effect of these parameters, we show the quality of the solutions obtained by ITTSBH when the value ω increases. In this case, the solution values are compared to those obtained by HY algorithm (that realizing the best solution available in the literature) by fixing the same runtime. We first do it on the instance SYS5 considered by Stoyan et al. [16]: we recall that HY uses a width-beam search that varies ω from 1 to the higher value (incrementing ω with one unit) corresponding to the fixed runtime limit t_{max}. The same process is applied by ITTSBH that combines hill-climbing and an estimated lower bound.

Figure 2a shows the behavior of HY when varying the width ω for the instance SYS5. From the curve labeled Sol., one can observe that the value of the length $L^{\star}$ oscillates by generating a series of local optima. One can observe that in some cases HY provides poor result for some higher values of ω.

4.2 Behavior of ITTSBH Versus Six Heuristics Available in the Literature (Set1)

Second, for the instances of Set1, Table 2 shows the results obtained by ITTSBH and those reached by SYS (Stoyan et al. [16]), BSA (Birgin and Sobral [2]), KBTG$_s$ (Kubach et al. [10]), its parallel version KBTG$_s$ (proposed in Kubach et al. [9]), LF2 (Akeb [1]) and HY (Hifi and Yousef [7]), where all the reported solutions are taken from [1, 2, 7, 10, 16].

Column 1 displays the instance's label. Column 2 reports the objective value $L^{\star}_{\text{SYS}}$ reached by SYS algorithm whereas column 3 displays BSAs' objective values (noted $L^{\star}_{\text{BSA}}$). Columns 4 and 5 report the best objective values (noted $L^{\star}_{\text{KBTG}_s}$ and

Table 2 Behavior of ITTSBH on instances of Set1

Label	$L^{\star}_{\text{SYS}}$	$L^{\star}_{\text{BSA}}$	$L^{\star}_{\text{KBTG}_s}$	$L^{\star}_{\text{KBTG}_p}$	$L^{\star}_{LF2}$	$L^{\star}_{HY}$	$L^{\star}$	ω^*
SYS1	9.912	9.7942	9.2874	9.2656	9.2234	9.2431	**9.1796**$^{\diamond}$	26
SYS2	9.623	–	9.1280	8.9301	9.1138	8.9164	**8.8922**$^{\diamond}$	29
SYS3	9.473	9.3090	8.9850	8.7178	8.9316	8.7055	**8.6702**$^{\diamond}$	31
SYS4	11.086	11.0962	10.8760	10.4042	10.7653	10.2357	**10.2012**$^{\diamond}$	36
SYS5	11.646	11.6211	11.3494	10.9865	11.1948	10.9359	**10.8954**$^{\diamond}$	34
SYS6	12.842	12.7215	12.3745	11.8399	12.2519	11.8178	**11.7943**$^{\diamond}$	16
Average	16.764	10.908	10.333	10.024	10.247	9.976	**9.939**	

The symbol "–" (resp. "$\diamond$") means that the value for this instance is not available (resp. corresponds to the best solution obtained by the corresponding algorithm)

$L^{\star}_{\text{KBTG}_p}$) realized by both the sequential KBTG algorithm and its parallel version (without fixing the runtime limit as considered in Kubach et al. [10]). Column 6 shows LF2's objective value (noted L_{LF2}), column 7 reports HY's objective value (noted L_{HY}) whereas column 8 displays the objective value (noted $L^{\star}$) realized by ITTSBH. Finally, column 9 reports the best value of ω for which ITTSBH's objective value is realized.

The percentage improvement (when it happens) yielded by ITTSBH according to the results displayed in Table 2 are reported in Table 3. The first part contains the fourth columns displaying the instance's information and the second part contains the variation of the percentage improvement realized by ITTSBH over the other tested algorithms (noted %SYS, %BSA, %KBTG$_s$, %KBTG$_p$ and %HY, respectively). The reported values correspond to the results reached by all algorithms when fixing the runtime limit fixed to one hour.

Table 3 Variation of the percentage improvements between all tested algorithms: ITTSBH, HY, SYS, BSA, LF2 and both KBTG$_s$ and KBTG$_p$ on instances of Set1

#Inst.				ITTSBH versus all methods (% Improvement)					
Label	n	H	W	%SYS	%BSA	%KBTG$_s$	%KBTG$_p$	%LF2	%HY
SYS1	25	5.5	6.9	7.24	6.70	1.17	0.94	0.48	0.69
SYS2	35	6.5	7.9	7.92	–	2.65	0.43	2.49	0.27
SYS3	40	5.5	6.9	8.82	7.37	3.63	0.55	3.01	0.41
SYS4	45	8.5	9.9	8.31	8.77	6.61	1.99	5.53	0.34
SYS5	50	8.5	9.9	6.49	6.66	4.17	0.84	2.75	0.37
SYS6	60	8.5	9.9	8.67	7.86	4.92	0.39	3.88	0.20
Average				7.91	7.47	3.86	0.85	3.02	0.38

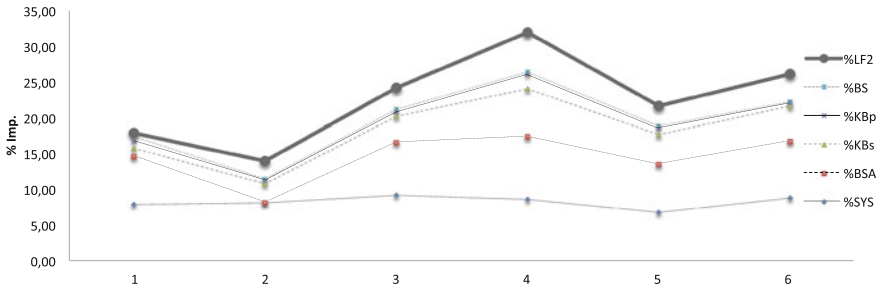


Fig. 3 Variation of the percentage improvement realized by ITTSBH on the instances of Set1

From both Tables 2 and 3, one can observe what follows:

1. First, ITTSBH outperforms SYS, BSA, KBTG$_s$ and LF2 since it is able to reach most of the best solutions for all instances of the first group Set1.
2. Second, the percentage of the improvement varies from 6.49 % (instance SYS5) to 8.82 % (instance SYS3) when comparing ITTSBHs' results to SYSs' ones.
3. Third, the percentage improvement remains interesting when comparing ITTSBHs' results to those reached by BSA: the improvement varies from 6.66 % (instance SYS5) and 8.77 % (instance SYS4).
4. Fourth and last, ITTSBHs' results remain better than those reached by the parallel algorithm KBTG$_p$ (we recall that the parallel algorithm ran without runtime limit) and the recent simple adaptation of the beam search considered in [1]. Indeed, all the solution values for the instances of Set1 are improved.

   a. On the one hand, ITTSBHs' results when compared to those of KBTG$_p$, the percentage of improvement varies from 0.43 % (instance SYS2) to 1.99 % (instance SYS4). which remains very interesting for a sequential algorithm.
   b. On the other hand, the improvement becomes more interesting when comparing ITTSBHs' results to those reached by LF2, as shown in Table 3).

Figure 3 shows the behavior of ITTSBH on the six instances of Set1: each curve represents the variation of the improvements (when it happens) when compared to SYS, BSA, KBTG$_s$, KBTG$_p$, LF2 and HY, respectively. Figure 4a displays the structure of the final solution reached by ITTSBH for the instance SYS1 whereas Fig. 4b shows the final configuration obtained by ITTSBH.

4.3 Performance of DSBH Versus KBTG, HY and LF2 Heuristics (Set2)

This section compares the results obtained by ITTSBH to those realized by both KBTG$_s$ and HY on instances of Set2 extracted from Kubach et al. [10] (since it is the only existing algorithms that have tested instances of Set2). For this type of

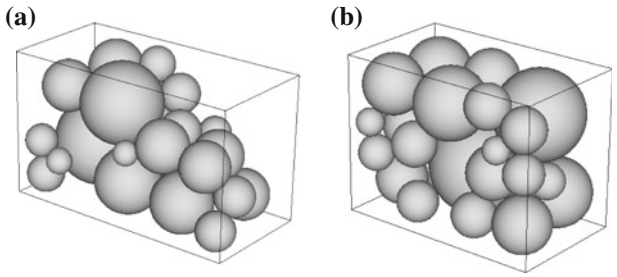


Fig. 4 Illustration of ITTSBH's behavior **a** on the instance SYS1 (realizing the smallest percentage improvement for the value $L^{\star} = 11.7943$) and **b** on the instance SYS1 (realizing the greatest percentage improvement for the value $L^{\star} = 9.1796$)

Table 4 Performance of ITTSBH versus KBTG$_s$, LF2 and HY algorithms on instances of Set2

#Inst.	$KBTG_s$	LF2		HY		ITTSBH	
Label	$d^{\star}_{KBTG_s}$	$d^{\star}_{LF2}$	$L^{\star}_{LF2}$	$d^{\star}_{HY}$	$L^{\star}_{HY}$	$L^{\star}_{ITTSBH}$	ω^{*}
KBTG1	54.096	54.494	11.2063	56.0092	10.9031	**10.8076**	23
KBTG2	*30.071*	30.071	*1.99*	30.071	*1.99*	*1.99*	23
KBTG3	51.387	51.693	18.9231	53.6243	18.2415	**18.1936**	24
KBTG7	55.372	55.824	13.5075	57.5662	13.0997	**12.9653**	14
KBTG8	45.060	46.639	2.6027	47.004	2.5825	**2.582**	13
KBTG9	52.732	52.732	29.7023	55.3203	27.8033	**27.7152**	26

instances, instead of determining the minimum length $L^{\star}$ of the target container $\mathscr{P}$, Kubach et al. [10] determined the density of all packed items in the final object $\mathscr{P}$. Therefore, in addition to the density, we report in the same table the best objective value $L^{\star}$, representing the best length of the final container $\mathscr{P}$, corresponding to that density.

Table 4 shows the results produced by the four methods: KBTG$_s$, LF2, HY and ITTSBH, respectively. Column 1 tallies the instance label and column 2 shows KBTG$_s$'s solution value (expressed in term of density; that is, the value extracted from Kubach et al. [9, 10]). Columns 3 and 4 display both the objective value $L^{\star}$ reached by LF2 and its corresponding density $d^{\star}$ (extracted from [1]) and columns from 5 to 6 tally HY's objective value and its corresponding density (extracted from Hifi and Yousef [7]). Finally, the last columns (7 and 8, respectively) report the best objective value realized by ITTSBH and the optimal value of ω for which the solution value is reached. From Table 4, one can observe that:

1. First, ITTSBH performs better than all approaches since it improves most solutions reached by KBTG$_s$, LF2 and HY, respectively. Indeed, five out of the six best solution values available in the literature are improved by ITTSBH while it matches the other solution (instance KBTG2).

2. Second, for the improved solutions, ITTSBH realizes an improvement gap varying from 0.0194 % (instance KBTG8) to 1.0366 % (instance KBTG7). Globally, the average improvement over all instances is equal to 0.4201 %.
3. Third, the improvement gap realized by ITTSBH when compared to LF2 solution values varies from 0.8017 % (KBTG8) to 7.1697 % (KBTG9). It realizes an average percentage gap of 3.3087 %, which can be considered as an impressive improvement for such a problematic.
4. Fourth and last, ITTSBH has a better behavior when comparing its results to those reached by $KBTG_s$ algorithm which applies the same strategy as used in Akeb [1].

4.4 Performance of ITTSBH Versus HY on a New Set of Instances (Set3)

In this section, we analysis the behavior of ITTSBH on some large-scale instances by varying its runtime limit. We do it by generating a set containing 22 new instances (noted Set3), where each instance is obtained by combining each paire of instances belonging to the first set (noted Set1—cf., Sect. 4.2). Table 5 shows the characteristics of these instances: for example, the instance SYS.1.3.a is obtained by adding all items of both instances SYS1 and SYS3 and the dimension W (resp. H) is taken in the intervalle {5.5; 6.5; 7; 8.5; 10} (representing all the dimensions of instances of Set1). All other instances are created following the same scheme, where the last instances have the largest number of items to pack.

Moreover, because the codes associated to SYS, $KBTG_s$, $KBTG_p$ and LF2 are not available, we then compared ITTSBHs' solution values to those reached by HY, where the same variation on the runtime limit is considered. Indeed, herein two runtime limits (for both algorithms) are considered: $t_1 = 3600$ and $t_2 = 7200$ (measured in seconds).

From Table 5, one can observe what follows.

1. The average value of 22.9845 confirms the superiority of ITTSBH when t_1 is considered as the runtime limit for both tested algorithms. In this case, ITTSBH is able to improve 12 solution values out of 22 (that represents a percentage of 54.55 %), it matches two solution values and it fails in eight occasions to provide better solutions.
2. By doubling the runtime (using t_2 as the limit for both algorithms), the behavior of ITTSBH becomes more interesting. Indeed, in this case, the proposed algorithm realizes an average solution value of 22.7952 (although HY algorithm realizes interesting average). In this case, ITTSBH improves 21 out of 22 instances (95.45 % of the tested instances) and it matches the other solution (for the instance SYS.1.6.b).

One can notice that ITTSBH can provide better solutions when increasing the runtime. This phenomenon can be explained by the fact that the diversification

Table 5 Behavior of both HY and ITTSBH on the large-scale instances (Set3)

#Inst.				The first runtime limit t_1				The second runtime limit t_2			
Label	n	W	H	HY		ITTSBH		HY		ITTSBH	
				L_{HY}	$\omega^\star$	$L^\star$	$\omega^\star$	L_{HY}	$\omega^\star$	$L^\star$	$\omega^\star$
SYS.1.3.a	65	5.5	6.9	18.0193	11	18.0180	12	17.8540	13	17.6599	17
SYS.1.3.b	65	5.5	9.9	17.9288	9	17.9692	10	17.8451	11	17.8404	17
SYS.1.4.a	70	5.5	6.9	34.0449	11	33.7273	16	33.8374	13	33.7266	17
SYS.1.4.b	70	8.5	9.9	14.5082	11	14.4873	14	14.5042	14	14.2626	15
SYS.1.5.a	75	5.5	6.9	34.8934	9	35.1003	14	34.8990	11	34.9285	13
SYS.1.5.b	75	8.5	9.9	15.0323	11	14.9848	14	14.9718	13	14.7812	15
SYS.1.6.a	85	5.5	6.9	37.7972	7	37.5275	12	37.3887	9	37.3751	13
SYS.1.6.b	85	8.5	9.9	16.2319	9	15.9455	15	15.9117	11	15.9117	18
SYS.2.3.a	75	6.5	7.9	15.4148	9	15.3529	14	15.3475	12	15.0985	16
SYS.2.3.b	75	5.5	6.9	21.2641	7	21.2641	12	21.2032	11	21.1228	14
SYS.2.4.a	80	5.5	7.9	26.6540	6	26.7174	14	26.7174	9	26.4935	16
SYS.2.4.b	80	8.5	9.9	15.9796	8	15.9347	14	15.9125	9	15.6993	15
SYS.2.5.a	85	6.5	7.9	27.5623	7	27.2781	12	27.3013	10	27.0718	13
SYS.2.5.b	85	8.5	9.9	16.2175	11	16.2881	12	16.2175	11	16.1536	14
SYS.2.6.a	95	6.5	7.9	29.3127	6	29.3524	10	29.2633	8	29.2633	11
SYS.2.6.b	95	8.5	9.9	17.4633	9	17.4588	12	17.4587	10	17.1971	13

(continued)

Table 5 (continued)

#Inst.				The first runtime limit t_1				The second runtime limit t_2			
Label	n	W	H	HY		ITTSBH		HY		ITTSBH	
				L_{HY}	$\omega^\star$	$L^\star$	$\omega^\star$	L_{HY}	$\omega^\star$	$L^\star$	$\omega^\star$
SYS.3.4.a	85	5.5	6.9	32.6665	7	32.6755	12	32.6665	7	32.5294	15
SYS.3.4.b	85	8.5	9.9	14.2483	7	14.2255	13	14.2055	11	14.0963	14
SYS.3.5.a	90	5.5	6.9	33.9108	8	33.9317	8	33.7800	9	33.7144	10
SYS.3.5.b	90	8.5	9.9	14.7881	9	14.7222	10	14.6783	11	14.5575	12
SYS.3.6.a	100	5.5	6.9	36.7125	6	36.7125	10	36.6254	7	36.3645	13
SYS.3.6.b	100	8.5	9.9	15.8566	6	15.9849	8	15.7994	9	15.6466	10
Average				23.0231		**22.9845**		22.9267		**22.7952**	
$>$						12				21	
$=$						2				1	
$<$						8				0	

strategy used (hill-climbing) sometimes requires more time to converge towards good solutions. On the other hand, the greedy procedure remains very heavy whenever the completion process is applied for all partial solutions.

5 Conclusion

In this paper, we proposed a hybrid algorithm for approximately solving the 3-dimensional sphere packing problem. The proposed algorithm applies a truncated tree search which is based (i) on building a subset of successive eligible nodes, (ii) on searching the best paths that are able to provide a series of complete solutions and (iii) on combining both hill-climbing and bounding strategies for accelerating the search process. The principle of the search process is based on building and exploring a series of neighborhoods related to a series of partial solutions. In order to complete/improve the quality of the partial solutions realized by the packing process, a greedy solution procedure is used. The performance of the proposed algorithm was computationally analyzed on a set of benchmark instances taken from the literature and a set of new generated instances. The provided results were compared to those realized by six recent algorithm available in the literature. The obtained results show that the truncated tree search-based heuristic was able to improve most existing solutions.

References

1. H. Akeb, A look-forward heuristic for packing spheres into a three-dimensional bin, in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, ed. by M. Ganzha, L. Maciaszek, M. Paprzycki, ACSIS, vol. 2, pp. 397–404 (2014). doi:10.15439/2014F145
2. E.G. Birgin, F.N.C. Sobral, Minimizing the object dimensions in circle and sphere packing problems. Comput. Oper. Res. **35**, 2357–2375 (2008). doi:10.1016/j.cor.2006.11.002
3. R.S. Farr, Random close packing fractions of log-normal distributions of hard spheres. Powder Technol. **245**, 28–34 (2013). doi:10.1016/j.powtec.2013.04.009
4. M. Hifi, R. M'Hallah, A literature review on circle and sphere packing problems: models and methodologies. Adv. Oper. Res., Article ID 150624, 22 p. (2009). doi:10.1155/2009/150624
5. M. Hifi, R. M'Hallah, Beam search and non-linear programming tools for the circular packing problem. Int. J. Math. Oper. Res. **1**, 476–503 (2009). doi:10.1504/IJMOR.2009.026278
6. M. Hifi, T. Saadi, A cooperative algorithm for constrained two-staged two-dimensional cutting problems. Int. J. Math. Oper. Res. **9**, 104–124 (2010). doi:10.1504/IJOR.2010.034363
7. M. Hifi, L. Yousef, A dichotomous search-based heuristic for the three-dimensional sphere packing problem. Cogent Eng. (Taylor and Francis Group). (To appear)
8. M. Hifi, L. Yousef, Width beam and hill-climbing strategies for the three-dimensional sphere packing problem, in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, ed. by M. Ganzha, L. Maciaszek, M. Paprzycki, ACSIS, vol. 2, pp. 397–404 (2014). doi:10.15439/2014F284
9. T. Kubach, A. Bortfeldt, T. Tilli, H. Gehring, Parallel greedy algorithms for packing unequal spheres into a cuboidal strip or a cuboid. Working Paper, Department of Management Science,

University of Magdeburg, (Diskussionsbeitrag der Fakultät für Wirtschaftswissenschaft der FernUniversität in Hagen). No 440, Hagen (2009)
10. T. Kubach, A. Bortfeldt, T. Tilli, H. Gehring, Greedy algorithms for packing unequal sphere into a cuboidal strip or a cuboid. Asia-Pac. J. Oper. Res. **28**(06), 739–753 (2011). doi:10.1142/S0217595911003326
11. Y. Li, W. Ji, Stability and convergence analysis of a dynamics-based collective method for random sphere packing. J. Comput. Phys. **250**, 373–387 (2013). doi:10.1016/j.jcp.2013.05.023
12. K. Lochmann, L. Oger, D. Stoyan, Statistical analysis of random sphere packings with variable radius distribution. Solid State Sci. **8**(12), 1397–1413 (2006). doi:10.1016/j.solidstatesciences.2006.07.01
13. R. M'Hallah, A. Alkandari, N. Mladenović, Packing unit spheres into the smallest sphere using VNS and NLP. Comput. Oper. Res. **40**(2), 603–615 (2013). doi:10.1016/j.cor.2012.08.019
14. R. M'Hallah, A. Alkandari, Packing unit spheres into a cube using VNS. Electron. Notes Discret. Math. **39**(1), 201–208 (2012)
15. K. Soontrapa, Y. Chen, Mono-sized sphere packing algorithm development using optimized Monte Carlo technique. Adv. Powder Technol. **24**(6), 955–961 (2013). doi:10.1016/j.apt.2013.01.007
16. Y. Stoyan, G. Yaskow, G. Scheithauer, Packing of various radii solid spheres into a parallelepiped. Cent. Eur. J. Oper. Res. **11**, 389–407 (2003)
17. A. Sutou, Y. Dai, Global optimization approach to unequal sphere packing problems in 3D. J. Optim. Theory Appl. **114**, 671–694 (2002). doi:10.1023/A:1016083231326
18. G. Wascher, H. Haussner, H. Schumann, An improved typology of cutting and packing problems. Eur. J. Oper. Res. **183**, 1109–1130 (2007). doi:10.1016/j.ejor.2005.12.047
19. M. Yavuza, Iterated beam search for the combined car sequencing and level scheduling problem. Int. J. Prod. Res. **51**, 3698–3718 (2013). doi:10.1080/00207543.2013.765068

Multi-Objective Meta-Evolution Method for Large-Scale Optimization Problems

Piotr Przystałka and Andrzej Katunin

Abstract The paper deals with the method for searching the proper values of behavioural (relevant) parameters of optimization algorithms for large-scale problems. The authors formulate the optimization task as multi-objective problem taking into account two criteria. The first criterion corresponds to the estimation of the accuracy of a solution, whereas the second one represents the time computational complexity of the main optimization algorithm. In the present study, predominant Pareto optimality concept is used to solve this problem. Moreover, the authors propose to use a much less complicated algorithm in the main optimization engine, while a more advanced approach in the meta-evolution core. The engine of the target optimization algorithm is realised applying the particle swarm optimization algorithm, while the core of the meta-evolution process is implemented by means of the multi-objective evolutionary algorithm. The advantages and limitations of the proposed meta-evolution method were examined employing well-practised test functions described in the literature.

1 Introduction

The meta-evolution which is also known in the literature as hyper-heuristic as well as super- or meta-optimization is a quite novel approach, which found numerous applications in the engineering problems. One of the earliest attempts to meta-optimization can be found in [1], where the genetic algorithm was used in order to find best mutation and crossover rates for another lower-level genetic algorithm. In the next years, there were similar trials to this problem by many authors, e.g. see [2–4]. Also in this subject, the authors of the paper [5] discussed the most important issues related to tuning evolutionary algorithm parameters by means of various meta-optimization methods. Their main conclusion was that it was no matter what kind of tuner algorithms to be used in this task, because for each case, it was possible to get a much

P. Przystałka · A. Katunin (✉)
Silesian University of Technology, Institute of Fundamentals of Machinery Design,
Konarskiego 18a, 44-100 Gliwice, Poland
e-mail: andrzej.katunin@polsl.pl

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_10

better result from evolutionary computations with meta-optimization than relying on own intuition and the usual parameter setting conventions. The similar strategy as in the case of evolutionary algorithms can be observed for other soft computing method. For example in [6], the authors proposed the concept in which a superordinate swarm ('superswarm') can be used to optimize the parameters of subordinate swarms ('subswarms'). Subordinate swarms were used for neural network training. Another point of the view is given in [7]. Branke and Elomari in their work proposed the method that could be used, in a single run, to identify the best parameter settings for all possible computational budgets. Their approach allows to save a lot of time.

The presented study deals with the recent advances in development of the multi-objective meta-evolution approach introduced in [8]. In this paper the authors performed the meta-optimization of wavelet parameters in wavelet-based algorithm used for damage identification in composite structures. Additionally, some tests were performed on multi-modal functions used for optimization benchmarking. In the present study the authors performed an extended analysis on benchmark functions, which were selected in such a way that the optimum achievement is difficult from the optimization point of view. Such an analysis allows for testing the convergence of the proposed algorithm and evaluate the specific behaviour of an algorithm during the searching of a global optimum. The calculation were performed for high-dimensional (up to $D = 500$) functions for specific benchmark functions, including the nowhere derivable Weierstrass function, which allows for generalization of the analysis. The proposed meta-optimization algorithm allows for acceleration of searching the global optimum with simultaneous automatic selection of behavioural parameters, which can find numerous applications in engineering practice.

2 Description of the Proposed Method

In this paper, the idea of the meta-optimization method corresponds to the data flow diagram presented in Fig. 1. This strategy is used in order to search the space of behavioural parameters of the target optimization algorithm. As it can be seen, the meta-optimization algorithm (MAC) evaluates a meta-objective function whereas the main optimization algorithm (OAE) computes the cost function in order to find an optimal solution with the minimum time complexity and maximum accuracy.

Meta-optimization concept can be realized in different ways, however one of the most promising approaches employs the multi-objective optimization algorithm. Consequently, the main purpose of the meta-optimization process is to tune values of behavioural parameters of the main optimization algorithm in order to minimize a multiple objective function **U**. This function can be formulated taking into account two fundamental criteria. The first criterion (MBF) corresponds to the estimation of the accuracy of a solution, whereas the second one (FES) represents the time computational complexity of the main optimization algorithm. When one assumes that both objectives are not conflicted then the multi-objective meta-optimization problem can be stated as follows:

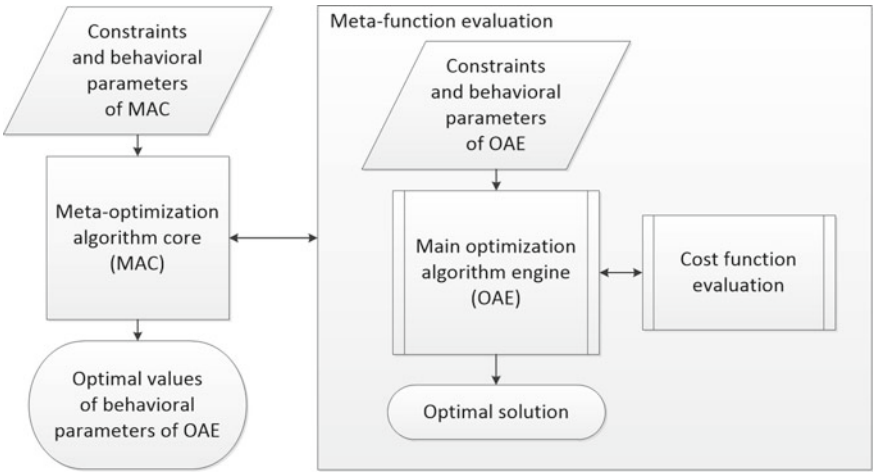


Fig. 1 A data flow diagram of the meta-optimization method

$$\begin{aligned} &\text{Minimize } \mathbf{U}(\mathbf{p}) = \left[\text{MBF}(\mathbf{p}) \;\; \text{FES}(\mathbf{p})\right] \\ &\text{subject to } \Omega(\mathbf{p}), \end{aligned} \tag{1}$$

where $\mathbf{p}$ is the set of behavioural properties of the main algorithm, Ω represents boundaries and constraints in the meta-optimization process. The accuracy of the solution that is found by the main optimization algorithm, can be computed taking into account the mean value and the corrected sample standard deviation of best scores of the cost function evaluations:

$$\text{MBF}(\mathbf{p}) = \bar{F}(\mathbf{p}) + \alpha_p \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(F\left[\mathbf{x}_i^*(\mathbf{p})\right] - \bar{F}\right)^2}, \tag{2}$$

where $N-1$ is the number of degrees of freedom in the vector of residuals, $\bar{F}$ is the sample mean of the cost function, α_p is a critical value corresponding to a given significance level. The optimal solution is assumed as:

$$\mathbf{x}_i^* = \underset{\bar{\Omega}(\mathbf{x}_i)}{\arg\min}\, F\left[\mathbf{x}_i(\mathbf{p})\right], \tag{3}$$

where $\bar{\Omega}$ denotes boundaries and constraints in the target optimization algorithm. On the other hand, the time complexity of the same algorithm is related to the number of steps (p_S) as well as the number of cost function evaluations at each step (p_E) and can be approximated using the total number of cost function evaluations:

$$\text{FES}(\mathbf{p}) = p_S \cdot p_E \tag{4}$$

Generally, multi-objective optimization problems do not have single global solution, and therefore there is the need to investigate a set of points, each of which satisfies the objectives. Due to this, in the present study, predominant Pareto optimality concept is mainly used. A solution is Pareto optimal if there is no other solution that improves at least one objective function without detriment another function [9]. It is often viewed the same as a non-dominated solution.

It is reasonable to expect that each of multi-objective versions of soft computing methods indicated often in the literature can be applicable in the task of meta-optimization. Nevertheless, the authors propose to use a much less complicated algorithm in the main optimization engine (OAE), while a more advanced approach in the meta-optimization core (MAC). In such manner, it is possible to obtain general values of relevant parameters of the main algorithm that can easily be implemented in the embedded system of the end-user device. In this study, it is decided that, the engine of the main optimization algorithm is prepared using the particle swarm optimization algorithm (PSO-OAE), while the core of the meta-optimization process is implemented by means of the multi-objective evolutionary algorithm (MOEA-MAC). MOEAs are known in the literature as the heuristic methods for solving optimization problems, which are based on the natural selection process that mimics biological evolution. The MOEA recommended in [10] is utilized herein to solve the meta-optimization problem defined as (1). Well-known and often practised genetic operators for multi-objective optimization are applied to obtain the convergence of a solution. In such a way, the problem of finding values of behavioural parameters is solved by computing the Pareto front, hence the set of evenly distributed non-dominated optimal solutions are determined. PSO is also classified into heuristic approaches, however this is a population-based stochastic optimization technique, which is inspired by simulation of social behaviour. In this paper, PSO proposed by [11] is adopted and applied to search for the optimal values of $\mathbf{x}$. The both optimization algorithms are implemented in the MATLAB® environment using Genetic Algorithm and Particle Swarm Optimization Toolboxes and these are discussed below in details.

2.1 Particle Swarm Optimization as OAE

The particle swarm optimization is a population-based stochastic optimization technique, which is inspired by simulation of the social behaviour reflected in flock of birds, bees and fish that adapt their movements in order to seek the best food sources as well as to avoid predators [12]. This type of optimization approach is also viewed as a parallel evolutionary computation technique.

Numerically, the basic PSO algorithm can be formulated in vector notation such as it is proposed in [13]. The velocity $\mathbf{v}_k$ is updated using its current value and a term which attracts the particle towards its own previous best position $\mathbf{p}_1$ and globally best position $\mathbf{p}_2$ in the whole swarm:

$$\mathbf{v}_{k+1} = a\mathbf{v}_k + b_1\mathbf{r}_1 \otimes (\mathbf{p}_1 - \mathbf{x}_k) + b_2\mathbf{r}_2 \otimes (\mathbf{p}_2 - \mathbf{x}_k), \tag{5}$$

while, the particle position $\mathbf{x}_k$ is dynamically actualized using its current value and the newly computed velocity $\mathbf{v}_{k+1}$:

$$\mathbf{x}_{k+1} = c\mathbf{x}_k + d\mathbf{v}_{k+1}, \tag{6}$$

where the symbol $\otimes$ denotes element-by-element vector multiplication, a is the momentum factor also known as the inertia weight, coefficients b_1 and b_2 are used to describe the strength of attraction (cognitive and social attraction coefficients, respectively), c and d are black-box tuning parameters, vectors of random numbers $\mathbf{r}_1$ and $\mathbf{r}_2$ that are randomness useful for good state space exploration (they are usually obtained using the generator of pseudo-random numbers with uniform distribution on the range of 0 to 1).

2.2 Multi-Objective Evolutionary Algorithm as MAC

Evolutionary algorithms are known as the methods for solving optimization problems, which are based on the natural selection process that mimics biological evolution. In this study a variant of the non-dominated sorting GA-II called controlled elitist genetic algorithm is used [10]. In order to apply such an optimization technique for tuning behavioural parameters of the main optimization algorithm (OAE) it is necessary to define the following properties of the algorithm: the representation of the individuals, the fitness function, selection and succession methods, crossover and mutation operators. Note that, the chromosome contains information about behavioural parameters of the target algorithm:

$$\mathbf{chr} = \mathbf{p} = \left[\, p_1 \;\; p_2 \;\; \ldots \;\; p_K \,\right] \tag{7}$$

where K is the number of the most relevant parameters of the target optimization algorithm.

The feasible population method is adapted to create a random well-dispersed initial population that satisfies all constraints and bounds Ω in (1). The fitness value of an individual is computed using the fitness function which is declared on the basis of the multi-objective function (1). The selection of the parents to the next generation is achieved using the non-dominated rank and a distance measure of the individuals in the current generation.

Additionally, the diversity of population for convergence to an optimal Pareto front is done by controlling the elite members of the population as the algorithm progresses. In this algorithm, the fraction parameter (e_c) and the distance function are used to control the elitism. The Pareto fraction parameter limits the number of individuals on the Pareto front (elite members), whereas the distance function is necessary to maintain diversity on a front by favouring individuals that are relatively far away on

the front. Moreover, it is decided to use a simple heuristic crossover operator that returns a child that lies on the line containing the two parents in such a way that the distance between the child and the better parent is determined by the user-defined parameter λ_c. The remaining individuals (other than elite and crossover children) are mutation children. They are obtained using the adaptive feasible method which randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation.

3 Benchmark Functions

In order to test the proposed meta-optimization approach several benchmark functions were selected. All of the selected functions are multi-modal. The modality of functions is an important criterion in optimization problems since if a certain function has many local minimums the searching algorithm may be trapped in one of them, which will cause wrong results of search process and may direct search away from the global minimum. Therefore, selection of such functions additionally complicates the optimization problem.

The functions were selected basing on those proposed in the CEC'2008 Special Session and Competition on Large Scale Global Optimization [14]. Their detailed description and parameters selected for the testing purposes are presented below.

The first considered function is a shifted version of the Rosenbrock function (Fig. 2) called after its inventor, H. Rosenbrock [15]. This function is non-convex and non-separable, and often used as a test function for optimization problems because of vary narrow global optimum, which is difficult to achieve. The general form of this function can be expressed as:

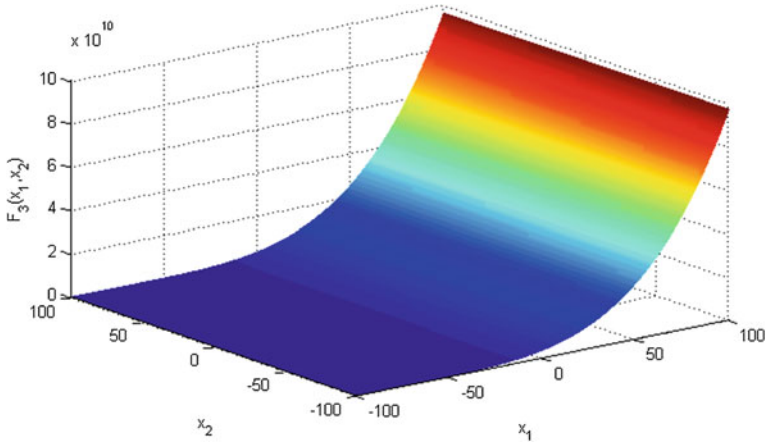


Fig. 2 2D shifted Rosenbrock function

$$F_3(\mathbf{x}) = \sum_{i=1}^{D-1} \left(100\left(z_i^2 - z_{i+1}\right)^2\right), \tag{8}$$

where $\mathbf{z} = \mathbf{x} - \mathbf{o} + \mathbf{1}$, $\mathbf{x} = [x_1, x_2, \ldots, x_D]$, $\mathbf{x} \in [-100, 100]^D$, $\mathbf{o} = [o_1, o_2, \ldots, o_D]$ is the shifted global optimum $\mathbf{x}^* = \mathbf{o}$, $F_3(\mathbf{x}^*) = 0$.

The next considered function is a shifted version of the Rastrigin's function [16], which is non-convex separable function with a huge number of local optima given by:

$$F_4(\mathbf{x}) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right), \tag{9}$$

where $\mathbf{x} \in [-5, 5]^D$, $\mathbf{z}$ is the same as in the previous function, the shifted global optimum $\mathbf{x}^* = \mathbf{o}$, $F_4(\mathbf{x}^*) = 0$. The 2D representation of a shifted Rastrigin's function is shown in Fig. 3.

Further, the shifted version of Griewank's function [17] was considered. This function is widely used for convergence testing of optimization algorithms. It is non-separable and has a huge number of regularly distributed local optima, thus it is not easy to match the global optimum (see Fig. 4). This function is defined as:

$$F_5(\mathbf{x}) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 \tag{10}$$

where $\mathbf{z} = (\mathbf{x} - \mathbf{o})$, $\mathbf{x} \in [-600, 600]^D$, the shifted global optimum $\mathbf{x}^* = \mathbf{o}$, $F_4(\mathbf{x}^*) = 0$.

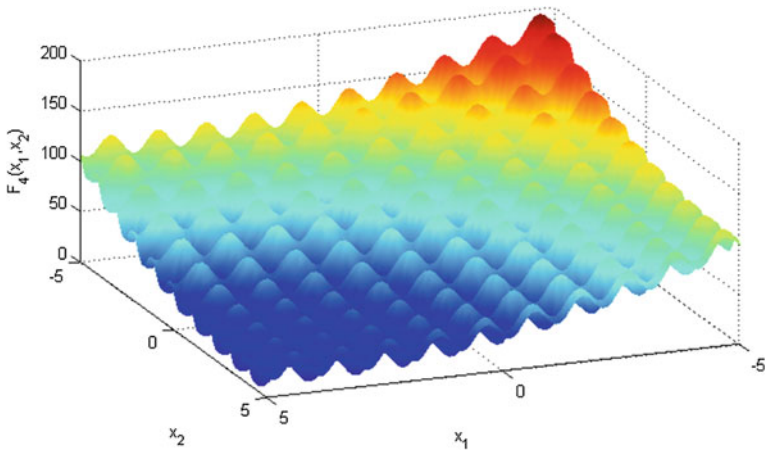


Fig. 3 2D shifted Rastrigin's function

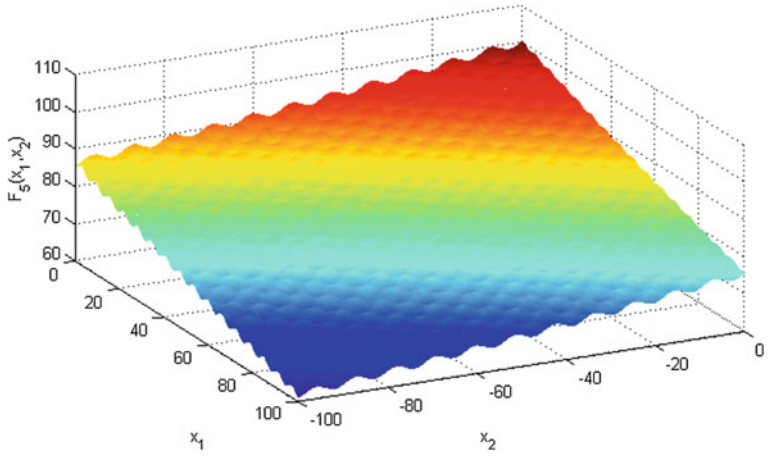


Fig. 4 2D shifted Griewank's function

The last of the commonly used test functions for optimization considered is the shifted version of the Ackley's function [18]. The main difficulty during searching the global optimum is its nearly flat outer region and a large hole in the centre. It is defined as:

$$F_6(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e \quad (11)$$

where $\mathbf{x} \in [-32, 32]^D$, $\mathbf{z}$ is the same as in F_5 function, the shifted global optimum $\mathbf{x}^* = \mathbf{o}$, $F_6(\mathbf{x}^*) = 0$. Its 2D representation is shown in Fig. 5.

Moreover, two fractal-like function were considered. The first function constructed by Macnish [19] is composed of a large number of base functions. In order to preserve the self-similarity property, the average number of base functions in any unit hypercube in the fractal function increases with the inverse of the size to the power of the dimension, which makes the optimization problem extremely difficult (see Fig. 6). This function is given by the following expression:

$$F_7(\mathbf{x}) = \sum_{i=1}^{D} \lambda_1\left(x_i + \lambda_2\left(x_{(i \bmod D)+1}\right)\right) + 1720 \quad (12)$$

where
$\lambda_1(x) \approx \sum_{k=1}^{3}\sum_{1}^{2^{k-1}}\sum_{1}^{\hat{o}_2}\lambda_3\left(x, \hat{o}_1, \frac{1}{2^{k-1}(2-\hat{o}_1)}\right)$,
$\lambda_2(y) = 4\left(y^4 - 2y^3 + y^2\right)$,

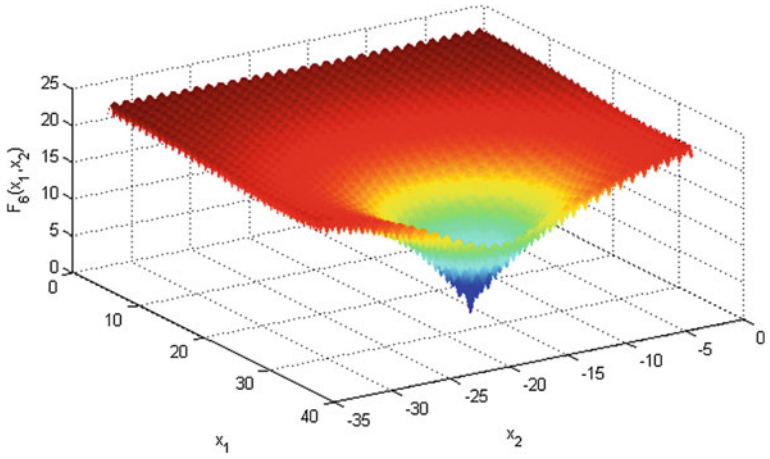


Fig. 5 2D shifted Ackley's function

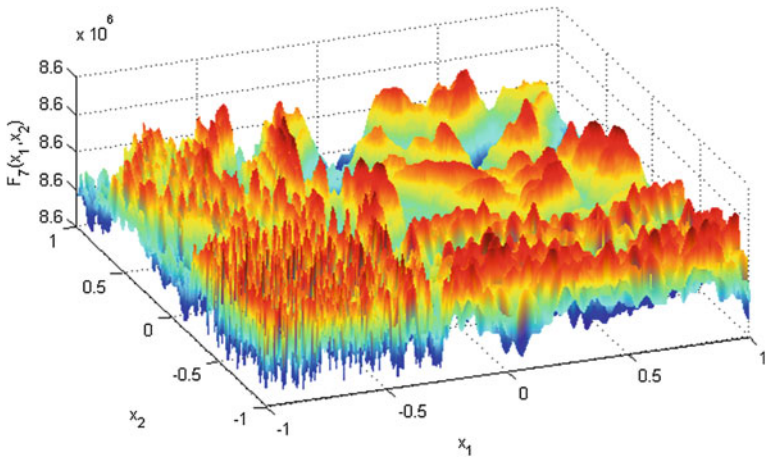


Fig. 6 2D Macnish fractal 'DoubleDip' function

$$\lambda_3(x, c, s) = \begin{cases} \left(-6144(x-c)^6 + 3088(x-c)^4 \right. \\ \left. -392(x-c)^2 + 1\right) s\,, x \in (-0.5, 0.5) \\ 0\,, \text{otherwise} \end{cases}$$

and $\mathbf{x} \in [-1, 1]^D$, the global optimum is unknown, $F_7(\mathbf{x}^*)$ is also unknown, $\hat{o}_1$ is a double precision variable, pseudo-randomly chosen, with seed o_1, with equal probability from the interval $[0, 1]$, $\hat{o}_2$ is an integer variable, pseudo-randomly chosen, with seed o_2, with equal probability from the set $\{0, 1, 2\}$.

The second considered fractal function is a shifted Weierstrass function (Fig. 7). This function is a pathological real-valued function on the real line, which is

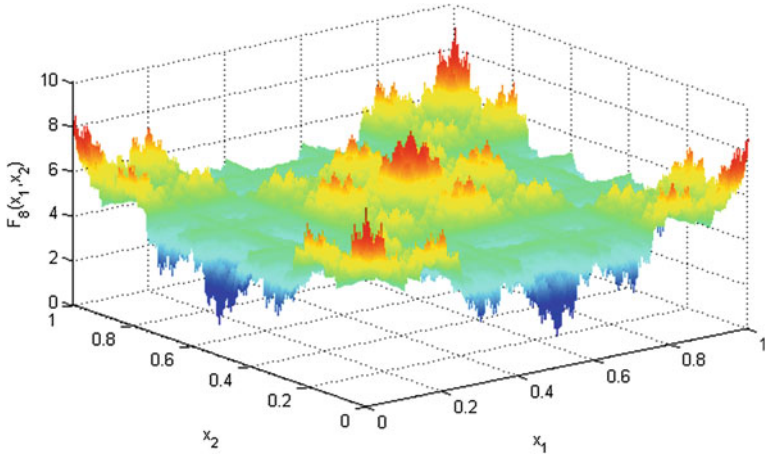


Fig. 7 2D Weierstrass function

continuous everywhere, but nowhere derivable. The function is self-similar and thus it has local optima in the entire domain, which makes the searching procedure of a global optimum extremely difficult. This function is given by:

$$F_8(\mathbf{x}) = 5 + \sum_{i=1}^{D} \left(\frac{1}{2}\right)^i \cos\left(5^i 2\pi z_i\right), \tag{13}$$

where $\mathbf{z} = (\mathbf{x} - \mathbf{o})$, $\mathbf{x} \in [0, 1]^D$, the shifted global optimum $\mathbf{x}^* = \mathbf{o}$, $F_8(\mathbf{x}^*) = 0$.

4 Verification Studies

The advantages and limitations of the proposed meta-evolution method were attempted in the following experiment. The aim of the case study was to validate the performance of the meta-optimization approach over a set of well-practised test functions described in the previous section. As it was mentioned above, the task of the main optimization was defined as a continuous minimization problem. The engine of the target optimization technique was prepared using the particle swarm optimization algorithm (PSO-OAE), while the core of the meta-optimization process was implemented by means of the multi-objective evolutionary algorithm (MOEA-MAC). The cost function in the PSO-OAE was computed using one of the benchmark functions (8–13). In this algorithm only few parameters of the algorithm are relevant to guarantee, as far as possible, to find the optimal solution of the problem. Taking into consideration the analytical proofs given in [13] parameters c and d can be set to unity and d can be set arbitrary without loss of generality. Therefore, behavioural

parameters such as the social and cognitive attraction coefficients ($p_1 = b_1$, $p_2 = b_2$), the number of particles ($p_3 = p_E$), and the total number of generations ($p_4 = p_S$) only were taken into account during the meta-optimization process realized by means of MOEA-MAC.

For each function, the lower and upper boundaries of the PSO-OAE were assumed corresponding to functions' properties (8–13), whereas in the case of MOEA-MAC boundaries were declared in such a way, that the number of particles was equal from D to $2D$, the total number of generations from 5 to 100, the cognitive and social attraction from 0 to 4. The heuristic rules given in the literature were employed to get the best results from the MOEA. The fitness function was declared following to (1), where MBF was computed by averaging the best fitness function (for $N = 25$ and $\alpha_p = 3$) and FES was obtained as the product of the number of particles and the total number of generations of the OAE. It was decided that individuals in the population of the MOEA were composed of genes representing real numeric values of behavioural parameters (the integer parts of p_S and p_E parameters were used during the computations). The total number of generations of MOEA was set to 30. The population size of this algorithm was equal 40. It was decided that the fraction parameter $e_c = 0.8$. For a heuristic crossover operator the user-defined parameter was set to 1.2, and the crossover probability was equal 0.8 (Table 1).

The meta-evolution process was carried out for four cases $D = 2$, $D = 100$, $D = 250$, $D = 500$. The achieved results are presented in Table 2. Besides, in Figs. 8a, c, e and g there are given graphs with the visualisation of selected Pareto fronts (benchmark functions F_5 and F_6 for $D = 500$ as well as F_7 and F_8 for $D = 100$) based on which the optimal values of behavioural parameters were chosen (objective 1 denotes MBF, objective 2 denotes FES). In this case study, the authors selected non-dominated optimal solutions that were characterized by the accuracy of the cost function as higher as possible (in a statistic sense) with the acceptable time complexity of the algorithm (that means the solution was exactly selected from the centre of Pareto front). In order to have much more understandable and comparable results the tuning of behavioural parameters was also carried out with the use of expert's knowledge and trial and error procedure. In the first case, the suggestions proposed in [20] were applied (cognitive attraction $b_1 = 0.5$, social attraction $b_2 = 1.25$). In the second case values of behavioural parameters were changed several times to obtaining satisfactory solutions.

The optimization process was run ten times for each case and afterwards the results were averaged. Overall, the comparison results of meta-evaluation ($\circ$) and classic strategies ($\square$, $\triangle$) for adjusting behavioural parameter values were included in Table 2. The most important statistic measures such as AVG and STD show that the best option is to find optimal values of behavioural parameters by means of the meta-evolution method. It is also confirmed by results presented in Fig. 8 b, d, f, h for functions F_5, F_6 ($D = 500$) and F_7, F_8 ($D = 100$), respectively. These plots demonstrate mean values of the best scores of the cost function (MS) versus the number of function evaluations (FES) for investigated cases. Each of these examples illustrates the effectiveness of the proposed meta-optimization method when compared it to classic approaches.

Table 1 Tuned values of behavioural parameters for benchmark functions

Bench function	Dim	Cognitive attraction (b_1)	Social attraction (b_2)	Generations (p_S)	Population size (p_E)	MBF	FES
F_3	2	1.14E+00	2.00E–01	1.90E+01	7.00E+00	3.68E+02	1.33E+02
	100	9.10E–01	1.34E+00	7.30E+01	3.49E+02	2.85E+09	2.55E+04
	250	1.11E+00	1.15E+00	6.60E+01	3.46E+02	9.63E+10	2.28E+04
	500	7.70E–01	1.13E+00	7.10E+01	1.55E+03	2.77E+11	1.10E+05
F_4	2	9.40E–01	1.13E+00	2.00E+01	1.60E+01	2.66E+00	3.20E+02
	100	1.56E+00	8.40E–01	7.90E+01	2.40E+02	1.04E+03	1.90E+04
	250	1.34E+00	7.30E–01	9.20E+01	4.16E+02	3.01E+03	3.83E+04
	500	9.70E–01	1.08E+00	7.10E+01	1.24E+03	6.77E+03	8.80E+04
F_5	2	3.90E–01	8.20E–01	3.40E+01	1.00E+01	1.40E–01	3.40E+02
	100	1.10E+00	1.25E+00	5.70E+01	2.11E+02	6.03E+02	1.20E+04
	250	1.08E+00	1.43E+00	9.70E+01	4.88E+02	2.56E+03	4.73E+04
	500	9.30E–01	1.14E+00	7.90E+01	6.03E+02	8.25E+03	4.76E+04
F_6	2	3.00E–01	1.04E+00	4.90E+01	9.00E+00	1.00E–02	4.41E+02
	100	1.57E+00	7.60E–01	2.80E+01	8.18E+02	2.03E+01	2.29E+04
	250	6.80E–01	1.14E+00	6.30E+01	3.49E+02	2.07E+01	2.20E+04
	500	8.20E–01	8.10E–01	9.50E+01	5.02E+02	2.07E+01	4.77E+04
F_7	2	4.40E–01	8.50E–01	2.10E+01	1.00E+01	8.60E+06	2.10E+02
	100	6.00E–01	1.03E+00	5.60E+01	1.14E+02	8.60E+06	6.38E+03
	250	8.90E–01	1.20E+00	7.30E+01	2.94E+02	8.60E+06	2.15E+04
	500	1.37E+00	1.15E+00	8.00E+00	9.71E+02	–8.76E+05	7.77E+03
F_8	2	1.00E–02	8.70E–01	3.00E+01	3.00E+00	4.27E+00	9.00E+01
	100	3.20E–01	1.43E+00	6.80E+01	1.06E+02	4.02E+00	7.21E+03

Table 2 Summary results of the target optimization process for tuned values of behavioural parameters

Function	Dim	Case	MAX	MIN	AVG	STD
F_3	2	○	1.110E+04	5.085E+01	1.867E+03	3.416E+03
		□	4.753E+04	1.078E+01	8.210E+03	1.696E+04
		△	4.113E+04	5.079E+01	1.066E+04	1.427E+04
	100	○	2.342E+09	7.491E+08	1.441E+09	5.707E+08
		□	2.507E+09	7.219E+08	1.704E+09	5.602E+08
		△	3.429E+09	5.543E+08	1.734E+09	8.027E+08
	250	○	7.747E+10	4.739E+10	6.744E+10	8.041E+09
		□	9.080E+10	6.669E+10	8.025E+10	9.324E+09
		△	8.431E+10	6.464E+10	7.502E+10	6.394E+09
	500	○	2.672E+11	1.993E+11	2.329E+11	1.956E+10
		□	3.088E+11	2.608E+11	2.804E+11	1.564E+10
		△	3.281E+11	2.323E+11	2.900E+11	2.902E+10
F_4	2	○	2.973E+00	2.581E–01	1.354E+00	9.647E–01
		□	7.051E+00	7.237E–03	1.732E+00	2.660E+00
		△	3.294E+00	8.167E–03	1.408E+00	9.575E–01
	100	○	9.968E+02	7.077E+02	8.443E+02	9.724E+01
		□	1.138E+03	8.519E+02	9.720E+02	8.372E+01
		△	1.043E+03	8.123E+02	9.128E+02	7.045E+01
	250	○	2.919E+03	2.544E+03	2.740E+03	1.017E+02
		□	3.128E+03	2.750E+03	2.891E+03	1.076E+02
		△	3.175E+03	2.624E+03	2.940E+03	1.584E+02
	500	○	6.912E+03	6.524E+03	6.752E+03	1.259E+02
		□	7.178E+03	6.559E+03	6.889E+03	1.856E+02
		△	7.112E+03	6.725E+03	6.951E+03	1.168E+02

(continued)

Table 2 (continued)

Function	Dim	Case	MAX	MIN	AVG	STD
F_5	2	○	4.181E–01	9.612E–03	1.296E–01	1.144E–01
		□	5.163E–01	4.213E–02	1.926E–01	1.826E–01
		△	2.831E–01	3.121E–02	1.053E–01	7.946E–02
	100	○	7.431E+02	4.877E+02	5.484E+02	7.541E+01
		□	8.097E+02	4.965E+02	6.261E+02	9.965E+01
		△	7.213E+02	4.997E+02	6.115E+02	8.054E+01
	250	○	2.425E+03	1.918E+03	2.104E+03	1.525E+02
		□	2.946E+03	2.027E+03	2.441E+03	2.685E+02
		△	2.762E+03	2.243E+03	2.422E+03	1.814E+02
	500	○	8.094E+03	7.253E+03	7.718E+03	2.926E+02
		□	9.298E+03	7.767E+03	8.478E+03	4.156E+02
		△	8.843E+03	7.704E+03	8.291E+03	3.734E+02
F_6	2	○	4.059E–02	1.063E–03	7.288E–03	1.186E–02
		□	2.714E–02	5.580E–04	8.941E–03	8.349E–03
		△	1.993E+01	3.226E–03	2.007E+00	6.299E+00
	100	○	2.030E+01	1.845E+01	1.923E+01	5.588E–01
		□	2.063E+01	1.797E+01	1.992E+01	8.114E–01
		△	2.017E+01	1.832E+01	1.930E+01	5.791E–01
	250	○	2.058E+01	2.025E+01	2.044E+01	8.948E–02
		□	2.073E+01	2.040E+01	2.057E+01	8.963E–02
		△	2.082E+01	2.046E+01	2.064E+01	1.184E–01
	500	○	2.075E+01	2.041E+01	2.060E+01	1.073E–01
		□	2.078E+01	2.060E+01	2.068E+01	5.179E–02
		△	2.077E+01	2.056E+01	2.064E+01	6.798E–02

(continued)

Table 2 (continued)

Function	Dim	Case	MAX	MIN	AVG	STD
F_7	2	○	8.600E+06	8.600E+06	8.600E+06	1.680E+00
		□	8.600E+06	8.600E+06	8.600E+06	1.264E+00
		△	8.600E+06	8.600E+06	8.600E+06	2.715E+00
	100	○	8.599E+06	8.599E+06	8.599E+06	3.753E+01
		□	8.599E+06	8.599E+06	8.599E+06	4.860E+01
		△	8.599E+06	8.599E+06	8.599E+06	3.792E+01
	250	○	8.598E+06	8.598E+06	8.598E+06	6.332E+01
		□	8.598E+06	8.598E+06	8.598E+06	8.737E+01
		△	598E+06	8.597E+06	8.598E+06	8.970E+01
	500	○	8.597E+06	8.597E+06	8.597E+06	3.916E+01
		□	8.597E+06	8.597E+06	8.597E+06	5.085E+01
		△	8.597E+06	8.597E+06	8.597E+06	4.024E+01
F_8	2	○	4.261E+00	4.250E+00	4.255E+00	3.676E–03
		□	4.379E+00	4.250E+00	4.289E+00	4.565E–02
		△	4.649E+00	4.251E+00	4.300E+00	1.232E–01
	100	○	4.019E+00	4.008E+00	4.011E+00	3.225E–03
		□	4.057E+00	4.005E+00	4.023E+00	1.686E–02
		△	4.041E+00	4.015E+00	4.026E+00	8.774E–03

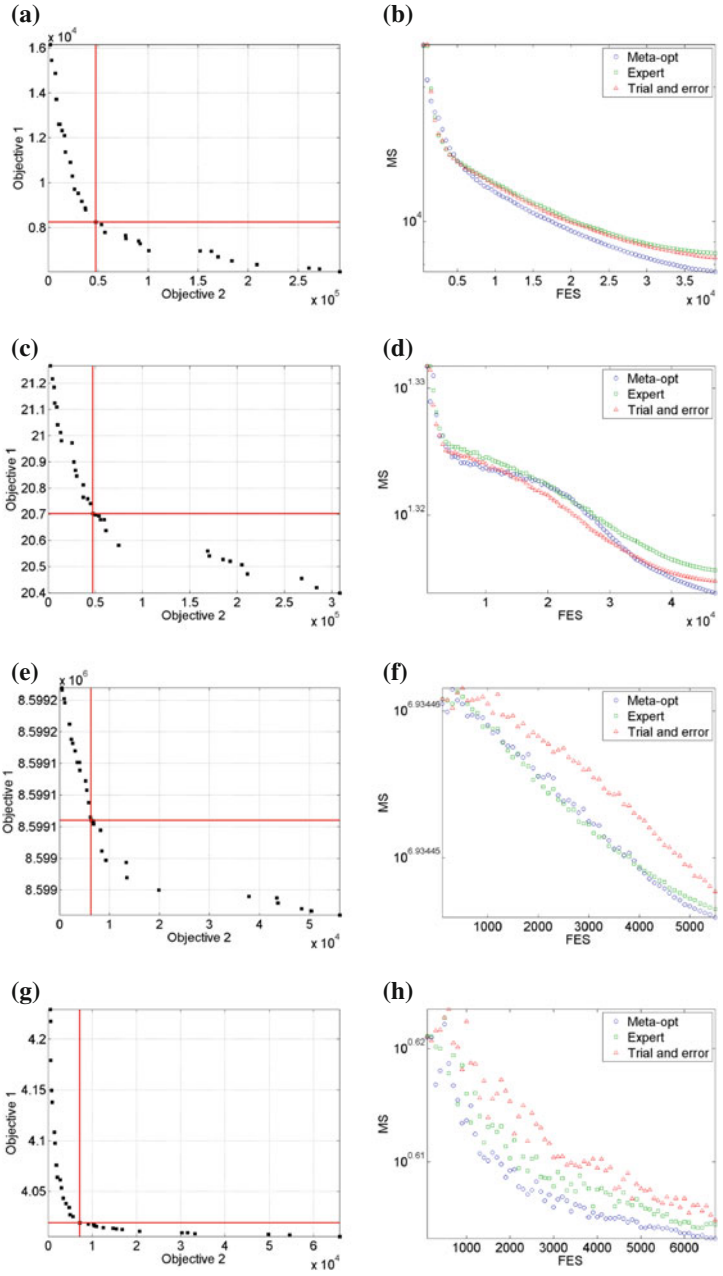


Fig. 8 The comparison results obtained using meta-optimization, expert, trail and error procedures for selection the behavioural parameter values. **a** Meta-opt. of F_5 ($D = 500$). **b** Opt. of F_5 ($D = 500$). **c** Meta-opt. of F_6 ($D = 500$). **d** Opt. of F_6 ($D = 500$). **e** Meta-opt. of F_7 ($D = 100$). **f** Opt. of F_7 ($D = 100$). **g** Meta-opt. of F_8 ($D = 100$). **h** Opt. of F_8 ($D = 100$)

5 Conclusions

The paper presents the upgraded approach for tuning values of behavioural parameters of optimization algorithms especially for large-scale problems. The proposed multi-objective method is based on two criteria. The first one corresponds to the estimation of the accuracy of a solution, whereas the second represents the time computational complexity of the target optimization algorithm. On the one hand, the particle swarm optimization algorithm was employed as the target optimization algorithm and on the other hand the multi-objective evolutionary algorithm was adapted as the core of the meta-evolution process. The proposed meta-evolution method were examined applying well-practised test functions described in the literature. The obtained results confirm that this approach can be used to find proper values of behavioural parameters for which funded solutions to be characterized by the accuracy of the cost function as higher as possible with the acceptable time complexity of the algorithm.

Acknowledgments The authors acknowledge a possibility of carrying out computations on the IBM BladeCenter HS21 in the Academic Computer Centre CYFRONET AGH under the computational grant No. MNiSW/IBM_BC_HS21/PSlaska/012/2013.

References

1. R.E. Mercer, J.R. Sampson, Adaptive search using a reproductive meta-plan. Int. J. Syst. Cybern. **7**, 215–228 (1977)
2. J.J. Grefenstette, Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man Cybern. **16**, 122–128 (1986)
3. A.J. Keane, Genetic algorithm optimization in multi-peak problems: studies in convergence and robustness. Artif. Intell. Eng. **9**, 75–83 (1995)
4. T. Back, Parallel optimization of evolutionary algorithms. Proc. Int. Conf. Evolut. Comput. 418–427 (1994)
5. S.K. Smit, A.E. Eiben, Comparing parameter tuning methods for evolutionary algorithms. Proc. IEEE Congr. Evolut. Comput. (CEC), 399–406 (2009)
6. M. Meissner, M. Schmuker, G. Schneider, Optimized particle swarm optimization (OPSO) and its application to artificial neural network Training. BMC Bioinform. **7**, (2006)
7. J. Branke, J.A. Elomari, Meta-optimization for parameter tuning with a flexible computing budget. in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO12)*, ed by T. Soule, New York (2012), pp. 1245–1252
8. A. Katunin and P. Przystałka, Meta-optimization method for wavelet-based damage identification in composite structures. in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS)*, eds by M. Ganzha, L. Maciaszek, M. Paprzycki, Warsaw, (2014), pp. 429–438
9. R.T. Marler, J.S. Arora, Survey of multi-objective optimization methods for engineering. Struct. Multidiscip. Optim. **26**, 369–395 (2004)
10. K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms. Wiley (2009)
11. S.M. Mikki, A.A. Kishk, Particle Swarm Optimization: A Physics-based Approach. Morgan and Claypool (2008)
12. J. Kennedy, R.C. Eberhart, Y. Shi, *Swarm Intelligence* (Morgan Kaufmann Publishers, San Francisco, 2001)

13. I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf. Process. Lett. **85**, 317–325 (2003)
14. K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, Z. Yang, *Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization* (Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007)
15. H.H. Rosenbrock, An automatic method for finding the greatest or least value of a function. Comput. J. **3**, 175–184 (1960)
16. L.A. Rastrigin, *Systems for Extremal Control* Nauka, Moscow (1974) (in Russian)
17. A.O. Griewank, Generalized decent for global optimization. J. Opt. Theo. Appl. **34**, 11–39 (1981)
18. D.H. Ackley, *A Connectionist Machine for Genetic Hillclimbing* (Kluwer Academic Publishers, Boston, 1987)
19. C. MacNish, Towards unbiased benchmarking of evolutionary and hybrid algorithms for real-valued optimisation. Connect. Sci. **19**, 361–385 (2007)
20. M. Clerc, J. Kennedy, The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE T. Evolut. Comput. **6**, 58–73 (2002)

Dispersive Flies Optimisation and Medical Imaging

Mohammad Majid al-Rifaie and Ahmed Aber

Abstract One of the main sources of inspiration for techniques applicable to complex search space and optimisation problems is nature. This paper introduces a new metaheuristic—Dispersive Flies Optimisation (DFO)—whose inspiration is beckoned from the swarming behaviour of flies over food sources in nature. The simplicity of the algorithm facilitates the analysis of its behaviour. A series of experimental trials confirms the promising performance of the optimiser over a set of benchmarks, as well as its competitiveness when compared against three other well-known population based algorithms. The convergence-independent diversity of DFO algorithm makes it a potentially suitable candidate for dynamically changing environment. In addition to diversity, the performance of the newly introduced algorithm is investigated using the three performance measures of accuracy, efficiency and reliability and its outperformance is demonstrated in the paper. Then the proposed swarm intelligence algorithm is used as a tool to identify microcalcifications on the mammographs. This algorithm is adapted for this particular purpose and its performance is investigated by running the agents of the swarm intelligence algorithm on sample mammographs whose status have been determined by the experts. Two modes of the algorithms are introduced in the paper, each providing the clinicians with a different set of outputs, highlighting the areas of interest where more attention should be given by those in charge of the care of the patients.

Keywords Multi-agent algorithm · Dispersive flies optimisation · Medical imaging · Mammographs

M.M. al-Rifaie (✉)
Department of Computing Goldsmiths, University of London,
SE14 6NW, London, UK
e-mail: m.majid@gold.ac.uk

A. Aber
Department of Cardiovascular Sciences, University of Leicester Royal Infirmary,
LE2 7LX, Leicester, UK
e-mail: ahmed.aber@nhs.net

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_11

1 Introduction

Throughout the history nature has been an inexplicable source of inspiration for scientists and researchers. Observations, many of which made unintentionally, have been triggering the inquisitive minds for hundreds of years. The task of resolving problems and its often present nature in the minds of scientists boosts the impact of these observations, which in cases led to discoveries. Among others, researchers in mathematics, physics and natural sciences have had their fair share of 'observations-leading-to-discoveries'.

Observing the magnificently choreographed movements of birds, behaviour of ants foraging, convergence of honey bees in search for food source and so forth has led several researchers to propose (inspired versus identical) models used to solve various optimisation problems. Genetic Algorithm [10], Particle Swarm Optimisation [11] and Ant Colony Optimisation [8] are only few such techniques belonging to the broader category of swarm intelligence; it investigates collective intelligence and aims at modelling intelligence by looking at individuals in a social context and monitoring their interactions with one another as well as their interactions with the environment.

The work presented here aims at proposing a novel nature-inspired algorithm based on the behaviours of flies hovering over food sources. This model—Dispersive Flies Optimisation or DFO—is first formulated mathematically and then a set of experiments is conducted to examine its performance when presented with various problems.

Afterwards an introduction to metastatic disease is given along with a brief explanation on how to detect metastasis. The swarm intelligence algorithm is adapted for the purpose of this research. Next, a brief summary of x-ray mammography and its use is presented, emphasising on mammographic film reading as a particularly demanding visual task, which could be facilitated using the technique presented in this paper.

2 Dispersive Flies Optimisation

Dispersive Flies Optimisation (DFO) is an algorithm inspired by the swarming behaviour of flies hovering over food sources. The swarming behaviour of flies is determined by several factors and that the presence of threat could disturb their convergence on the marker (or the optimum value). Therefore, having considered the formation of the swarms over the marker, the breaking or weakening of the swarms is also noted in the proposed algorithm.

In other words, the swarming behaviour of the flies, in Dispersive Flies Optimisation, consist of two tightly connected mechanisms, one is the formation of the swarms and the other is its breaking or weakening. The algorithm and the mathematical formulation of the update equations are introduced below.

The position vectors of the population are defined as:

$$\boldsymbol{x}_i^t = \left[x_{i1}^t, x_{i2}^t, \ldots, x_{iD}^t\right], \qquad i = 1, 2, \ldots, NP \tag{1}$$

where t is the current time step, D is the dimension of the problem space and NP is the number of flies (population size).

In the first generation, when $t = 0$, the ith vector's jth component is initialised as:

$$x_{id}^0 = x_{min,d} + r\left(x_{max,d} - x_{min,d}\right) \tag{2}$$

where r is a random number drawn from a uniform distribution on the unit interval $U\ (0, 1)$; x_{min} and x_{max} are the lower and upper initialisation bounds of the dth dimension, respectively. Therefore, a population of flies are randomly initialised with a position for each flies in the search space.

On each iteration, the components of the position vectors are independently updated, taking into account the component's value, the corresponding value of the best neighbouring fly (consider ring topology) with the best fitness, and the value of the best fly in the whole swarm:

$$x_{id}^t = x_{nb,d}^{t-1} + \mathrm{U}\ (0, 1) \times (x_{sb,d}^{t-1} - x_{id}^{t-1}) \tag{3}$$

where $x_{nb,d}^{t-1}$ is the value of the *n*eighbour's *b*est fly in the dth dimension at time step $t - 1$; $x_{sb,d}^{t-1}$ is the value of the *s*warm's *b*est fly in the dth dimension at time step $t - 1$; and $\mathrm{U}\ (0, 1)$ is the uniform distribution between 0 and 1.

The algorithm is characterised by two principle components: a dynamic rule for updating flies position (assisted by a social neighbouring network that informs this update), and communication of the results of the best found fly to other flies.

As stated earlier, the swarm is disturbed for various reasons; one of the positive impacts of such disturbances is the displacement of the disturbed flies which may lead to discovering a better position. To consider this eventuality, an element of stochasticity is introduced to the update process. Based on this, individual components of flies' position vectors are reset if the random number, r, generated from a uniform distribution on the unit interval $\mathrm{U}\ (0, 1)$ is less than the *disturbance threshold* (dt). This guarantees a proportionate disturbance to the otherwise permanent stagnation over a likely local minima.

Algorithm 1 summarises the DFO algorithm.[1]

[1] The source code can be downloaded from the following page: http://doc.gold.ac.uk/~map01mm/DFO/.

Algorithm 1 Dispersive Flies Optimisation

while FE $< 300,000$ **do**
  for $i = 1 \rightarrow NP$ **do**
    $\boldsymbol{x}_i$.fitness $\leftarrow f(\boldsymbol{x}_i)$
  end for
  $sb \leftarrow \{sb,\ \forall\ f(\boldsymbol{x}_{sb}) = \min(f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_{NP}))\}$
  $nb \leftarrow \left\{nb,\ \forall\ f(\boldsymbol{x}_{nb}) = \min\left(f(\boldsymbol{x}_{\text{left}}), f(\boldsymbol{x}_{\text{right}})\right)\right\}$
  for $i = 1 \rightarrow NP$ **do**
    for $d = 1 \rightarrow$ D **do**
      $\tau_d \leftarrow x^{t-1}_{nb,d} + \text{U}(0, 1) \times (x^{t-1}_{sb,d} - x^{t-1}_{id})$
      if $(r < dt)$ **then**
        $\tau_d \leftarrow x_{min,d} + r\left(x_{max,d} - x_{min,d}\right)$
      end if
    end for
    $\boldsymbol{x}_i \leftarrow \boldsymbol{\tau}$
  end for
end while

The next section briefly presents three population-based algorithms which will be used to compare the performance of DFO, and then the results of a series of experiments conducted on DFO over a set of benchmark functions are reported.

3 Experiments

This section presents a set of experiment investigating the performance of the newly introduced Dispersive Flies Optimisation (DFO) and discusses the results. Then, to understand whether disturbance plays an important role in the optimisation process, a *control* algorithm is presented DFO-c where no disturbance is inflicted upon the population of flies.

Recognising the lose of diversity as a common issue in all distribution based evolutionary optimisers (since dispersion reduces with convergence), the impact of disturbance on preserving the diversity of the population is also studied. Additionally, an optimal value for disturbance threshold, *dt*, is suggested. Afterwards the performance of DFO is compared against few other well-known population-based algorithms, namely Particle Swarm Optimisation (PSO), Differential Evolution (DE) and Genetic Algorithm (GA).

3.1 Experiment Setup

The benchmarks used in the experiments (see Table 1) are divided in two sets, f_{1-14} and g_{1-14}; more details about these functions (e.g. global optima, mathematical formulas, etc.) are reported in [3, 15]. The first set, f_{1-14}, have been used by several

Table 1 Benchmark Functions

Fn	Name	Class	D	Feasible Bounds
f_1	Sphere/Parabola	Unimodal	30	$(-100, 100)^D$
f_2	Schwefel 1.2	Unimodal	30	$(-100, 100)^D$
f_3	Generalized Rosenbrock	Multimodal	30	$(-30, 30)^D$
f_4	Generalized Schwefel 2.6	Multimodal	30	$(-500, 500)^D$
f_5	Generalized Rastrigin	Multimodal	30	$(-5.12, 5.12)^D$
f_6	Ackley	Multimodal	30	$(-32, 32)^D$
f_7	Generalized Griewank	Multimodal	30	$(-600, 600)^D$
f_8	Penalized Function P8	Multimodal	30	$(-50, 50)^D$
f_9	Penalized Function P16	Multimodal	30	$(-50, 50)^D$
f_{10}	Six-hump Camel-back	Low dimensional	2	$(-5, 5)^D$
f_{11}	Goldstein-Price	Low dimensioal	2	$(-2, 2)^D$
f_{12}	Shekel 5	Low dimensioal	4	$(0, 10)^D$
f_{13}	Shekel 7	Low dimensioal	4	$(0, 10)^D$
f_{14}	Shekel 10	Low dimensioal	4	$(0, 10)^D$
g_1	Shifted Sphere	Unimodal	30	$(-100, 100)^D$
g_2	Shifted Schwefel 1.2	Unimodal	30	$(-100, 100)^D$
g_3	Shifted Rotated High Conditioned Elliptic	Unimodal	30	$(-100, 100)^D$
g_4	Shifted Schwefel 1.2 with Noise in Fitness	Unimodal	30	$(-100, 100)^D$
g_5	Schwefel 2.6 Global Optimum on Bounds	Unimodal	30	$(-100, 100)^D$
g_6	Shifted Rosenbrock	Multimodal	30	$(-100, 100)^D$
g_7	Shifted Rotated Griewank without Bounds	Multimodal	30	$(-600, 600)^D$
g_8	Shifted Rotated Ackley with Global Optimum on Bounds	Multimodal	30	$(-32, 32)^D$
g_9	Shifted Rastrigin	Multimodal	30	$(-5, 5)^D$
g_{10}	Shifted Rotated Rastrigin	Multimodal	30	$(-5, 5)^D$
g_{11}	Shifted Rotated Weierstrass	Multimodal	30	$(-0.5, 0.5)^D$
g_{12}	Schwefel Problem 2.13	Multimodal	30	$(-\pi, \pi)^D$
g_{13}	Expanded Extended Griewank plus Rosenbrock	Expanded	30	$(-5, 5)^D$
g_{14}	Shifted Rotated Expanded Scaffer	Expanded	30	$(-100, 100)^D$

authors [3, 12, 14] and it contains the three classes of functions recommended by Yao et al. [17]: unimodal and high dimensional, multimodal and high dimensional, and low dimensional functions with few local minima. In order not to initialise the flies on or near a region in the search space known to have the global optimum, *region scaling* technique is used [9], which makes sure the flies are initialised at a corner of the search space where there are no optimal solutions.

The second test set, g_{1-14}, are the first fourteen functions of CEC 2005 test suite [15] and they present more challenging features of the common functions from the aforementioned test set (e.g. shifted by an arbitrary amount within the search space and/or rotated). This set has also been used for many researchers.

One hundred flies were used in the experiments and the termination criterion for the experiments is set to reaching 300,000 function evaluations (FEs). There are 50 Monte Carlo simulations for each experiment and the results are averaged over these independent simulations. Apart from the disturbance threshold which is set to $dt = 0.001$, there are no adjustable parameters in DFO's update equation.

The aim of the experiments is to study and demonstrate the qualities of the newly introduced algorithm as a population based continuous optimiser. The behaviour of the DFO algorithm is compared against its control counterpart and some other population based algorithms.

In this work, a standard particle swarm version, Clerc-Kennedy PSO (PSO-CK) is used. In terms of DE, *DE/best/1* variation of mutation approaches is deployed with CR and F set to 0.5. In GA algorithm, the probabilities of crossover and mutation of the individuals is set to $p_c = 0.7$ and $p_m = 0.9$ respectively. The tournament size of the tournament selection is set to two, and elitism with an elite size of one is deployed to maintain the best found solution in the population.

The details of these algorithms and the rest of configuration is given in [1].

3.2 Performance Measures and Statistical Analysis

In order to conduct the statistical analysis measuring the presence of any significant difference in the performance of the algorithms, Wilcoxon 1×1 non-parametric statistical test is deployed. The performance measures used in this paper are error, efficiency, reliability and diversity which are described below.

Error is defined by the quality of the best agent in terms of its closeness to the optimum position (if knowledge about the optimum position is known *a priori*, which is the case here). Another measure used is *efficiency* which is the number of function evaluations before reaching a specified error, and *reliability* is the percentage of trials where a specified error is reached. These performance measures are defined as below:

$$\text{ERROR} = \left| f\left(\boldsymbol{x}_g\right) - f\left(\boldsymbol{x}_o\right) \right| \tag{4}$$

$$\text{EFFICIENCY} = \frac{1}{n} \sum_{i=1}^{n} \text{FEs} \tag{5}$$

$$\text{RELIABILITY} = \frac{n^{'}}{n} \times 100 \tag{6}$$

where $\boldsymbol{x}_g$ is the best position found and $\boldsymbol{x}_o$ is the position of the known optimum solution; n is the number of trials in the experiment and $n^{'}$ is the number of successful trials, FEs is the number of function evaluations before reaching the specified error, which in these experiments, set to 10^{-8}.

In this work, *diversity*, which is the degree of convergence and divergence, is defined as a measure to study the population's behaviour with regard to exploration and exploitation. There are various approaches to measure diversity. The average distance around the population centre is shown [13] to be a robust measure in the presence of outliers and is defined as:

$$\text{DIVERSITY} = \frac{1}{NP} \sum_{i=1}^{NP} \sqrt{\sum_{j=1}^{D} \left(x_i^j - \bar{x}^j\right)^2} \tag{7}$$

$$\bar{x}^j = \frac{1}{NP} \sum_{i=1}^{NP} x_i^j \tag{8}$$

where NP is the number of flies in the population, D is the dimensionality of the problem, x_i^j is the value of dimension j of agent i, and $\bar{x}^j$ is the average value of dimension j over all agents.

3.3 Performance of Dispersive Flies Optimisation

The error, efficiency and reliability results of DFO performance over the benchmarks are reported in Table 2. The first five columns detail the error-related figures and the last column highlights the median efficiency along with the reliability (shown between brackets) of the algorithm in finding the optima. The algorithm exhibits a promising performance in optimising the presented problem set where half the benchmarks ($f_{1-2,5-11}$ and $g_{1-2,7,9}$) are optimised with the specified accuracy. The figures in the table are expanded in the following categories:

Unimodal, high dimensional ($\boldsymbol{f_{1,2}, g_{1-5}}$) The algorithm optimises 57 % of the benchmarks in this category; while both functions in the first set are optimised ($f_{1,2}$), only two out of five benchmarks in the second and more challenging set are optimised to the specified accuracy. All optimised benchmarks achieve 100 % success.

Table 2 DFO—Dispersive Flies Optimisation

	Min.	Max.	Median	Mean	StdDev	Eff. (Rel. %)
f_1	6.46E−47	1.97E−40	1.75E−43	1.07E−41	3.49E−41	46850 (100)
f_2	2.24E−12	6.01E−10	6.46E−11	1.08E−10	1.26E−10	239850 (100)
f_3	1.74E−04	1.45E+01	3.65E−01	2.17E+00	3.62E+00	∞ (0)
f_4	3.89E−07	5.05E−03	2.87E−05	2.49E−04	7.81E−04	∞ (0)
f_5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	84850 (100)
f_6	2.84E−14	6.39E−14	3.91E−14	3.88E−14	6.49E−15	121200 (100)
f_7	0.00E+00	1.54E−01	1.85E−02	3.25E−02	3.74E−02	47450 (28)
f_8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	50950 (100)
f_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	55550 (100)
f_{10}	0.00E+00	2.22E−16	0.00E+00	4.00E−17	8.62E−17	1700 (100)
f_{11}	0.00E+00	8.10E+01	8.10E+01	5.51E+01	3.82E+01	2100 (32)
f_{12}	5.05E+00	5.05E+00	5.05E+00	5.05E+00	0.00E+00	∞ (0)
f_{13}	5.27E+00	5.27E+00	5.27E+00	5.27E+00	0.00E+00	∞ (0)
f_{14}	5.36E+00	5.36E+00	5.36E+00	5.36E+00	0.00E+00	∞ (0)
g_1	5.68E−14	2.27E−13	1.71E−13	1.49E−13	4.28E−14	45300 (100)
g_2	4.55E−12	9.78E−10	3.88E−11	1.03E−10	1.57E−10	234100 (100)
g_3	3.58E+05	3.22E+06	1.40E+06	1.38E+06	6.23E+05	∞ (0)
g_4	1.40E+00	2.38E+02	2.18E+01	3.71E+01	4.74E+01	∞ (0)
g_5	3.47E+03	1.82E+04	8.95E+03	9.26E+03	3.17E+03	∞ (0)
g_6	1.66E−03	1.51E+02	3.06E+00	1.41E+01	3.05E+01	∞ (0)
g_7	3.31E−11	2.64E−01	1.97E−02	2.93E−02	4.05E−02	236800 (10)
g_8	2.00E+01	2.02E+01	2.01E+01	2.01E+01	3.11E−02	∞ (0)
g_9	1.14E−13	2.27E−13	1.71E−13	1.52E−13	3.71E−14	89450 (100)
g_{10}	1.29E+02	3.42E+02	2.34E+02	2.38E+02	5.62E+01	∞ (0)
g_{11}	2.46E+01	4.02E+01	3.11E+01	3.12E+01	3.23E+00	∞ (0)
g_{12}	9.73E+01	1.58E+04	2.34E+03	3.62E+03	3.51E+03	∞ (0)
g_{13}	9.34E−01	2.01E+00	1.48E+00	1.48E+00	3.07E−01	∞ (0)
g_{14}	1.23E+01	1.40E+01	1.35E+01	1.35E+01	3.69E−01	∞ (0)

Low dimensional and few local minima (f_{10-14}) In this category, 40 % of the benchmarks are optimised, with 100 % reliability for f_{10} and 32 % for f_{11}. However, none of the Shekel functions (f_{12-14}) are optimised; Shekel is known to be a challenging function to optimise due to the presence of several broad sub-optimal minima; also the proximity of a small number of optima to the Shekel parameter $\boldsymbol{a}_i$ is another reason for the difficulty of optimising these set of functions.

Multimodal, high dimensional (f_{3-9}, g_{6-14}) The optimiser is able to optimise 50 % of the benchmarks in this category (f_{5-9} and $g_{7,9}$), 71 % of which achieve 100 % success rate (all except f_7, g_7 with 28 and 10 % success rates respectively). The optimiser exhibit a promising performance when dealing with the difficult Rosenbrock functions (f_3, g_6), reaching the error of 10^{-4} and 10^{-3} respectively. The algorithm

performs exceptionally well in optimising the infamous Rastrigin functions, both common and shifted mode (i.e. f_5 and g_9), achieving 100 % success rate; however it does show weakness in the more challenging g_{10} rotated version.

The success of the optimiser in optimising the notorious Rastrigin function in its common and shifted modes will be discussed in the context of DFO's dimension-to-dimension disturbance mechanism induced by the algorithm.

In order to provide a better understanding of the behaviour of the algorithm, in the next section, the disturbance is discarded and the diversity of the algorithm is studied.

3.4 Diversity in DFO

Most swarm intelligence and evolutionary techniques commence with exploration and, over time (i.e. function evaluations or iterations), lean towards exploitation. Maintaining the right balance between exploration and exploitation phases has proved to be difficult. The absence of the aforementioned balance leads to a weaker diversity when encountering a local minimum and thus the common problem of pre-mature convergence to a local minimum surfaces. Similar to other swarm intelligence and evolutionary algorithms, DFO commences with exploration and over time, through its mechanism (i.e. gradual decrease in the distance between the members of the population and as such, each agent's local and global best positions), moves towards exploitation. However, having implemented the disturbance threshold, a dose of diversity (i.e. *dt*) is introduced in the population throughout the optimisation process, aiming to enhance the diversity of the algorithm.

Figure 1 illustrates the convergence of the population towards the optima and their diversities in three random trials over three benchmarks (i.e. $g_{1,7,9}$ chosen from the second set) as examples from unimodal and multimodal functions. The difference between the error and the diversity values demonstrates the algorithm's ability in exploration while converging to the optima whose fitness reach as low as 10^{-13} in g_1 and g_9.

Exploring the role of disturbance in increasing diversity, a control algorithm is proposed (DFO-c) where there is no disturbance ($dt = 0$) during the position update process. The graphs in Fig. 1 illustrate the diversity of DFO-c populations in randomly chosen trials over three sample benchmarks (again $g_{1,7,9}$). The graphs illustrate that the diversity of the population in DFO-c is less than DFO, thus emphasising the impact of disturbance in injecting diversity which in turn facilitates the escape from local minima (e.g. as demonstrated in case of the highly multimodal Rastrigin functions f_5, g_9). Note the gradual shrinkage of diversity in g_9 ($\approx 10^{-13}$) which is a clear indication of a premature convergence to a local minima with very poor chance of escape.

In order to compare the performance of DFO and its control counterpart, Table 3 presents the result of optimising the benchmarks using DFO-c. Additionally, a statistical analysis is conducted and the output is reported in Table 4 where the performance

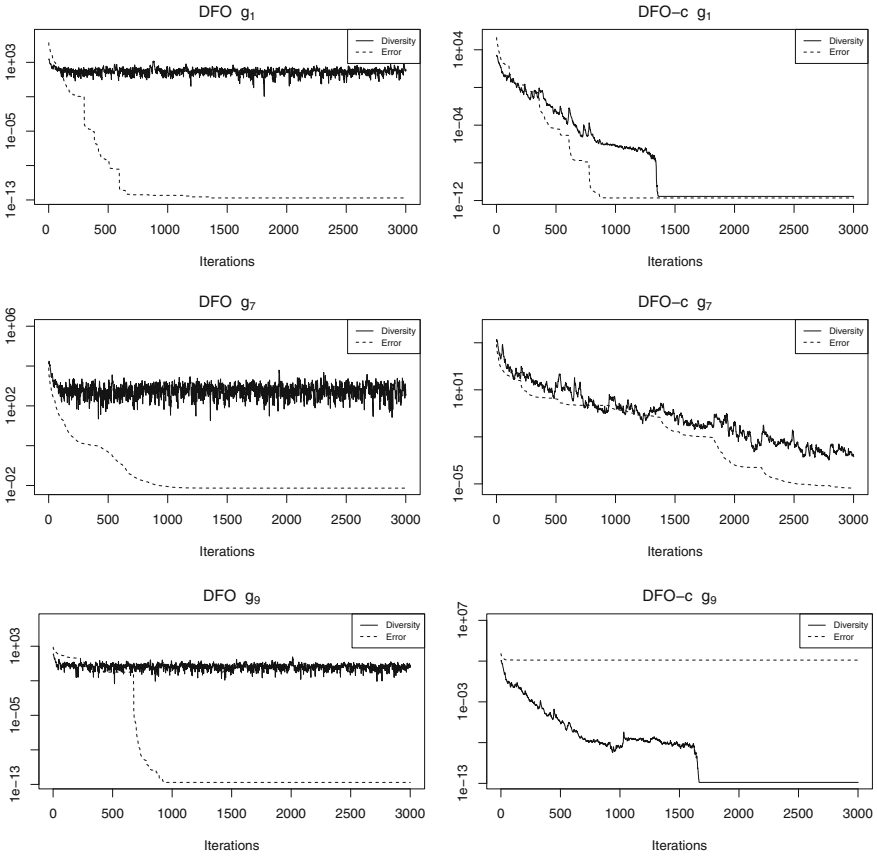


Fig. 1 DFO and DFO-c: diversity and error in $g_{1,7,9}$

is compared using the three aforementioned measures of error, efficiency and reliability (see Sect. 3.2 for the definitions of the measures). The results show that in 89 % of cases (where there is a significant difference between the two algorithms), DFO is performing significantly better than its control counterpart (DFO-c) which is stripped from the diversity inducing disturbance. Furthermore, in all multimodal functions (f_{3-9} and g_{6-12}), whenever there is a statistically significant difference between DFO and DFO-c, the former demonstrates significant outperformance over the later.

Following on the results from measuring error, Table 4 also shows that in terms of efficiency and reliability measures, DFO is 79 % more efficient than its control counterpart, and 92 % more reliable.

Table 3 DFO-c—Control DFO Algorithm

	Min.	Max.	Median	Mean	StdDev	Eff. (Rel. %)
f_1	1.44E−56	3.09E−36	1.27E−45	9.65E−38	4.55E−37	65400 (100)
f_2	7.29E−09	3.23E+01	1.28E−04	7.60E−01	4.60E+00	298200 (2)
f_3	5.27E−05	1.61E+02	5.08E+00	1.67E+01	3.08E+01	∞ (0)
f_4	4.48E−09	3.20E+03	1.55E+03	1.40E+03	8.66E+02	141500 (2)
f_5	1.87E+02	4.17E+02	2.96E+02	2.94E+02	5.76E+01	∞ (0 %)
f_6	1.97E+01	2.00E+01	1.98E+01	1.98E+01	5.24E−02	∞ (0)
f_7	2.22E−16	6.00E+00	9.30E−02	3.51E−01	8.72E−01	64050 (8)
f_8	1.03E−32	3.30E+02	2.14E+00	2.35E+01	5.84E+01	132950 (24)
f_9	0.00E+00	1.57E+02	1.54E−01	5.35E+00	2.27E+01	176500 (30)
f_{10}	0.00E+00	2.22E−16	0.00E+00	7.99E−17	1.08E−16	1700 (100)
f_{11}	0.00E+00	8.10E+01	8.10E+01	5.99E+01	3.59E+01	2100 (26)
f_{12}	5.05E+00	5.05E+00	5.05E+00	5.05E+00	0.00E+00	∞ (0)
f_{13}	5.27E+00	5.27E+00	5.27E+00	5.27E+00	0.00E+00	∞ (0)
f_{14}	5.36E+00	5.36E+00	5.36E+00	5.36E+00	0.00E+00	∞ (0)
g_1	5.68E−14	9.37E−05	1.14E−13	1.91E−06	1.33E−05	70600 (94)
g_2	1.68E−09	2.23E+01	1.23E−04	4.63E−01	3.14E+00	257700 (2)
g_3	2.18E+05	5.38E+06	1.67E+06	1.73E+06	9.39E+05	∞ (0)
g_4	2.23E+02	1.74E+04	1.80E+03	2.91E+03	3.36E+03	∞ (0)
g_5	5.79E+03	1.38E+04	8.50E+03	8.69E+03	2.00E+03	∞ (0)
g_6	2.25E−04	9.53E+01	8.61E+00	1.68E+01	2.52E+01	∞ (0)
g_7	3.01E−10	2.13E−01	3.02E−02	4.17E−02	4.41E−02	263900 (2)
g_8	2.00E+01	2.02E+01	2.00E+01	2.01E+01	3.89E−02	∞ (0)
g_9	8.36E+01	2.64E+02	1.62E+02	1.64E+02	4.61E+01	∞ (0)
g_{10}	1.22E+02	4.93E+02	2.69E+02	2.71E+02	7.69E+01	∞ (0)
g_{11}	1.98E+01	4.11E+01	3.10E+01	3.13E+01	3.97E+00	∞ (0)
g_{12}	2.32E+02	1.38E+04	3.04E+03	4.78E+03	3.88E+03	∞ (0)
g_{13}	4.79E+00	3.56E+01	1.47E+01	1.58E+01	6.47E+00	∞ (0)
g_{14}	1.28E+01	1.45E+01	1.36E+01	1.37E+01	3.38E−01	∞ (0)

3.5 Fine Tuning Disturbance Threshold

The role of disturbance in increasing the diversity of DFO population is discussed earlier (Sect. 3.4). Also, the importance of disturbance is investigated on the optimisation capability of DFO by introducing a control algorithm which lacks the disturbance mechanism and the results demonstrate the positive impact of this mechanism. The aim of this section is to recommend a value for the disturbance threshold, *dt*. The range of disturbance probabilities used in this experiment is between 1 and 10^{-9} and the values were chosen according to:

$$dt_n = 10^{-n}, \quad 0 \leq n \leq 9$$

Table 4 Comparing DFO and DFO-c Performance

DFO—DFO-c			
	Error	Efficiency	Reliability
f_1	o–X	1–0	–
f_2	X–o	1–0	1–0
f_3	X–o	–	–
f_4	X–o	0–1	0–1
f_5	X–o	1–0	1–0
f_6	X–o	1–0	1–0
f_7	X–o	1–0	1–0
f_8	X–o	1–0	1–0
f_9	X–o	1–0	1–0
f_{10}	o–X	0–1	–
f_{11}	–	0–1	1–0
f_{12}	–	–	–
f_{13}	–	–	–
f_{14}	–	–	–
g_1	–	1–0	1–0
g_2	X–o	1–0	1–0
g_3	X–o	–	–
g_4	X–o	–	–
g_5	–	–	–
g_6	–	–	–
g_7	X–o	1–0	1–0
g_8	–	–	–
g_9	X–o	1–0	1–0
g_{10}	X–o	–	–
g_{11}	–	–	–
g_{12}	–	–	–
g_{13}	X–o	–	–
g_{14}	X–o	–	–
	16–2	11–3	11–1

Based on Wilcoxon 1×1 Non-Parametric Statistical Test, if the *error* difference between each pair of algorithms is significant at the 5 % level, the pairs are marked. *X–o* shows DFO is significantly outperforming its counterpart algorithm; and *o–X* shows that the algorithm compared to DFO is significantly better than DFO. In terms of the *efficiency* and *reliability* measures, *1–0* (or *0–1*) indicates that the left (or right) algorithm is more efficient/reliable. The figures, *n–m*, in the last row present a count of the number of X's or 1's in the respective columns

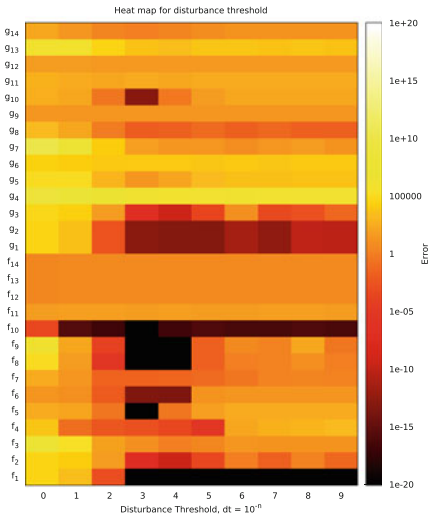


Fig. 2 Fine tuning disturbance threshold

Figure 2 illustrates the performance of DFO using these *dt* probabilities. Both set of benchmarks (i.e. f_{1-14} and g_{1-14}) have been used to find a suitable value for the disturbance threshold. As the heat map highlights, the optimal range is $10^{-2} < dt < 10^{-4}$ and the overall recommended value of $dt = 10^{-3}$ is suggested as a good compromise.

3.6 Comparing DFO with Other Population-Based Optimisers

Having presented the performance of the DFO algorithm (taking into account the three performance measures of error, efficiency and reliability, as well as the diversity of its population and the impact of disturbance on its behaviour), this section focuses on contrasting the introduced algorithm with few well-known optimisation algorithms. The three population algorithms deployed for this comparison are Differential Evolution, Particle Swarm Optimisation and Genetic Algorithm. In this comparison, only the second and the more challenging set of benchmarks, g_{1-14} are used. Table 5 presents the optimising results of the aforementioned algorithms, and as shown, the algorithms have optimised some of the benchmark to the specified accuracy, 10^{-8}. Table 6 shows the result of the statistical analysis comparing DFO with the other three optimisers. Based on this comparison, whenever there is a significant difference between the performance of DFO and the other algorithms, DFO significantly outperforms DE, PSO and GA in 66.67, 58.33 and 85.71 % of the cases, respectively. Table 7 summaries the efficiency results of the three optimisers with that of DFO; note that only the efficiency of functions reaching the specified

Table 5 DE (Differential Evolution), PSO (Particle Swarm Optimisation) and GA (Genetic Algorithm)

	DE		PSO		GA	
	Error	Eff. (Rel. %)	Error	Eff. (Rel. %)	Error	Eff. (Rel. %)
g_1	1.38E−13	21500 (100)	5.23E−14	656236 (100)	5.04E−05	∞ (0)
g_2	1.72E−07	∞ (0)	1.33E−01	∞ (0)	1.21E+04	∞ (0)
g_3	9.65E+06	∞ (0)	1.52E+06	∞ (0)	1.47E+07	∞ (0)
g_4	4.92E−01	∞ (0)	7.89E+03	∞ (0)	5.13E+04	∞ (0)
g_5	2.34E+03	∞ (0)	5.04E+03	∞ (0)	2.09E+04	∞ (0)
g_6	2.30E+00	265800 (12)	2.16E+01	∞ (0)	7.23E+02	∞ (0)
g_7	5.39E−01	∞ (0)	1.04E−02	279653 (10)	5.48E+03	∞ (0)
g_8	2.09E+01	∞ (0)	2.09E+01	∞ (0)	2.04E+01	∞ (0)
g_9	3.47E+01	∞ (0)	9.59E+01	∞ (0)	2.20E+01	∞ (0)
g_{10}	1.47E+02	∞ (0)	1.14E+02	∞ (0)	1.39E+02	∞ (0)
g_{11}	3.65E+01	∞ (0)	3.00E+01	∞ (0)	1.17E+01	∞ (0)
g_{12}	5.85E+05	∞ (0)	9.51E+03	∞ (0)	8.14E+03	∞ (0)
g_{13}	5.70E+00	∞ (0)	5.35E+00	∞ (0)	2.70E+00	∞ (0)
g_{14}	1.34E+01	∞ (0)	1.25E+01	∞ (0)	1.39E+01	∞ (0)

Table 6 Comparing Error in DFO with DE, PSO and GA

	DFO - DE	DFO - PSO	DFO - GA
g_1	–	o–X	X–o
g_2	X–o	X–o	X–o
g_3	X–o	–	X–o
g_4	o–X	X–o	X–o
g_5	o–X	o–X	X–o
g_6	o–X	X–o	X–o
g_7	X–o	o–X	X–o
g_8	X–o	X–o	X–o
g_9	X–o	X–o	X–o
g_{10}	o–X	o–X	o–X
g_{11}	X–o	–	o–X
g_{12}	X–o	X–o	X–o
g_{13}	X–o	X–o	X–o
g_{14}	–	o–X	X–o
$\sum$	8–4	7–5	12–2

Based on Wilcoxon 1×1 Non-Parametric Statistical Test, if the difference between each pair of algorithms is significant at the 5 % level, the pairs are marked. *X–o* shows that the left algorithm is significantly better than the right one; and *o–X* shows that the right one is significantly better than the left. *n–m* in the row labeled Σ is a count of the number of X's in the columns above

Table 7 Comparing Efficiency in DFO with DE, PSO and GA in this table, 1–0 (0–1) indicates that the left (right) algorithm is more efficient

	DFO - DE	DFO - PSO	DFO - GA
g_1	0–1	1–0	1–0
g_2	1–0	1–0	1–0
g_6	0–1	–	–
g_7	1–0	1–0	1–0
g_9	1–0	1–0	1–0
$\sum$	3–2	4–0	4–0

The figures, *n–m*, in the last row present a count of the number of 1's in the respective columns. Note that non-applicable functions have been removed from the table

Table 8 Comparing Reliability in DFO with DE, PSO and GA in this table, 1–0 (0–1) indicates that the left (right) algorithm is more reliable

	DFO - DE	DFO - PSO	DFO - GA
g_2	1–0	1–0	1–0
g_6	0–1	–	–
g_7	1–0	–	1–0
g_9	1–0	1–0	1–0
$\sum$	3–1	2–0	4–0

The figures, *n–m*, in the last row present a count of the number of 1's in the respective columns. Note that non-applicable functions have been removed from the table

error is given. As shown in the table, DFO, in the majority of cases, outperforms the other algorithms. In other words, although, when compared with DE, DFO only outperforms marginally (60 %), it outperforms both PSO and GA in all cases (100 %). The reliability comparison of DFO with the other optimisers is given in Table 8. DFO is shown to be the most reliable algorithm in this comparison. While DFO outperforms DE in 75 % of cases, it show 100 % outperformance when compared with PSO and GA. In order to compare the diversity of the DFO algorithm with the other three optimisers, three benchmarks were chosen from unimodal and multimodal categories ($g_{1,7,9}$). The result of this comparison is illustrated in Fig. 3. It is shown that DE has the least diversity in both uni- and multimodal functions. On the other hand, the diversity of the population in PSO decreases as the population converges towards an optimum (see g_1); however, when convergence does not occur (e.g. in $g_{7,9}$), PSO maintain its high diversity throughout the optimisation process. GA shows a similar pattern to that of PSO in multimodal functions, which is the gradual diversity decrease over time; however it maintains a higher diversity for the unimodal function than PSO (perhaps attributable to the difference in the fitness of the best positions found in both algorithms). In terms of DFO, diversity is less convergence-dependent and more stable across all modalities.

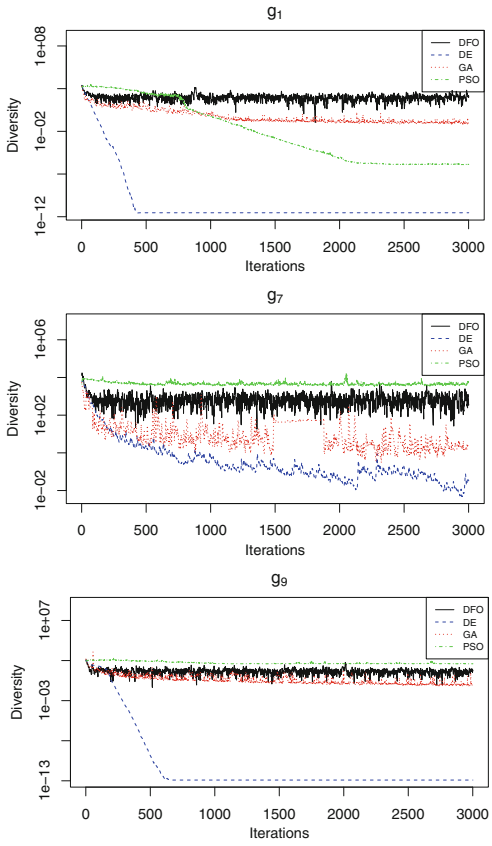


Fig. 3 Diversity of the population in DFO, DE, PSO and GA over three random trials in $g_{1,7 \text{ and } 9}$

4 Computer Aided Diagnosis and Metastatic Disease

Computer aided diagnosis (CAD) is an emerging field in medicine. The technique introduced in this paper can help radiologists to examine the image in greater depth and has the potential to help doctors from different medical disciplines to interpret medical imaging with greater confidence. Furthermore CAD is a promising learning tool for both medical students and junior doctors to develop basic diagnostic skills. This paper presents a new CAD approach in which a recently developed swarm intelligence algorithm—Dispersive Flies Optimisation [1]—is applied to a medical imaging modality where the potential areas of microcalcifications on the x-ray mammography are detected (Fig. 4).

X-ray mammography has been shown to be effective as a method for detecting early breast cancer, but the success of mass screening depends critically on the availability of highly skilled film readers to interpret the images. The majority of film readers in the UK are consultant radiologists and in order to maintain a sufficiently high standard of interpretation, readers are required to undergo training, to keep in

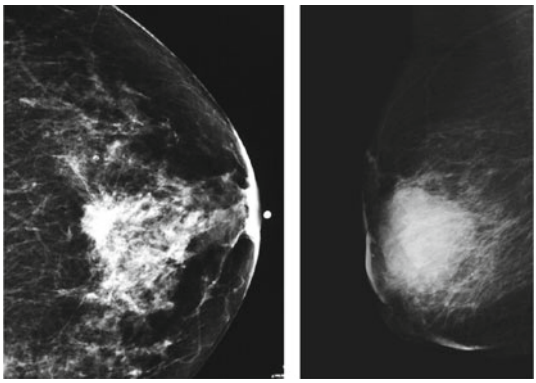


Fig. 4 Mammograph

practice and to evaluate their performance at regular intervals [2]. Mammographic film reading is a particularly demanding visual task. In screening programmes, the film reader must search for extremely infrequent and often very subtle signs of cancer superimposed on complex and variable backgrounds. Early breast cancer may appear in a variety of forms: a few particles of microcalcification; a small ill-defined or speculated mass; abnormal asymmetry between right and left breast images, or subtle distortion of the underlying structure of the breast. These abnormalities vary in size, shape, structure, brightness and location and may share a great deal of similarity with normal mammographic appearances. False negative cases, in which signs of cancer are missed by a reader, sometimes occur. Retrospective evaluation of the previous screening films of cancers detected between screening rounds (interval cancers) and screen-detected cancers show evidence of abnormality in between 16 and 27 % of cases. Some of these signs are very subtle, and may have been seen by the readers but dismissed as being insignificant, but others are clear signs of malignancy [4, 5, 16]. However, different readers miss different cancers, as is evidenced by the success of double reading in which two readers independently read the films [6]. The most accurate method of interpretation is double reading with arbitration, where a third reader reviews cases about which the two readers disagree [6, 7]. In the UK particularly with the National Health Service Breast Screening Programme (NHSBSP) there is an increased demand for skilled manpower to effectively interpret mammographs and double or triple reading of the mammograph is not viable option due to the increased workload. A novel and different method of coping with this is the use of computer-based aids. Researchers have been developing algorithms to detect mammographic abnormalities for more than 30 years with the aim of either automating mammographic interpretation or, more realistically, providing a tool which will enhance human film-reading performance. There are two basic approaches to the problem of detecting abnormalities in mammograms: either to search the images for specific appearances suggestive of cancer, or to characterize normal mammographic appearance to the extent that it is possible to detect anything that fails to conform to the generated model of normality.

The purpose of the current study is to apply for the first time an swarm intelligence algorithm namely dispersive flies optimisation to perform the task of identifying the microcalcifications on the mammographs.

5 Applying Dispersive Flies Optimisation

In this paper, we are presenting a unique approach by deploying the recently developed DFO algorithm to detect microcalcifications on the mammographs. This approach demonstrates a promising ability to undertake this task with similar level of sensitivity. The scan used in this paper is processed by the DFO agents which are responsible for locating the affected areas.

The reproducibility and the accuracy of the DFO algorithm can be utilised in developing a standardised system to interpret bone scans and mammographs preventing operator errors and discrepancies. This technology can be employed as an adjunct to help radiologists assess the various parts of the bone scans and mammographs making the diagnosis of the lesions more thorough and less time consuming. Additionally this technique can be effectively used to develop programs for teaching and training medical students and junior doctors.

5.1 Experiments and Results

This section presents the technical details and the experiment setup, followed by the results and discussions of the performance of the algorithm.

The number of agents used in this experiment is 50,000. This figure depends on the size of the input scan (in the case of the paper the size of the scan is 500×667 pixels) and the algorithm is run for 25 iterations (i.e. 25 cycles of test and diffusion phases). The output images shown later in the paper are snapshots taken after every 5 iterations recoding the behaviour of the agents at each stage. As stated earlier, in the beginning of the process, all the agents are initialised randomly throughout the search space.

DFO is adapted here to search for areas of metastasis or calcifications in the feasible solution space. Given that the problem is a multi-objective problem, on the contrary to Eq. 3 the local neighbourhood architecture of the algorithm is implemented as shown below:

$$x_{id}^{t} = x_{nb,d}^{t-1} + \mathrm{U}(0, 1) \times (x_{nb,d}^{t-1} - x_{id}^{t-1}) \tag{9}$$

In order to evaluate the fitness of each agent, a radius (*rad*) value is specified which determine how many pixels around the pixel chosen by the agent is used to calculate the fitness of each agent. In Model I of the algorithm the radius is set to

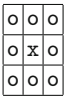


Fig. 5 *rad* = 1 for Model I. The symbol x represents the position of the agent and the o's represent the pixels used in the calculation of the fitness value of the DFO agent

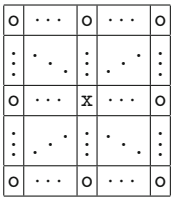


Fig. 6 *rad* = 10 for Model II. The symbol x represents the position of the agent and the o's represent the pixels used in the calculation of the fitness value of the DFO agent

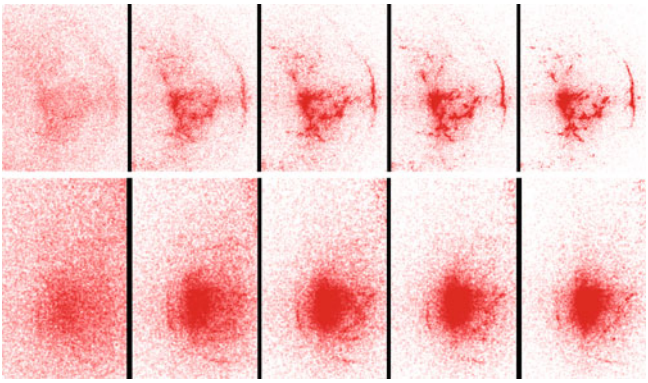


Fig. 7 *Mode I* Detecting calcifications

1, *rad* = 1 as shown in Fig. 5. In this model, the purpose is to highlight the area of calcification by allowing the DFO agents to converge on the areas of interest. In Mode II, radius is set to *rad* = 10 in order to segregate the areas that radiologists should pay particular attention. In this mode, the exact points of high calcifications are not marked but DFO agents form a border around the area of interest (Fig. 6).

As shown in Figs. 7 and 8 areas with higher potential of metastasis and calcifications are identified using Mode I and II respectively. These figures visually present the technique used, illustrating how agents congregate over the areas of interest over time (i.e. iterations) when fed with the scans as inputs of the algorithm. As the figures show, DFO agents converge to the areas of interest (as confirmed by the medical experts) throughout the entire search space.

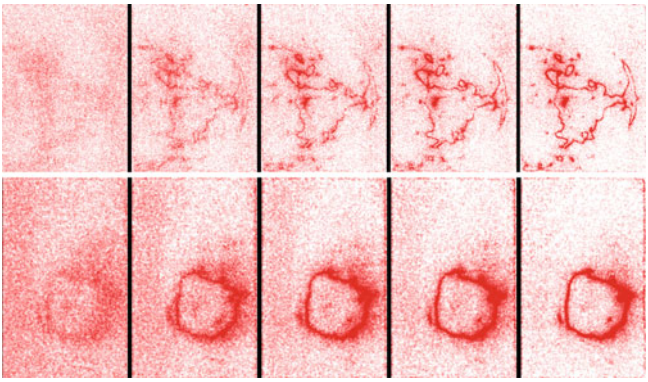


Fig. 8 *Mode II* Detecting calcifications

6 Conclusion

Dispersive Flies Optimisation (DFO), a simple numerical optimiser over continuous search spaces, is a population based stochastic algorithm, proposed to search for an optimum value in the feasible solution space; despite its simplicity, the algorithm's competitiveness over an exemplar set of benchmark functions is demonstrated. As part of the study and in an experiment, a control algorithm is proposed to investigate the behaviour of the optimiser. In this experiment, the algorithm's induced disturbance mechanism shows the ability to maintain a stable and convergence-independent diversity throughout the optimisation process. Additionally, a suitable value is recommended for the *disturbance threshold* which is the only parameter in the update equations to be optimised. This parameter controls the level of diversity by injecting a component-wise disturbance (or restart) in the flies, aiming to preserve a balance between exploration and exploitation.

In addition to diversity, DFO's performance has been investigated using three other performance measures (i.e. error, efficiency and reliability). Using these measures, it is established that the newly introduced algorithm, outperforms few generic population based algorithms (i.e. differential evolution, particle swarm optimisation and genetic algorithm) in all of the aforementioned measures over the presented benchmarks. In other words, DFO is more efficient and reliable in 84.62 and 90 % of the cases, respectively; furthermore, when there exists a statistically significant difference, DFO converges to better solutions in 71.05 % of problem set.

Additionally, this paper details the promising results of the novel application of DFO in detecting areas of interest and the identification of the potential microcalcifications on the mammographs. Two modes are proposed to further investigate the behaviour of the agents in the population and offer two representations of the outcome in order to emphasis on the area of interest and draw the attention of the clinicians in charge.

Finally, it is emphasised that the presented technique could be effectively utilised as an adjunct to the expert's eyes of a specialist.

References

1. M.M. al-Rifaie, Dispersive flies optimisation, in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, Annals of Computer Science and Information Systems, vol. 2, ed. by M. Ganzha, L. Maciaszek, M. Paprzycki. IEEE (2014), pp. 529–538. http://dx.doi.org/10.15439/2014F142
2. C. Beam, D. Sullivan, P. Layde, Effect of human variability on independent double reading in screening mammography. Acad. Radiol. **3**(11), 891–897 (1996)
3. D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in *Proceedings of the Swarm Intelligence Symposium*. (IEEE, Honolulu, 2007), pp. 120–127
4. R. Brem, J. Baum, M. Lechner, S. Kaplan, S. Souders, L. Naul, J. Hoffmeister, Improvement in sensitivity of screening mammography with computer-aided detection: a multiinstitutional trial. Am. J. Roentgenol. **181**(3), 687–693 (2003)
5. A. Burgess, On the noise variance of a digital mammography system. Med. Phys. **31**, 1987–1995 (2004)
6. E. Burnside, E. Sickles, R. Sohlich, K. Dee, Differential value of comparison with previous examinations in diagnostic versus screening mammography. Am. J. Roentgenol. **179**(5), 1173–1177 (2002)
7. D. Chakraborty, Maximum likelihood analysis of free-response receiver operating characteristic (froc) data. Med. Phys. **16**, 561 (1989)
8. M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2006)
9. D. Gehlhaar, D. Fogel, Tuning evolutionary programming for conformationally flexible molecular docking, in *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming* (1996), pp. 419–429
10. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley Longman Publishing Co., Inc., Boston, 1989)
11. J. Kennedy, R.C. Eberhart, Particle swarm optimization, in *Proceedings of the IEEE International Conference on Neural Networks*. vol. IV. (IEEE Service Center, Piscataway, 1995), pp. 1942–1948
12. C.Y. Lee, X. Yao, Evolutionary programming using mutations based on the Lévy probability distribution. IEEE Trans. Evolut. Comput. **8**(1), 1–13 (2004)
13. O. Olorunda, A.P. Engelbrecht, Measuring exploration/exploitation in particle swarms using swarm diversity, in *IEEE Congress on Evolutionary Computation. CEC 2008. (IEEE World Congress on Computational Intelligence)*. (IEEE, 2008), pp. 1128–1134
14. J. Peña, Theoretical and empirical study of particle swarms with additive stochasticity and different recombination operators, in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. GECCO'08. ACM, New York (2008), pp. 95–102, http://doi.acm.org/10.1145/1389095.1389109
15. P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, IIT Kanpur (2005)
16. J. Sumkin, D. Gur, Computer-aided detection with screening mammography: improving performance or simply shifting the operating point? Radiology **239**(3), 916–918 (2006)
17. X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999)

An Efficient Solution of the Resource Constrained Project Scheduling Problem Based on an Adaptation of the Developmental Genetic Programming

Grzegorz Pawiński and Krzysztof Sapiecha

Abstract An adaptation of the Developmental Genetic Programming (DGP) for solving an extension of the Resource-Constrained Project Scheduling Problem (RCPSP) is investigated in the paper. In DGP genotypes (the search space) and phenotypes (the solution space) are distinguished and a genotype-to-phenotype mapping (GPM) is used. Thus, genotypes are evolved without any restrictions and the whole search space is explored. RCPSP is a well-known NP-hard problem but in its original formulation it does not take into consideration initial resource workload and it minimises the makespan. We consider a variant of the problem when resources are only partially available and a deadline is given but it is the cost of the project that should be minimized. The goal of the evolution is to find a procedure constructing the best solution of the problem for which the cost of the project is minimal. The paper presents new evolution process for the DGP as well as a comparison with other genetic approaches. Experimental results showed that our approach gives significantly better results compared with other methods.

Keywords Minimisation · Project management · Scheduling · Search problems · Resource allocation · Evolutionary computations · Genetic algorithms · Developmental genetic programming

1 Introduction

The Resource-Constrained Project Scheduling Problem (RCPSP) attempts to reschedule project tasks efficiently using limited renewable resources minimising the maximal completion time of all activities [4, 6, 7]. RCPSP is an NP-complete problem which is computationally very hard [3], therefore optimal solutions for

G. Pawiński (✉) · K. Sapiecha
Department of Computer Science, Kielce University of Technology,
Kielce, Poland
e-mail: g.pawinski@tu.kielce.pl

K. Sapiecha
e-mail: krzysztof.sapiecha@gmail.com

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_12

real-life systems may be found only by using efficient heuristics. The RCPSP occurs frequently, in high scale project management such as software development, power plant building, and military industry projects such as design, development and building of nuclear submarines [31]. The authors in [25] states that it is one of the hardest problems of Operational Research.

Various RCPSP extensions have been developed for solving practical problems [15]. However, there is still free room for research. In this paper we will attack the RCPSP where resources have already got their own schedule (like in a software house). Such tasks cannot be moved. Hence, the resources are available only in particular time periods, what makes the problem computationally even more complex. The goal is to allocate resources to the project tasks, taking into consideration the availability of resources, in order to minimise the total cost of the project and complete it before a deadline. To overcome this complexity Developmental Genetic Programming (DGP) [17, 23] will be applied. It is an adaptation of Genetic Programming (GP) [21] to the optimisation problems. DGP is quite new (from 1999) but it has already been successfully applied in the design of electronic circuits, control algorithms [23], strategy algorithms in computer games, the synthesis of embedded systems [10], etc.,

To summarize, the purpose of the paper is to introduce a new evolution process to the DGP approach for solving RCPSP when resources are partially available and compare the result with other genetic approaches. Next section of the paper contains a brief description of the DGP. Related work and a motivation to the research is given in Sect. 3. Section 4 presents an idea of the adaptation of the DGP for a solution of the RCPSP extension. In Sect. 5, computational experiments to evaluate our approach and a comparison with other methods are given. The paper ends with conclusions.

2 Developmental Genetic Programming

For many problems, restrictions are imposed on how the structure of genotype may be created. GP algorithms handle the problems by constrained genetic operators in the manner, which makes them produce only legal individuals. The method achieved respectable results for the generation of efficient programs in different domains, e.g. mathematical calculations, robot control, text recognition, etc. In 36 cases, obtained results were as good as or even better than known solutions [22]. However, constrained operators create infeasible regions in the search space, also eliminating sequences of genes which may lead to high quality solutions. In the DGP approach the problem does not exists anyway. The genotype (search space) and phenotype (solution space) are distinguished and a genotype to phenotype mapping is used prior to fitness evaluation of the phenotype. Because of separating the search space from the solution space, legal as well as illegal genotypes are evolved, while each genotype is mapped onto a legal phenotype. It is worth to notice that the evolution of an illegal genotype may lead to the legal genotype constructing the expected result. Thus, the whole search space is explored. Genotypes usually are represented by trees.

Nodes of the tree are genes specifying construction functions for a solution of the problem. The edges indicate an order of execution of these functions (a procedure of construction).[1] Genotype to phenotype mapping (GPM) is performed by the execution of this procedure, starting from the root. So, if the target solution (phenotype) is a sequence of tasks with allocated resources, which is usually created by the project manager, then the construction of the solution will be a method showing how the project manager selects a resource to be allocated to each of the tasks. Therefore, the DGP does not evolve a project schedule but the project manager itself. A genotype defines how the project manager uses resource allocation strategies to create a project schedule.

3 Resource Constrained Project Scheduling Problem

Researchers' attention has been focused on making the best use of scarce resources available since PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method) developed in the late 1950s [16]. RCPSP addresses the task of allocating limited resources over time, in order to perform a set of activities subject to constraints on the order, in which the activities may be executed [18].

3.1 Classical Approach

RCPSP attempts to schedule the project tasks, efficiently using limited renewable resources, minimizing the maximal completion time of all activities $V = v_1, \ldots, v_n$. Each activity has a specific processing time p_i and it requires resources $R = r_1, \ldots, r_m$ to be processed. In general, activities may not be interrupted during their processing (non-preemption) and cannot be processed independently from each other, due to limited resource capacity and additional technological requirements. Technological requirements are represented by precedence relationships that specify a fixed processing order between pairs of activities. The finish–start relationship with zero time lags means that the activity can be started immediately after all its predecessors are completed. An example of a project plan with precedence constraints is shown on Fig. 1. The objective of the RCPSP is to find feasible completion times for all activities such that the makespan of the project is minimized, while the precedence of activities and limits of resources are not violated [17].

3.2 RCPSP Extensions

Classical RCPSP is a rather basic model with assumptions that are too restrictive for many practical applications. Consequently, various extensions of the RCPSP have

[1] A genotype in classical GAs represents a solution of the problem, while in the DGP a genotype comprises a procedure for constructing that solution.

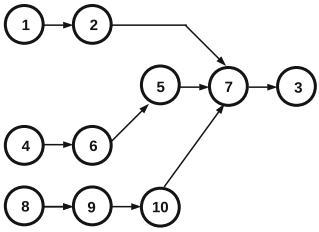


Fig. 1 Task's precedence constraints

also been developed [15, 34]. The authors outline generalizations of the activity concept, alternative precedence and resource constraints, as well as, deal with different objectives, task graph characteristic and the simultaneous consideration of multiple projects.

In the classical approach a goal of optimization is to minimize the makespan, but in many practical problems the goal is to minimize the cost i.e. to minimize the number of resources or the cost of using resources required for executing all tasks, while time constraints should be satisfied. Such problems are defined as resource investment problems [12] or resource renting problems [26]. Example of practical application of this extension is the optimization of distributed embedded systems [10], especially implemented as a network on chip architectures or based on multicore embedded processors [11]. Moreover, existing RCPSP approaches do not take into consideration initial resource workload and resources are assumed to be steadily available during an execution of the whole project. In spite of that, developers in a software house or resources of an enterprise building houses, for example, may have initial workloads when starting a new project. Such constraint better fits real-life project management problems. Dealing with more than one project is common in IT business, for example, where managers have to use a resource-sharing approach. An extension of the problem, where resources are only partially available, since they may be involved in many projects, was also investigated in [29, 30].

3.3 Solutions of the Problem

RCPSP has become a well-known standard of optimization, which has attracted numerous researchers who developed both, exact and heuristic scheduling algorithms. Among the first ones, Demeulemeester and Herroelen [6] proposed a depth-first branching scheme with dominance criteria and the bounding rules. In [4] a branching scheme which starts from a graph representing a set of conjunctions and disjunctions was used. Another method, a tree search algorithm was presented in [24]. It is based on a mathematical formulation that uses lower bounds and dominance criteria. According to [2], the optimal solution can be achieved by exact procedures only for small projects, usually containing less than 60 tasks and not highly constrained. Moreover, exact methods may require a significant amount of

computation time. Therefore heuristics have been preferred instead of exact methods due to substantial limitations of these latter ones. In-depth study of the performance of recent RCPSP heuristics can be found in Kolisch and Hartmann [19]. Heuristics described by the authors, include X-pass methods, also known as priority rule based heuristics, classical metaheuristics, such as Genetic algorithms (GAs), Tabu search (TS), Simulated annealing (SA), and Ant Colony Optimisation (ACO). They give a performance comparison of these methods as applied to different standard instances sets, generated by ProGen in the PSPLIB [20]. Two approaches of TS, for artificially created dataset instances, but based on real-world instances (got from Volvo IT and verified by experienced project manager), were investigated in [32].

Another metaheuristic algorithm, driven by a metric of the gain of optimization (MAO) [8], was also applied to the RCPSP [27]. The advantage of the algorithm is that it has a capacity of getting out of local minima. The authors adapted the algorithm to take into account specific features of human resources participating in a project schedule. The computational experiments showed significant efficiency of the approach in optimizing the RCPSP and an extension of the problem, where resources are only partially available, since they may be involved in many projects [29].

One of the latest review papers on solving RCPSP by exact methods and heuristics has been published by Deiranlou and Jolai [5], who paid particular attention to GAs. They introduced a new crossover operator and auto-tuning for adjusting the rates of crossover and mutation operators. Two approaches for solving the problem with GAs and GP are given in [13]. The authors achieved good quality results by the use of GAs. With GP, they described a methodology to evolve scheduling heuristics in a small amount of time. Yet, they state that GAs, as a technique, are inappropriate for dynamic environments and for projects with large number of activities, because of their uncertainty and amount of time required to obtain satisfactory results. The authors propose GP to find a solution of an acceptable quality within a reasonable time.

4 Adaptation of the DGP to the RCPSP

The adaptation consists in creating a tree which defines a method of allocating resources to tasks by a project manager. Then, a transformation of the tree into a target solution (a target system) is given, in order to evaluate a quality of results. Genetic operators are adapted to the tree, too.

4.1 Evolution Process

An evolution process in DGP is similar to other genetic approaches. It starts with an initial population with m individuals. Subsequently, individuals are randomly drawn from the population and are subject to operations of crossover, mutation and reproduction, such that a size of each population Π is as follows:

$$\Pi = \Psi + \Omega + \Phi = \alpha(\gamma \cdot m + \delta \cdot m + (m - \Psi - \Omega)) = \alpha m \quad (1)$$

$$\gamma + \delta \leq 1 \quad (2)$$

where
$\Psi = \alpha \cdot \gamma \cdot m$—the number of individuals created by the crossover,
$\Omega = \alpha \cdot \delta \cdot m$—the number of individuals created by the mutation,
$\Phi = \alpha(m - \Psi - \Omega)$—the number of individuals created by the reproduction,
$\gamma \in [0, 1]$—the probability of crossover,
$\delta \in [0, 1]$—the probability of mutations,
m—the number of tasks in the project,
α—a constant.

If they satisfy time constraints, they are passed to the next generation. During evolution only genotypes are evolved, while the genotype-to-phenotype mapping is used to create phenotypes. Then, the fitness of newly created genotypes is calculated. In that way, a quality of the phenotype may be evaluated in order to find procedures constructing the best solution. This iterative process is repeated over many generations until a predefined number of generations has been reached.

4.2 Genotypes and Phenotypes

In our method a genotype of the project manager is represented by a binary tree that comprises resource allocation strategies and a way of applying them for the activities. The tree edges represent a division of the list of activities into two subgroups, while nodes specify a location of the division d_i (important only for the internal nodes) and a resource allocation strategy s_i (used only by leaves) (Fig. 2). The strategy, which will be assigned to each of the subgroups is specified in an appropriate child of the node. The method (actually the project manager) uses a list of possible strategies, which might be associated to the genes, for resource assignment. These are presented in Table 1. The first two strategies, search for a resource which is the fastest or the cheapest, its load is not taken into consideration. Strategies 3 and 4 refer to tasks execution time. Task may be assigned to a resource, which will start the task as fast as possible or execute it as soon as possible, respectively. The last two strategies check, how the resource assignment and the task allocation affect current duration and current cost of the schedule that is being built.

A root of the genotype tree specifies a construction of an embryonic system, while all other nodes correspond to functions that schedule tasks, according to the

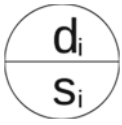


Fig. 2 Tree node
d_i—dividing point,
s_i—decision strategy

Table 1 Strategies for implementation of tasks

No.	Strategy
0	The fastest resource
1	The cheapest resource
2	The earliest start of the task
3	The earliest finish of the task
4	The smallest local duration of schedule
5	The smallest local cost of the system

assigned strategies. The initial population consists of individuals generated randomly by recursively creating nodes until a pre-established maximum tree height H is reached. An increase of the H causes doubling the divisions and hence the number of leaves of the tree. Therefore, in order to get all possible variations of strategies, the number of leaves (3) should be at least the number of activities in the project. At the beginning, the genotype is a full tree, where each node has one of the strategies assigned with the same probability and a random d_i, which is inversely proportional to H. It has to be verified whether nodes contain improper values of d_i. The dividing point cannot be bigger than the number of activities in currently considered subgroup. One of the repairing mechanisms could be a "deleting repair" that removes all children of the invalid node. The process is similar to withering of unused features in live organisms, like in intron splicing [1, 33]. However, we used a "replacing repair" that replaces the invalid node by any of its children, instead of removing the entire branch. Therefore, more genetic information will be kept in the genotype. Our tests have shown that 4685 repairs were required, for a project with 4 resources and a deadline equal to 80, out of 6000 chromosomes, which were generated during 100 generations of 30 individuals. With no repair, about 2 leaves would have been cut off each time (2.17 on average). The repair allowed to save 1466.9 leaves of the tree which is 14.5 %, where each leaf may have one of 6 different strategies.

$$Leafs_{num} = 2^{H-1} \tag{3}$$

4.3 Genotype-to-Phenotype Mapping

A GPM is used to transform the tree structure into a sequence of decision strategies, corresponding to the project activities. The sequence of strategies is obtained by traversing the tree in the depth-first order starting from the top node (the root). It corresponds to the list of leaves starting from left to right. If a node has children, a d_i is used for dividing currently considered project activities into two subgroups and strategies are assigned to them. The left child defines a strategy for the first subgroup

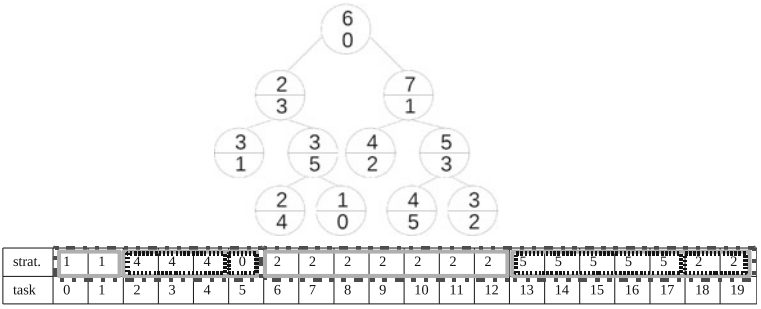


Fig. 3 Genotype with 6 decision strategies (0–5) represented by a binary tree ($H = 4$) and a sequence of decision strategies (*strat.*) after the decoding, corresponding to the project activities (*task*)

and the right child for the other. Then, subgroups of activities are passed to offspring nodes and the process is continued. Finally, as a result, we obtain a sequence of decision strategies (*strat*), assigned to the tree leaves, each corresponding to the given project activity. For the tree from Fig. 3 this works as follows:

- First, tasks are partitioned into subgroups: $\{T_0-T_5\}$ and $\{T_6-T_{19}\}$.
- Next, the first subgroup is divided into $\{T_0-T_1\}$, $\{T_2-T_5\}$, and the second group is divided into $\{T_6-T_{12}\}$, $\{T_{13}-T_{19}\}$.
- Then, the subgroup $\{T_0-T_1\}$ is associated with the leaf, hence the strategy 1 is assigned to tasks T_0 and T_1. The subgroup $\{T_2-T_5\}$ is partitioned into $\{T_2-T_4\}$ and $\{T_5\}$.
- Afterwards, the strategies 4 and 0 are assigned to tasks T_2, T_3, T_4 and T_5 respectively, and so on.

Subsequently, the decision strategies are used to construct the target schedule (phenotype), according to strategies specified by the genotype nodes. The project schedule is constructed by executing functions corresponding to the nodes. Each function takes into consideration all requirements and constraints. Thus, a genotype is always mapped onto a legal phenotype. The following steps have to be carried out, to create a task schedule:

1. search activities from the task graph, according to the precedence relationships, in order to find a list of ready-to-start tasks,
2. assign the strategies with corresponding tasks and execute the strategy to calculate a resource to allocate,
3. schedule tasks—calculate a start time for each task, based on the earliest precedence relationships and the feasible time of a resource,
4. repeat the first step, until there are unassigned tasks.

Finally, a feasible project schedule is obtained. The phenotype is used for evaluation of the quality (fitness) of the corresponding genotype. The goal of the evolution is to find the genotype giving the best result.

4.4 Fitness Function

Each individual in the population is measured in order to check the quality of the solution. A numerical value, called fitness, is calculated for the project schedule obtained after GPM. It is defined as the project cost (C), using the following equation:

$$C = \sum_{j=1}^{r} C_e(j) \cdot T_p + \sum_{j=1}^{n} \left(C_u(j) + \sum_{i=1}^{m} T(i, j) \cdot C_a(j) \right) \tag{4}$$

where $C_a(j)$—cost of task execution per time unit by the resource R_j, $C_e(j)$—the cost of the employment of the resource R_j, T_p—the project duration, $C_u(j)$—resource R_j unit cost, $T(i, j)$—safe time estimate of task i being executed by the resource R_j, n—the number of resources used in the project schedule, m—the number of tasks, r—the number of resources in the resource library. The first sum corresponds to the resource maintenance cost in the project. These are constant in execution time of the whole project no matter how high a workload of a team developing the project is. The second term occur only when a resource is being assigned to the project. These comprise a resource deployment cost and a sum of execution costs of all allocated activities. The genotypes that produce lower project cost are considered to be the better ones.

4.5 Genetic Operators

The genetic operations that are performed during the run (i.e. crossover, mutation, reproduction) are based on techniques described in [9]. A crossover is applied with the probability $\gamma \in [0, 1]$ on a randomly selected pairs of individuals. Next, the decision trees in pairs are pruned by removing a randomly selected edge. Then, subtrees are swapped between both parent genotypes. Similarly, a mutation is applied on each genotype in the population with the probability $\delta \in [0, 1]$. Afterwards, one of the following modifications, selected with the same probability, is done on the decision tree:

1. a randomly selected node is changed to another,
2. a randomly selected edge is pruned and the subtree is removed,
3. two random nodes are created and added to a randomly selected leaf.

Modifications are done only when the subtree contains more than one node. If the newly created tree is too high, it is pruned to the allowed height. Then, faulty nodes are removed to preserve the correct tree decoding. Implementation of genetic operators ensures that the correct genotype-tree structure is always kept. Moreover, they neither break precedence constraints nor produce infeasible schedules.

5 Experimental Results

Based on the above adaptation an algorithm was implemented and tested on projects from PSPLIB, developed by [20]. However, we considered an extension of RCPSP where resources have already got their own schedule, randomly allocated from a different project instance, located in the same group (so called an initial schedule). Such activities cannot be moved and therefore the resources were available only in particular time periods. Thus, our results cannot be compared with the optimal ones, because of different problem statement. So, we had to compare our results with results obtained by the use of methods which were available in the literature. The comparative study is described in Sect. 5.3. In most of the tests, the number of generations (GEN) was set to 100 and the tree height was set to 10, because for bigger values the difference of the parameters has a little effect on the results.

5.1 Test Instances

The library for RCPSP contains 2040 projects with 30, 60, 90 and 120 activities for which either optimal, best-known or lower bound solutions are given. For each problem size, a set consists of 480 instances in groups of ten, which have been systematically generated by varying three parameters: network complexity, resource factor, and resource strength. The parameters have a big impact on the hardness of the project instances [20]. The set with 30 non-dummy activities is the hardest standard set of RCPSP-instances for which all optimal solutions are currently known [6].

In our study we used project instances with 30 non-dummy activities. The renewable resources were randomly generated such that the resource unit cost $C_u(j)$ and the execution cost of activity $C_e(j)$ might vary up to 10 % from default values, which were 20 and 1 respectively. They are general purpose resources, so they might execute each activity. Values of resource parameters were set as shown in Table 2. We tested 2 randomly selected instances from each of the groups. For each project instance 10 independent runs were performed and the results were averaged.

Table 2 Values of resource parameters in the experiments

j	$C_a(j)$	$C_d(j)$
1	0.95	20
2	0.96	20
3	1.07	20
4	0.9	22
5	1.03	19
6	0.97	20
7	1.0	22
8	0.99	19
9	0.91	20
10	1.1	22

5.2 Analysis of the Evolutionary Process

5.2.1 Selection of Individuals

At first, we tested three reproduction methods: ranking, roulette-wheel and tournament selection (tournament size equals 3). In the roulette-wheel method (Fig. 4a), the cost after 100 generations was approximately the same for various probabilities of mutation and crossover. However, the best results were obtained for a small probability of crossover. The population contained randomly selected individuals with the probability proportional to their fitness and with no certainty that the best one would be reproduced. At the beginning the population was the most varied and its diversity lowered in further generations. But the best result in every subsequent generation was getting worse. The project cost was almost the same, along with the increase of the probability of both genetic operators, while in other methods it rapidly grew when the sum of the probabilities was close to 1. The reason was too small number of good individuals that were reproduced and too large number of newly created individuals in generations. In the tournament method (Fig. 4b) and ranking method (Fig. 4c) good quality results started to dominate in the population very quickly along with the increasing number of generations and therefore the project cost was decreasing. The convergence of the ranking method was slightly faster than the tournament method. In the former, individuals with high cost were quickly eliminated and the best result was found faster (in 6 generation) than in the latter (in 9 generation). Yet, the final result obtained by the tournament method was better, on average, than obtained by the ranking and therefore it was chosen for further tests. Further improvement was very slight, but it occurred till the last generation.

5.2.2 Testing Various Probabilities of Genetic Operators

Next, tests were executed in order to examine how various probabilities of crossover (Fig. 5a, b) and mutation (Fig. 5c, d) influence the algorithm performance.

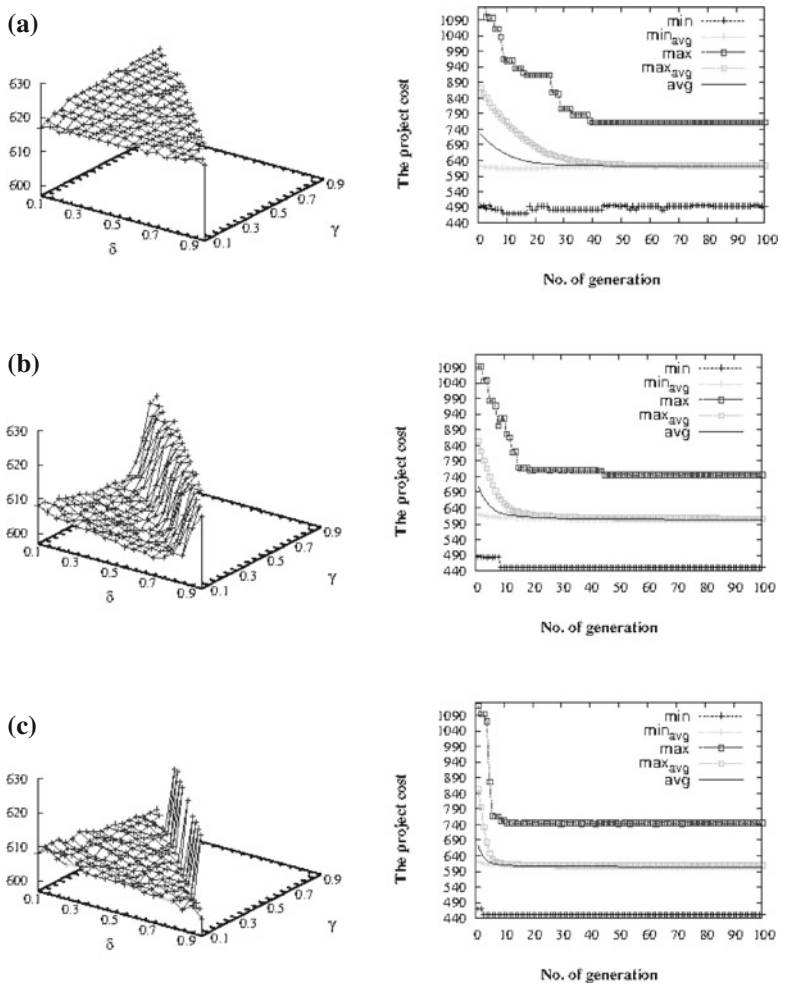


Fig. 4 The average project cost from the best individuals in each test run after 100 generations (*left column*) and the project cost in generations for $\delta = 0.05$, $\gamma = 0.65$ (*right column*), $\Pi = 90$ min_{avg}—the average project cost from the best individuals in each test run max_{avg}—the average project cost from the worst individuals in each test run, *min*—the minimal; *max*—the maximal; *avg*—the average project cost, from all individuals of a given generation and all test runs. **a** Roulette-wheel selection. **b** Tournament selection. **c** Ranking selection

The Figures show the average project cost from the best individuals in each test run. Usually, the project cost becomes lower along with increasing γ as well as increasing δ, because the operators produce more new genotypes and the population is more diverse. Thus, the chance of finding the optimal solution is bigger. However, too large number of newly created individuals makes the best ones, that were obtained in the evolution process, do not survive and the evolution itself is more random.

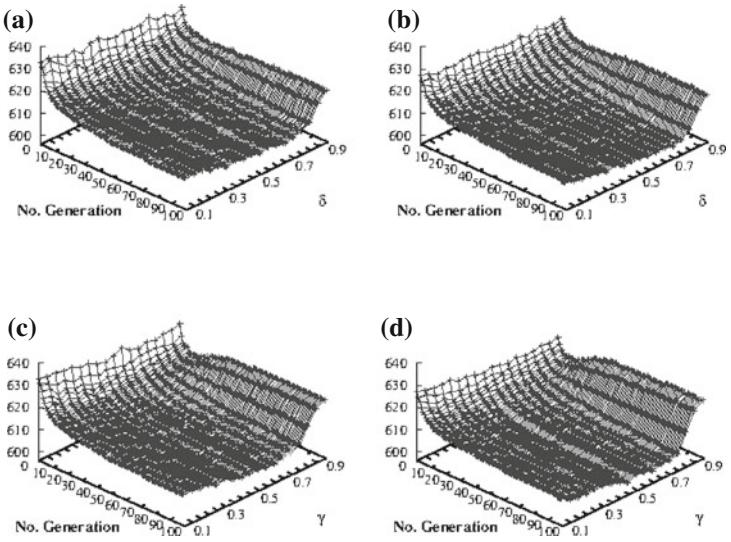


Fig. 5 The average project cost from the best individuals in each test run. **a** $\Pi = 60, \gamma = 0.05$. **b** $\Pi = 120, \gamma = 0.05$. **c** $\Pi = 60, \delta = 0.05$. **d** $\Pi = 120, \delta = 0.05$

In both probabilities of mutation and crossover the best amount of individuals that were evolved was about 70–75 %. This characteristic is even more noticeable for a greater number of individuals in the population. If $\Pi = 120$, the results were improved but for the probabilities greater than 0.7 they were even worse than for $\Pi = 60$.

5.3 Comparative Study

In order to highlight strong and weak points of our adaptation of the DGP, it was compared with a metaheuristic MAO [8], GA of Hartmann [14] and GP that use the same evolution process but vary in a way of coding the genotype. MAO is based on iterative improvements of target solution driven by a metric of the gain of optimisation, and it has the capacity of getting out of a local minimum. In [28] the MAO was adapted to take into account specific features of human resources participating in a project schedule. Their research showed high efficiency of the algorithm for resource allocation. GAs have similar evolution process but they differ in a way of coding the genotype. In GA the genotype does not have a tree structure. Genetic operators are performed directly on a sequence of resources corresponding to project activities. Mutation changes a resource on a randomly drawn position; crossover swaps two subsequences of resources, which were cut at a random position. On the other hand,

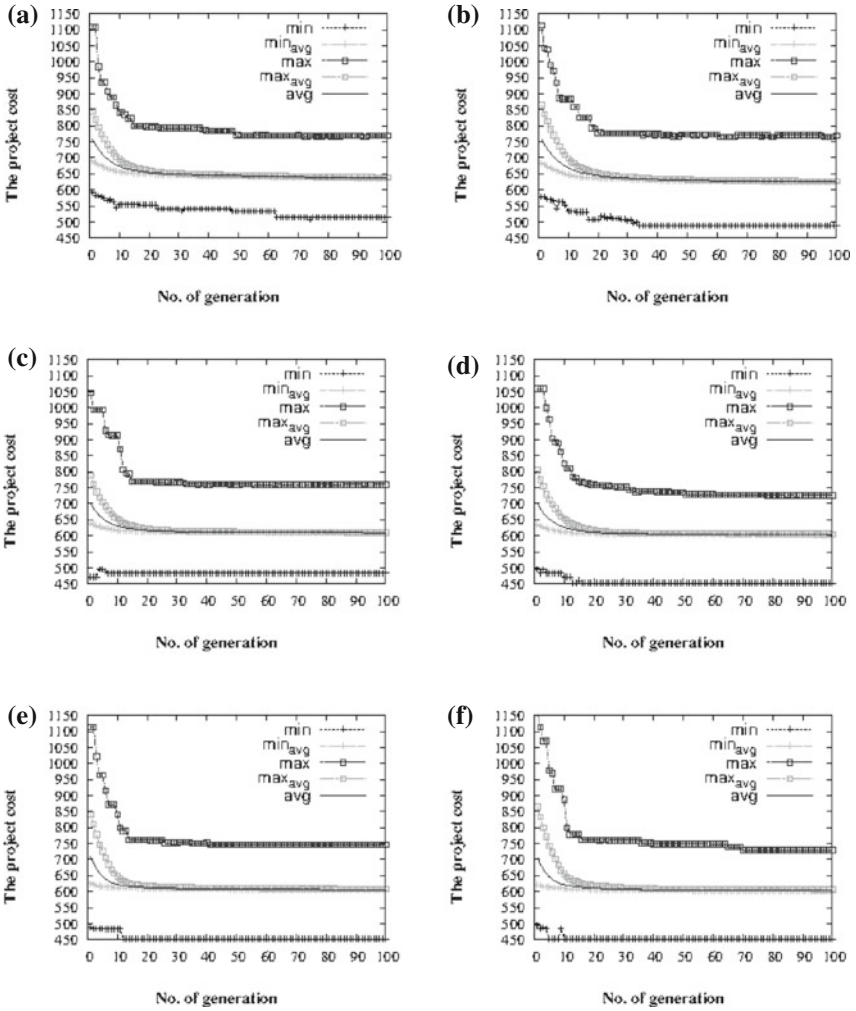


Fig. 6 Project cost in each generation for $\gamma = 0.65, \delta = 0.05$ min_{avg}—the average project cost from the best individuals in each test run max_{avg}—the average project cost from the worst individuals in each test run, *min*—the minimal; *max*—the maximal; *avg*—the average project cost, from all individuals of a given generation and all test runs. **a** $\Pi = 60$, GA. **b** $\Pi = 120$, GA. **c** $\Pi = 60$, GP. **d** $\Pi = 120$, GP. **e** $\Pi = 60$, DGP. **f** $\Pi = 120$, DGP

GP uses a genotype represented by a binary tree, but with leaves that comprise the number of resource to allocate instead of the strategy of a resource assignment.

The Fig. 6 shows a comparison of the evolution process in genetic approaches with different genotype representation. In GA, the evolution is long lasting. Genetic operations are performed randomly and without any context. In case of GP and DGP tree structures allow for preserving information about the way of constructing the result.

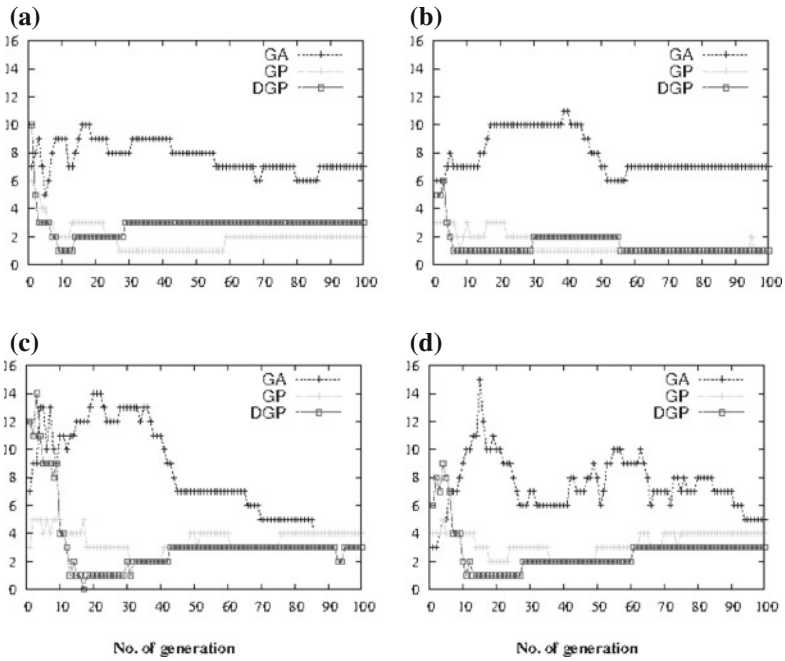


Fig. 7 The uncorrected sample standard deviation σ of the project cost in generations. **a** $\gamma = 0.3, \delta = 0.3, \Pi = 60$. **b** $\gamma = 0.3, \delta = 0.3, \Pi = 120$. **c** $\gamma = 0.15, \delta = 0.7, \Pi = 60$. **d** $\gamma = 0.15, \delta = 0.7, \Pi = 120$

Thus, convergences of GP and DGP are much faster. Moreover, the usage of these strategies causes that genotypes are of better quality from the very beginning.

The same average project cost from the best individuals in each test run was obtained after 10 generations by DGP while GP needed 20 generations. Further improvement is very slight, but it occurs till the last generation with a predominance of DGP. Both the minimal and the minimal average cost were lower with the use of DGP. This can be also observed on Fig. 7, where the uncorrected sample standard deviation of the project cost is close to 0 after 10 generations, by the use of DGP. The diversity of the population in GP is lower than in DGP, reaching its minimum after 20 generations. After that it is slightly higher but it remains on the same level.

We have also performed efficiency test on all 480 project instances where 10 project schedules were computed for each test case (Fig. 8). The results showed that after 100 generations the DGP outperformed the GA by about 2.4 % and MAO by about 7.3 % as concerns project cost reduction (Table 3). Results obtained by the use of the GP and of the DGP are similar. On the other hand DGP is about 50 % slower than GA and about 25 % slower than GP for $\Pi = 120$. However, the timing is just for illustration only, because the measurement was inaccurate due to operating system dependency. It is worth noticing that the convergence of the DGP is fast and it does not require as many generations to achieve good quality results. If we stopped the

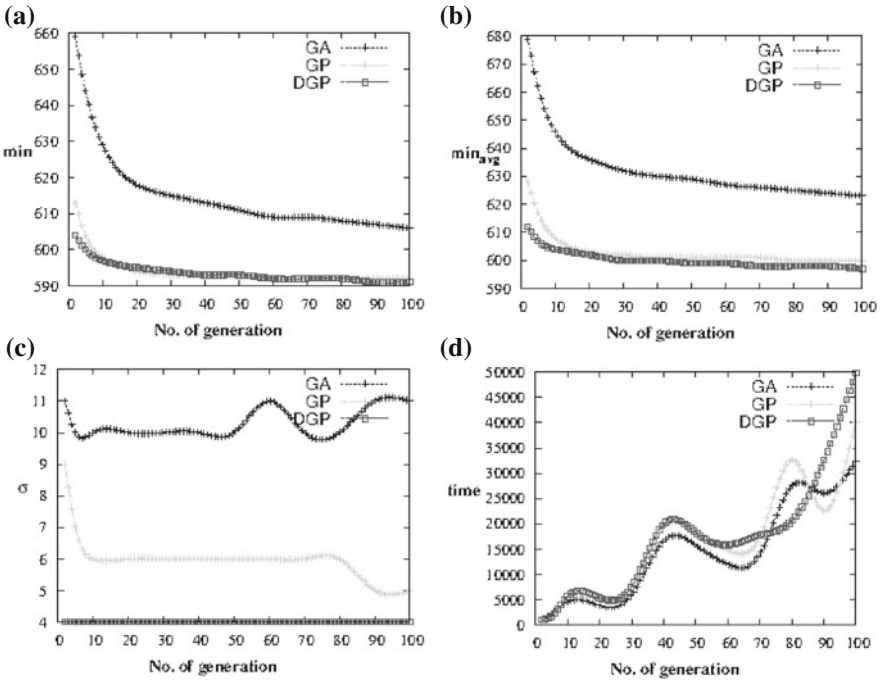


Fig. 8 Experimental results for different methods for $\gamma = 0.65$, $\delta = 0.05$, $\Pi = 120$ min_{avg}—the average project cost from the best individuals in each test run, *min*—the minimal project cost, from all individuals of a given generation and all test runs, σ—The uncorrected sample standard deviation, *time*—computation time. *GA* genetic algorithm, *GP* genetic programming, *DGP* developmental genetic programming

Table 3 Experimental results for different methods for $\gamma = 0.65$, $\delta = 0.05$, $\Pi = 120$

Scheduling method	min_C	min_T	Computation time (s)
MAO	638.25	103.02	4560
GA	606.37	94.50	32.463
GP	592.16	90.76	40.162
DGP	591.77	90.57	49.862

min_C—the minimal project cost, from all individuals of a given generation and all test runs, min_T—the minimal project time, from all individuals of a given generation and all test runs

calculations after 5 generations, it would occurred that the DGP allowed for obtaining the project cost which is lower by 7 % than the GA. Moreover, the result was better than the one obtained by the GA after 100 generations, in over 19 times greater computation time. The same result was obtained slower with the use of the GP (by 18 %). So, the main strength of the DGP is in generating quite good results much faster than other genetic approaches. Furthermore, the uncorrected sample standard

deviation of the DGP was lower than the deviation of the GA (by 55 %), and also lower than the deviation of the GP (by 21 %). Thereby DGP is better, on average. This means that it does not require as many test reruns, because every attempt is close to the best one.

6 Conclusions

The objective of this research was to introduce and evaluate a new heuristic that can efficiently solve an extension of the RCPSP. It is based on the idea of developmental genetic programming. An algorithm, which was worked out, searches for the best resource allocation strategies in a project. The method of constructing a project schedule takes the form of a decision tree that evolves, instead of evolving the solution itself. The quality of the solution is evaluated after the GPM.

The fitness function was defined as the project cost. Genetic operators specified for RCPSP were presented as well. Three reproduction methods were tested, from which the tournament method gave the best results. Then, the influence of various probabilities of mutation and crossover on the algorithm performance was evaluated. Usually, the project cost was lower along with increasing γ as well as increasing δ. The best proportion between newly created individuals in each generation, by the crossover and mutation, and individuals selected from the current population is about 3:1. The project cost also decreases as the number of generations increases. Yet, 10 generations is enough to obtain good quality results. Further reduction of the project cost may be achieved by increasing the population size.

Experimental results showed that our adaptation of DGP is efficient and may be used for solving the extension of the RCPSP. It was compared with other genetic approaches with different genotype representations. The usage of the trees accelerates the evolution. This makes genotypes are of better quality from the beginning, which accelerates the convergence. The DGP gives significantly better results than the other methods. After 100 generations it is 2.4 % better than the GA and 7.3 % better than the MAO, as concerns project cost reduction. However, the main advantage of the DGP is that it generates quite good results much faster than the other genetic approaches. Just after 5 generations it allowed for obtaining the project cost which is better than the one obtained by the GA after 100 generations, and this was done over 19 times faster.

References

1. T. Ahn, S. Erenguc, The resource constrained project scheduling problem with multiple crashable modes: a heuristic procedure. Eur. J. Oper. Res. **107**(2), 250–259 (1998). http://linkinghub.elsevier.com/retrieve/pii/S0377221797003317
2. J. Alcaraz, C. Maroto, A robust genetic algorithm for resource allocation in project scheduling. Ann. Oper. Res. **102**, 83–109 (2001)

3. J. Blazewicz, J.K. Lenstra, A.H.G.R. Kan, Scheduling subject to resource constraints: classification and complexity. Discret. Appl. Math. **5**, 11–24 (1983)
4. P. Brucker, S. Knust, A. Schoo, O. Thiele, A branch-and-bound algorithm for the resource-constrained project scheduling problem. Eur. J. Oper. Res. **107**, 272–288 (1998)
5. M. Deiranlou, F. Jolai, A new efficient genetic algorithm for project scheduling under resource constrains. World Appl. Sci. J. **7**(8), 987–997 (2009)
6. E.L. Demeulemeester, W.S. Herroelen, New benchmark results for the resource-constrained project scheduling problem. Manag. Sci. **43**, 1485–1492 (1997)
7. E.L. Demeulemeester, W.S. Herroelen, *Project scheduling—a research handbook*, International Series in Operations Research, Management Science (Kluwer Academic Publishers, Boston, 2002)
8. S. Deniziak, Cost-efficient synthesis of multiprocessor heterogeneous systems. Control Cybern. **33**, 341–355 (2004)
9. S. Deniziak, K. Wieczorek, Evolutionary optimization of decomposition strategies for logical functions, in *Proceedings of 11th International Conference on Artificial Intelligence and Soft Computing*, vol. 7269, Lecture Notes in Computer Science (2012), pp. 182–189
10. S. Deniziak, A. Gorski, Hardware/software co-synthesis of distributed embedded systems using genetic programming, *Evolvable Systems: From Biology to Hardware* (Springer, New York, 2008), pp. 83–93
11. S. Deniziak, L. Ciopiński, G. Pawiński, K. Wieczorek, Cost optimization of real-time cloud applications using developmental genetic programming, in *IEEE/ACM 7th International Conference on Utility and Cloud Computing*. IEEE Computer Society (2014), pp. 182–189
12. A. Drexl, A. Kimms, Optimization guided lower and upper bounds for the resource investment problem. J. Oper. Res. Soc. **52**, 340–351 (2001)
13. T. Frankola, M. Golub, D. Jakobovic, Evolutionary algorithms for the resource constrained scheduling problem, in *Proceedings of 30th International Conference on Information Technology Interfaces*, vol. 7269, Information Technology Interfaces (2008), pp. 715–722
14. S. Hartmann, A competitive genetic algorithm for resource-constrained project scheduling. Nav. Res. Logist. **45**, 733–750 (1998)
15. S. Hartmann, D. Briskorn, Survey of variants and extensions of the resource-constrained project scheduling problem. Eur. J. Oper. Res. **207**, 1–15 (2010)
16. C. Hendrickson, T. Au, Project management for construction: fundamental concepts for owners, engineers, architects, and builders, Chris Hendrickson (2008)
17. R.E. Keller, W. Banzhaf, The evolution of genetic code in genetic programing, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999)*. Information Technology Interfaces (1999), pp. 1077–1082
18. R. Klein, *Resource-constrained scheduling problems* (Springer, Berlin, 2000)
19. R. Kolisch, S. Hartmann, Experimental investigation of heuristics for resource-constrained project scheduling: An update. Eur. J. Oper. Res. **174**, 23–37 (2006)
20. R. Kolish, A. Sprecher, Psplib—a project scheduling library. Eur. J. Oper. Res. **96**, 205–216 (1996)
21. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, MA, USA, 1992)
22. J.R. Koza, R. Poli, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (Springer, New York, 2005)
23. J. Koza, M.A. Keane, M.J. Streeter, W. Mydlowec, J. Yu, G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence* (Kluwer Academic Publishers, Boston, 2003)
24. A. Mingozzi, V. Maniezzo, S. Ricciardelli, L. Bianco, An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Manag. Sci. **44**, 714–729 (1998)
25. R.H. Möhring, A.S. Shulz, F. Stork, M. Utez, Solving project scheduling problems by minimum cut computations. Manag. Sci. **49**(3), 330–350 (2003)

26. H. Nübel, The resource renting problem subject to temporal constraints. OR-Spektrum **23**(3), 359–381 (2001)
27. G. Pawiński, K. Sapiecha, Resource allocation optimization in critical chain method. Ann. Universitatis Mariae Curie-Sklodowska sectio Informaticales **12**(1), 17–29 (2012)
28. G. Pawiński, K. Sapiecha, Cost-efficient project management based on distributed processing model, in *Proceedings of the 21th International Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2013)*, IEEE Computer Society (2013), pp. 157–163
29. G. Pawiński, K. Sapiecha, Cost-efficient project management based on critical chain method with partial availability of resources. Control Cybern. **43**(1), 1485–1492 (2014)
30. G. Pawiński, K. Sapiecha, A developmental genetic approach to the cost/time trade-off in resource constrained project scheduling, in *2014 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE (2014), pp. 171–179
31. M. Pinedo, X. Chao, *Operations Scheduling with applications in Manufacturing*, 2nd edn. (Irwin/McGraw-Hill, Boston, 1999)
32. M.E. Skowronski, P.B. Myszkowski, M. Adamski, P. Kwiatek, Tabu search approach for multi-skill resource-constrained project scheduling problem, *Federated Conference on Computer Science and Information Systems (FedCSIS 2013)*, IEEE (2013), pp. 153–158
33. J.D. Watson, N.H. Hopkins, J.W. Roberts, J.A. Steitz, A.M. Weiner, *Molecular Biology of the Gene* (Benjamin Cummings, Menlo Park, 1992)
34. J. Weglarz, J. Józefowska, M. Mika, G. Waligóra, Project scheduling with finite or infinite number of activity processing modes-a survey. Eur. J. Oper. Res. **208**(3), 177–205 (2011)

Bayesian-Based Approach to Application of the Genetic Algorithm to Localize the Abrupt Atmospheric Contamination Source

A. Wawrzynczak, M. Jaroszynski and M. Borysiewicz

Abstract We apply the Bayesian inference in combination with the Genetic algorithm (GA) to the problem of the atmospheric contaminant source localization. The algorithm input data are the on-line incoming concentrations of released substance registered by sensors network. The proposed reconstruction algorithm is firstly adjusted and tested based on the data from the synthetic experiment. The proposed GA scan 5-dimensional parameters space searching for the contaminant source coordinates (x,y), release strength (Q) and the atmospheric transport dispersion coefficients. Based on the performed tests the most efficient GA configuration is identified. To speed up the algorithm the dynamical termination criteria, founded on the probabilistic requirements regarding the searched parameters value, is proposed. Then, we apply developed algorithm to localize the release source utilizing data from the field tracer experiment conducted in May 2001 at the Kori nuclear site. We demonstrate successful localization of the continuous contamination source in very complicated hilly terrain surrounding the Kori nuclear site. Results indicate the probability of a source to occur at a particular location with a particular release strength.

1 Introduction

Accidental atmospheric releases of harmful material pose high risks to human health and the environment. In the event of an atmospheric release of chemical, but also radioactive, biological materials, emergency responders need quickly to predict the

A. Wawrzynczak (✉) · M. Borysiewicz
National Centre for Nuclear Research, Świerk-otwock, Poland
e-mail: a.wawrzynczak@ncbj.gov.pl

M. Borysiewicz
e-mail: manhaz@ncbj.gov.pl

A. Wawrzynczak
Institute of Computer Sciences, Siedlce Univesity, Siedlce, Poland

M. Jaroszynski
Siedlce Univesity, Siedlce, Poland
e-mail: marcinjaro89@gmail.com

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_13

current and future locations and concentrations of the substance in the atmosphere. Therefore, it is valuable to develop the emergency system, which based on the concentrations of a dangerous substance can estimate the probable location of the release source. Moreover, the location of the contamination source should be found as soon as possible. The most obvious way is to propose the simulation that gives the same substance point concentrations as registered by the sensors. However, to create the model realistically imitating the real situation based only on the sparse point-concentrations is not trivial. This task requires the specification of the set of model parameters. The event reconstruction problem can be reformulated into a task of sampling an ensemble of simulations, guided by comparisons with data.

A comprehensive literature review of past works on solutions of the inverse problem for atmospheric contaminant releases can be found in (e.g. [1]). The problem of the source term estimation was studied in the literature grounded both on the deterministic and probabilistic approach. In [2] was implemented an algorithm based on integrating the adjoint of a linear dispersion model backward in time to solve a reconstruction problem. In [3] were introduced dynamic Bayesian modeling, and the Markov Chain Monte Carlo (MCMC) sampling approaches to reconstruct a contaminant source. The effectiveness of MCMC in the localization of the atmospheric contamination source based on the synthetic experiment data was presented in [4, 5]. The advantage of the Sequential Monte Carlo over the MCMC in the estimation of the probable values of the source coordinates was presented in [6].

The problem of finding the 'best fitted' model parameters, for which a forward atmospheric dispersion model output will reach agreement with real observations, can be considered as the optimization problem. Metaheuristics, such as genetic algorithms (GA), are broadly used to solve various optimization problems. The concept of GA was to use the power of evolution to create a stable and universal tool reliable of solving optimization problems [7, 8]. Since GA introduction and propagation, the GA have been often used as an alternative to the conventional optimization methods and has been successfully applied in a variety of areas. For example it was used in control engineering [9], finding hardware bugs [10] and much more e.g. [11]. The GA has been also used in environmental sciences problem e.g. in the addressing air quality problem [12].

Application of the metaheuristic like GA requires defining the values of several algorithm components and parameters. These parameters have a large impact on performance and efficiency of the algorithm (e.g., [13–15]). Therefore, it is important to estimate the parameters of the algorithm best suitable for the considered optimization problem. The optimal values for the parameters depend mainly on: (a) the problem; (b) the domain of the problem to deal with; and (c) computational time that can be spent on solving the problem. Usually, in the algorithm parameters tuning a compromise between solution quality and computational time should be achieved.

In this paper, we apply the GA together with the Bayesian approach to the problem of localizing the abrupt atmospheric contamination source based on point-concentrations reported by the sensors network. Using the synthetic experiment data we found the efficient GA configuration best suitable for the algorithm performance. Then we run the proposed algorithm for the real field tracer experiment data.

1.1 Synthetic Data

Our primary goal is to propose the efficient algorithm being able to conduct dynamic inference of an unknown atmospheric release. To test the proposed method we require some concentration data. To satisfy this requirement we have performed the simulation with the use of the atmospheric dispersion second-order Closure Integrated PUFF model (SCIPUFF) [16]. SCIPUFF is an ensemble mean dispersion model designed to compute the time-dependent field of expected concentrations resulting from one or more sources. The model solves the transport equations using a second-order closure scheme and treats releases as a collection of Gaussian puffs.

In simulation, we assumed that we have 10 sensors distributed over 15 km × 15 km area, the location of sensors was chosen randomly within the domain (Fig. 1). The atmospheric contamination source was located at $x = 3$ km, $y = 8$ km, $H = 25$ m within the domain. The simulated release was continuous with rate $Q = 8000$ g/s and started one hour before first sensors measurements. The wind was directed along x axis with speed 5 m/s. Further, in this paper we assume that the algorithm input

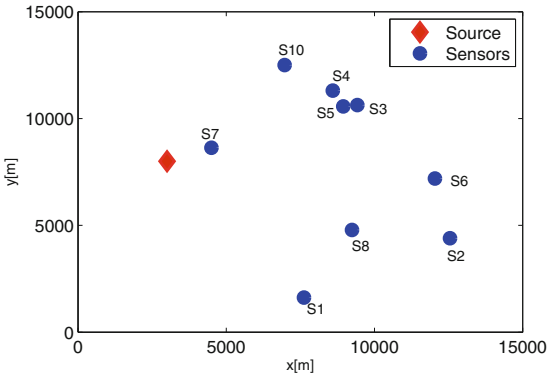


Fig. 1 Distribution of the sensors and the release source within the considered domain

Table 1 Concentrations [g/m^3] reported by sensors in subsequent time intervals

Sensor	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6
S1	0	0	0	0	0	0
S2	0	3.62E-09	4.93E-09	6.98E-09	4.15E-09	6.65E-09
S3	9.15E-09	2.88E-08	1.97E-08	1.88E-08	1.69E-08	1.62E-08
S4	3.83E-12	1.77E-11	4.89E-12	6.53E-12	2.31E-12	7.77E-12
S5	1.14E-08	1.83E-08	1.25E-08	1.20E-08	1.10E-08	1.03E-08
S6	2.91E-06	4.85E-04	4.77E-04	4.71E-04	4.43E-04	4.49E-04
S7	3.28E-05	3.27E-05	3.21E-05	3.13E-05	3.01E-05	2.87E-05
S8	2.29E-11	2.15E-10	1.05E-10	1.17E-10	7.56E-11	1.14E-10
S9	0	0	0	0	0	0
S10	0	0	0	0	0	0

information are reported every 15 min (in subsequent time steps) during 1.5 h concentrations of the dispersed substance registered by 10 sensors (Table 1). We run algorithm localizing the source of contamination just after the first recording from sensors (t = 1) and update the obtained probabilities with use of the developed algorithm by subsequent sensors data.

2 The Reconstruction Procedure

2.1 Bayesian Inference

The Bayes' theorem, as applied to an emergency release problem, can be stated as follows:

$$P(M|D) \propto P(D|M)P(M) \tag{1}$$

where M represents possible model configurations or parameters and D are observed data. For our application, Bayes' theorem describes the conditional probability $P(M|D)$ of certain source parameters (model configuration M) given observed measurements of concentration at sensor locations (D). This conditional probability $P(M|D)$ is also known as *a posteriori* distribution and is related to the probability of the data conforming to a given model configuration $P(D|M)$, and to the possible model configurations $P(M)$, before taking into account the measurements. The probability $P(D|M)$, for fixed D, is called the *likelihood* function, while $P(M)$-*a priori* distribution [17]. To estimate the unknown source parameters M using formula (1), the posteriori distribution $P(M|D)$ must be sampled. $P(D|M)$ quantifies the likelihood of a set of measurements D given the source parameters M.

Value of likelihood for a sample is computed by running a forward dispersion model with the given source parameters M and comparison of the model predicted concentrations in the points of sensors location (within a considered domain) with actual observations D. The closer the predicted values are to the measured ones, the higher is the likelihood of the sampled source parameters. As the sampling procedure we use an GA to obtain the posterior distribution $P(M|D)$ of the source term parameters given the concentration at sensor locations. This way we replace the Bayesian formulation with a sampling procedure to explore the model parameters space and to obtain a probability distribution for the contamination source parameters.

A measure indicating the quality of the current GA population is expressed in terms of a *likelihood function*. This function compares the predicted from the model and observed data at the sensor locations as:

$$\lambda(M) = -\sum_{i=1}^{N}[log(C_i^M) - log(C_i^E)]^2, \tag{2}$$

where λ is the likelihood function, C_i^M are the predicted by the forward model concentrations at the sensor locations i, C_i^E are the sensor measurements and N is the number of sensors.

The *posterior probability distribution* (1) is computed directly from the resulting GA generations and is estimated as:

$$P(M|D) = \frac{1}{n}\sum_{i=1}^{n}\delta(M_i - M), \tag{3}$$

which represents the probability of a particular model configuration M giving results that match the observations at sensor locations. Equation (3) is a sum over the entire GA generation. Thus $\delta(M_i - M) = 1$ when $M_i = M$, and 0 otherwise. If in the generation many chromosomes have the same configuration $P(M|D)$ increases through the summation increasing the probability for those contamination source parameters.

2.2 Forward Dispersion Model

A forward model is needed to calculate the concentration C_i^M at the points i of sensor locations for the tested set of model parameters M at each GA step. As a testing forward model we selected the fast-running Gaussian plume dispersion model (e.g. [18]). The Gaussian plume dispersion model for uniform steady wind conditions can be written as follows:

$$C(\widetilde{x},\widetilde{y},z) = \frac{Q}{2\pi\sigma_y\sigma_z U} exp\left[-\frac{1}{2}\left(\frac{\widetilde{y}}{\sigma_y}\right)^2\right] \times \left\{exp\left[-\frac{1}{2}\left(\frac{z-H}{\sigma_z}\right)^2\right] + exp\left[-\frac{1}{2}\left(\frac{z+H}{\sigma_z}\right)^2\right]\right\} \tag{4}$$

where $C(\widetilde{x},\widetilde{y},z)$ is the concentration of the emission (in micrograms per cubic meter) at any point $\widetilde{x}$ meters downwind of the source, $\widetilde{y}$ meters laterally from the centerline of the plume, and z meters above ground level, U is the wind speed directed along x axis, Q is the emission rate or the source strength and H is the effective height of the release equal to the sum of the release height and plume rise. In the Eq. (4) σ_y and σ_z are the standard deviation of concentration distribution in the crosswind and vertical direction. These two parameters were defined empirically for different stability conditions [19, 20]. In this case we restrict the diffusion to the stability class C (Pasquill type stability for rural area). In scanning algorithm we assumed that we do not know exact behavior of the plume and consider those coefficients as unknown. Thus, the parameters σ_y and σ_z are taken as: $\sigma_y = z_1 \cdot \widetilde{x} \cdot (1 + \widetilde{x} \cdot 4 \cdot 10^{-5})^{-0.5}$, $\sigma_z = z_2 \cdot \widetilde{x}$ where values z_1 and z_2 are sampled by algorithm within interval [0.001, 0.35]. The simple mathematical transformation of the coordinate system is required to apply formula (4) to search for the contamination source position (x, y) within the domain 15 km × 15 km (Fig. 1).

2.3 Genetic Algorithm

The localization of the contamination source within the predefined domain requires the recognition of the atmospheric dispersion model parameters for which the model output at the sensors location meets the real data. In this context, we can say that the problem can be seen as the optimization problem for which GA can be applied. Figure 2 presents the concept of GA application in the Bayesian estimation of the unknown model parameters. The algorithm starts with the defining the initial population. The population is composed of the predefined number of chromosomes, $P(\tau) = x_1^\tau, \ldots, x_n^\tau$, for the generation τ, being initially randomly drawn from the admissible set of values. This set is explicitly defined by the space of explored parameters. GA chromosome is configured as a binary value representing the real value of searched parameters. The quality of each chromosome in the current population is evaluated based on the cost, or objective/likelihood function. Various objective functions can be applied; its form depends upon the problem being solved. We use the function presented by Eq. (2). The application of the genetic operators 'improves' the current population.

Figure 2 presents the sequence of applied genetic operators:

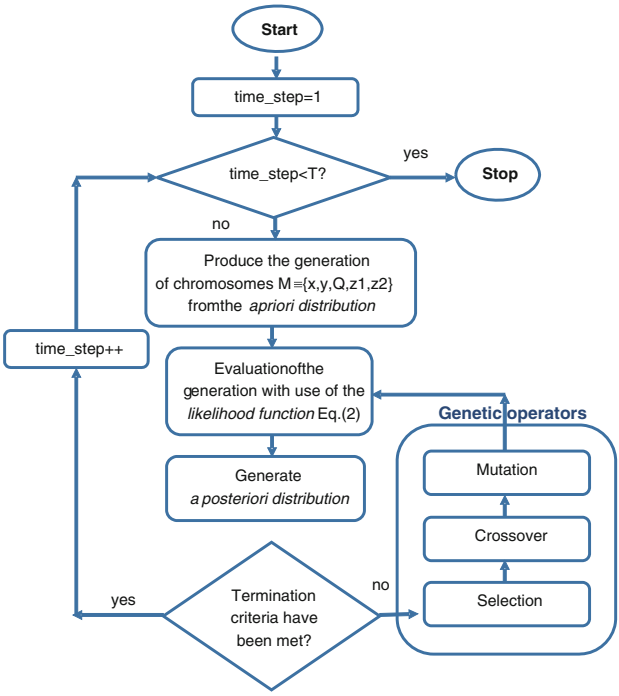


Fig. 2 Flow chart of the stochastic reconstruction procedure

1. **Selection**. Information on the quality of population's chromosomes is used to perform a selection. The portion of the population that is replaced in each generation is done based on the likelihood function (Eq. 2) value obtained during the evaluation of the population (various in each algorithm iteration). There are many ways of dealing with GA selection e.g. roulette selection, rank selection, hard and soft tournament. For the problem presented the all mentioned methods were tested. In this paper, we present the results of selection based on rank and hard tournament selection. Results obtained applying these two selections are compared further in this paper. In the rank selection, the better likelihood function results in the lower rank value leading to higher probability of being drawn to the next population. Pseudo code presents Algorithm 1. In the case of hard tournament selection of size 2, as the result of the tournament from each pair of the selected chromosomes one with the better objective function value passes to the next population. Pseudo code presents Algorithm 2.
2. **Crossover**. Crossover is process of replacing parents with their children in the current population. Children are created by blending of the parents at the randomly chosen crossover point. The crossover probability determines the number of crossovers that occurs within the population. Similarly to the previous operator there are many ways of dealing with GA crossover e.g. single point crossover, multi-point crossover, uniform crossover, arithmetic crossover. For a given problem, the best results were achieved applying the multi-point crossover. Procedure begins with performing, for each chromosome, the test for being a parent according to the crossover probability CP. From the parents' population the unexploited pair is chosen, then one crossover point for each parameter encoded in the chromosome is drawn, i.e. five points for the problem presented. Parents are split at the crossover points for each encoded parameter, then (in term of each encoded parameter) bits are swap resulting in two children. Pseudo code presents Algorithm 3.
3. **Mutation**. It changes the chromosome's features. By giving a chance of changing chromosome's individual bits mutation allows the algorithm to search for the entire solution's space and not to converge to local extremes. The mutation probability determines the number of mutations that occurs. The most frequently used are uniform mutation and not-uniform mutation. For the given problem the best results were achieved with uniform mutation in which all chromosome's bits are mutated with the mutation probability MP. Pseudo code presents Algorithm 4.

After performing the selection, crossover and mutation, the new generation $(\tau + 1)$, being subject to the new evaluation, is established. After some number of generations the algorithm converges—it is expected that the best chromosome represents a near-optimum (reasonable) solution. The process stops when the termination criterion is fulfilled. The most common termination criterion is a limited number of generations, but in this paper we present another possibility.

In the presented in Sect. 3 tests the scanned parameters space M is five-dimensional i.e. $M \equiv \{x, y, Q, z1, z2\}$. Correspondingly each chromosome $M(i)$ stores the following information:

Algorithm 1 Rank Selection

```
ascSortMByLikelihoodFunction();
MProbabilityRange = 0;
FOR i=1 to N LOOP %N–population size
    M(i).rank = i-1; %M–chromosome
     probability = 2*(N-M(i).rank)/N*(N+1);
     MProbabilityRange += probability;
    M(i).probability = MProbabilityRange;
END LOOP
FOR i=1 to N LOOP
     randVal = drawNumberFrom0To1();
     FOR j=1 to N LOOP
        IF M(j).probability >= randVal
            newPopulation(i) = M(j);
            break;
        END IF
     END LOOP
END LOOP
```

Algorithm 2 Hard tournament selection

```
FOR i=1 to N LOOP
  FOR j=1 to TS LOOP
          tournamentGroup(j)=
                  =drawChromosomeFromPopulation();
  END LOOP
  sortTournamentGroupByLikelihoodFunction();
  newPopulation(i) = getBestTournamentChromosome();
END LOOP
```

- x, y—coordinates of contamination's source in [m],
- Q—strength of release in [g/s],
- $z1$, $z2$—terms in the turbulent diffusion parametrization.

The parameters M are searched within the intervals $x \in \langle 0, 15000\rangle$, $y \in \langle 0, 15000\rangle$, $Q \in \langle 1, 8000\rangle$, $z1 \in \langle 0.001, 0.350\rangle$ and $z2 \in \langle 0.001, 0.350\rangle$. The precision P for parameters x,y equals $P_{x,y} = 1\,(\text{m})$, for Q: $P_Q = 1\,(\text{g/s})$, and $P_{z1} = P_{z2} = 0.001$. The example of the encoded chromosome presents Fig. 3.

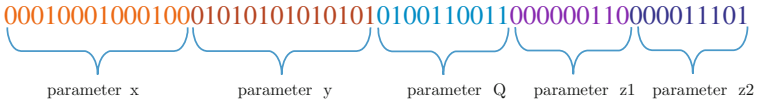


Fig. 3 Example of the chromosome representing the searched model parameters

Algorithm 3 Multi-point Crossover.

```
FOR i=1 to N LOOP %N–population size
    IF drawNumberFrom0To1() <= CP
      currentPopulation(i).isParrent(true);
    END IF
END LOOP

WHILE existsTwoNotUsedParents() LOOP
   firstParent  = popParent();
   secondParent = popParent();

   xCrossPoint = drawNumberFrom0ToParameterXLength();
   yCrossPoint = drawNumberFrom0ToParameterYLength();
   qCrossPoint = drawNumberFrom0ToParameterQLength();
   z1CrossPoint= drawNumberFrom0ToParameterZ1Length();
   z2CrossPoint= drawNumberFrom0ToParameterZ2Length();

   tmpXBin1 = firstParent.getXParameterBinaryForm();
   tmpYBin1 = firstParent.getYParameterBinaryForm();
   tmpQBin1 = firstParent.getQParameterBinaryForm();
   tmpZ1Bin1= firstParent.getZ1ParameterBinaryForm();
   tmpZ2Bin1= firstParent.getZ2ParameterBinaryForm();

   tmpXBin2 = secondParent.getXParameterBinaryForm();
   tmpYBin2 = secondParent.getYParameterBinaryForm();
   tmpQBin2 = secondParent.getQParameterBinaryForm();
   tmpZ1Bin2= secondParent.getZ1ParameterBinaryForm();
   tmpZ2Bin2= secondParent.getZ2ParameterBinaryForm();

   firstChildX  = tmpXBin1(0,CrossPoint)+
                  + tmpXBin2(CrossPoint+1);
   firstChildY  = tmpYBin1(0,CrossPoint)
                + tmpYBin2(CrossPoint+1);
   firstChildQ  = tmpQBin1(0,CrossPoint)+
                + tmpQBin2(CrossPoint+1);
   firstChildZ1 = tmpZ1Bin1(0,CrossPoint)+
                + tmpZ1Bin2(CrossPoint+1);
   firstChildZ2 = tmpZ2Bin1(0,CrossPoint)+
                + tmpZ2Bin2(CrossPoint+1);

   secondChildX = tmpXBin2(0,CrossPoint)+
                + tmpXBin1(CrossPoint+1);
   secondChildY = tmpYBin2(0,CrossPoint)+
                + tmpYBin1(CrossPoint+1);
   secondChildQ = tmpQBin2(0,CrossPoint)+
                + tmpQBin1(CrossPoint+1);
   secondChildZ1= tmpZ1Bin2(0,CrossPoint)+
                + tmpZ1Bin1(CrossPoint+1);
   secondChildZ2= tmpZ2Bin2(0,CrossPoint)+
                + tmpZ2Bin1(CrossPoint+1);

   firstChild = firstChildX+firstChildY+firstChildQ
              + firstChildZ1+firstChildZ2;
   secondChild= secondChildX+secondChildY+secondChildQ
              + secondChildZ1+secondChildZ2;

   currentPopulation(firstParent.getId())=firstChild;
   currentPopulation(secondParent.getId())=secondChild;
END LOOP
```

Algorithm 4 Uniform Mutation

```
FOR i=1 to N LOOP   %N-population size
   FOR j=1 to L LOOP %L-length of chromosome
                                    %binary form
      IF drawNumberFrom0To1() <= MP
         currentPopulation(i).swapBitValue(j);
      END IF
   END LOOP
END LOOP
```

In the reconstruction of the atmospheric contamination source the following GA configuration was applied:

- Size of population N = 150;
- Selection:
  - rank selection,
  - hard tournament of size 2;
- Multi-point crossover with probability $CP = 0.75$, with 5 crossover points (5 is a number of searched parameters);
- Uniform mutation with probability $MP = 0.02$.

The size of population, crossover probability and mutation probability were selected based on the numerical tests presented in [21].

3 Reconstruction of the Synthetic Experiment

We assume that the concentrations from the sensors arrives subsequently in six-time steps (Table 1). We start to search for the source location (x, y), release rate (Q) and parameters z_1 and z_2 after first sensors measurements. Thus, reconstruction algorithm is run with obtaining the first measurements from the sensors ($t = 1$ at Table 1). We assume that initially we have no *a priori* information about the parameters values. So, the initial value of each parameter is draw randomly from the predefined interval with the use of the uniform distribution. Then generation is evaluated with the utilization of the likelihood function (Eq. 2). The subsequent generations are iteratively updated by the genetic operators until the stop criterion is met. Of course, there arises the question how to specify the termination criteria? The usual criterion applied in GA is fixed number of generations. For the problem defined in this paper the time of giving the answer is crucial, so the constant number of generations is not optimal. In the estimation of the source of the atmospheric contamination, the most important is to assess its location. Thus, crucial is assessment of the x and y coordinates of the source. Applying the Bayesian approach we can ask what probability of estimation of these parameters will be acceptable. So, after applying the

last genetic operator, i.e. mutation, the histograms of x and y parameters encoded in the current chromosomes generation are evaluated. If many chromosomes have the same parameters configuration the probability of individual value of parameter increases. Consequently, the reconstruction algorithm is terminated when values of parameters x , y will be obtained with probability greater than 0.8 and Q with probability greater than 0.7. If this condition is fulfilled, the *posterior* distributions of all parameters are calculated. Obtained *posterior* distributions are considered as the *prior* distributions in the subsequent time step. Consequently, in the next time step, when new data from the sensors arrive the initial population is drawn uniformly from the *prior* distribution i.e. *posterior* distribution from the previous time step [22].

We have run the reconstruction applying two types of the selection i.e. the rank and the hard tournament selection. The number of generations required to fulfill the dynamical termination criterion for both selections is presented in Table 2. Comparison makes obvious that the rank selection is much more efficient, precisely it requires ten times fewer generations than the algorithm with the hard tournament selection. Thus, in the practical application is should be preferred. Figure 4 illustrates the distribution of the estimated by the GA contamination source coordinates x and y in subsequent generations in the first-time step. It is seen that in the 1st generation the chromosomes are equally distributed within the scanned domain. However, the applied genetic operators improve population quality in further generations and the chromosomes gradually focus on the actual source location. Finally, for 19th generation the estimated by the GA contamination source location approaches the target site. Figures 5 and 6 present the posterior distributions for x, y and Q parameters obtained in the succeeding time steps. These distributions were obtained based on the chromosomes configurations in the last generation at given reconstruction algorithm iteration. Based on the searched parameters value, encoded in the final chromosomes population, the histogram for each parameter has been assessed. Obtained histograms

Table 2 Number of generations used in the reconstruction algorithm with the dynamic termination criteria for the hard tournament of size 2 selection and with the rank selection, $CP = 0.75$ and $MP = 0.02$

Time step	Hard tournament selection		Rank selection	
	Generation's number	Forward dispersion model's runs	Generation's number	Forward dispersion model's runs
t = 1	163	24 300	20	3 000
t = 2	71	10 650	10	1 500
t = 3	136	20 400	9	1 350
t = 4	125	18 750	10	1 500
t = 5	124	18 600	12	1 800
t = 6	99	14 850	13	1 950
Summary	717	107 550	74	11 100

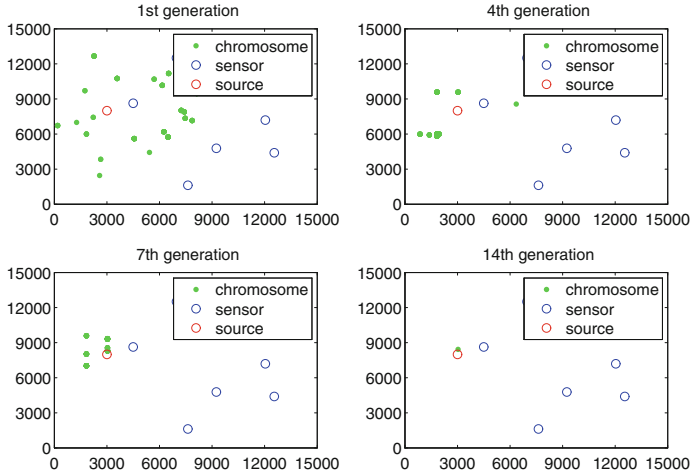


Fig. 4 Distribution of the x and y coordinates estimates during the GA runs for the given generation in 1st time step (rank selection)

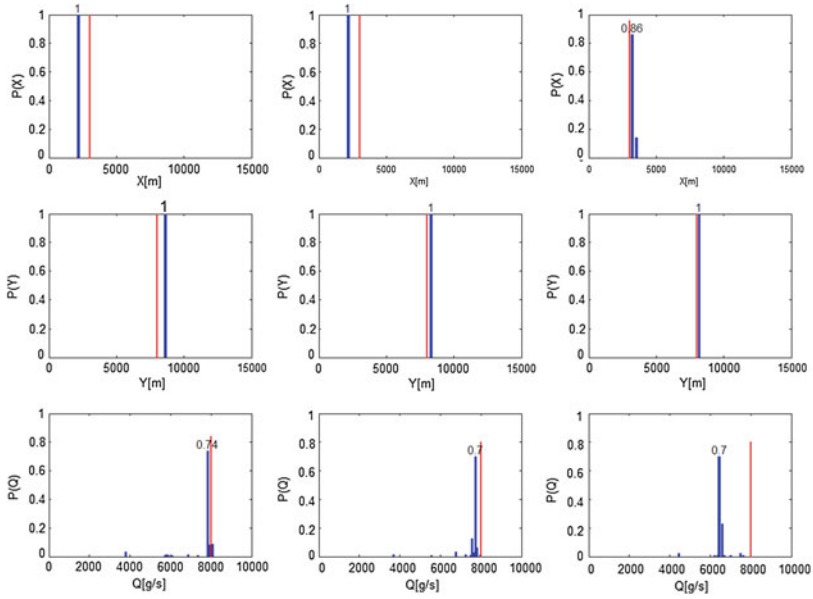


Fig. 5 Probability distributions of the models parameters x, y, and Q for the last generations in 1st, 2nd and 3rd time step (rank selection). *Vertical red lines* represent the target value

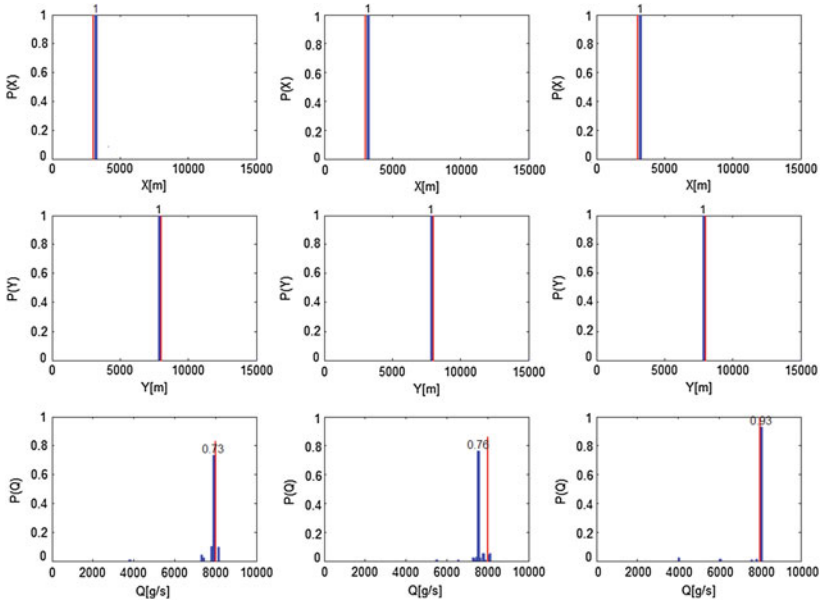
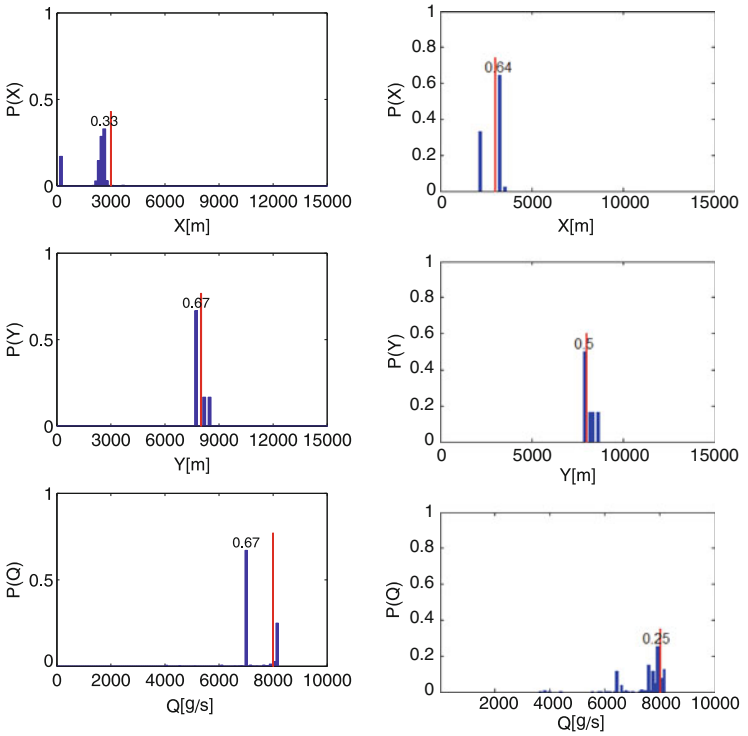


Fig. 6 Probability distributions of the models parameters x, y, and Q for the last generations in 4th, 5th and 6th time step (rank selection). *Vertical red lines* represent the target value

show which values of the parameters were the most frequent in the last generation, which directly is reflected in its probability.

Figures 5 and 6 presents that the first sensors measurements allow to estimate the searched parameters quite close to the target values. However, its estimation varies in the subsequent time-steps. The probability distributions in subsequent time steps reflect how the sensors data support or not the obtained distributions. As the estimated parameter value, we provide the central value of the histogram bar with the highest probability and as the error the half of the bar width. In the 6th time step the following parameters were estimated $P(x = 3050 \pm 75) = 1$, $P(y = 7950 \pm 75) = 1$ and $P(Q = 8100 \pm 40) = 0.93$. To adequately compare the results reported by two proposed algorithms we have estimated the joint marginal distribution of x, y and Q parameters. Figure 7 present the *posterior* distributions averaged over all time steps for the GA algorithm with the hard tournament and with rank selection selection, respectively. The algorithm applying rank selection as the most probable has pointed the parameters $P(x = 3225 \pm 75) = 0.64$, $P(y = 7875 \pm 75) = 0.5$ and $P(Q = 7880 \pm 40) = 0.25$, while the algorithm applying the hard tournament the parameters $P(x = 2925 \pm 75) = 0.33$, $P(y = 7725 \pm 75) = 0.64$ and $P(Q = 7000 \pm 40) = 0.67$. We do not show the distributions for the $z1$ and $z2$ because we do not know the target values for these coefficients. The reason is that the SCIPUFF model used to generate the synthetic concentration data do not allow to specify it directly. In the reconstruction procedure we could of course fix these coefficients



Hard tournament of size 2 selection. Rank selection.

Fig. 7 Cumulative probability distributions of the models parameters x, y, and Q averaged over all time steps for two selection types. *Vertical red lines* represent the target value

according to the stability class pointed by the terrain and wind speed which in this case could be the stability class C for which $z1 = 0.22$ and $z2 = 0.2$. However, our numerical tests showed that we obtain better results when we do not restrict the dispersion coefficients to the one given value. The 'freed' the dispersion coefficients in some acceptable interval assumption allows to better fit the Gaussian plume to the 'real' data.

Comparison of the obtained results leads to the conclusion that algorithms applying both selection methods return similar results for the x and y parameters, at the same time the algorithm using the hard tournament selection as the most probable denotes $Q = 7000$ g/s which differs from the true release rate for 1000 g/s, while for the rank selection algorithm hits the target value. Consequently, we can pointed the algorithm applying the rank selection as more effective. The reason is the applied dynamical termination criteria that resulted in the ten times less computational time for rank selection than for the hard tournament selection. Thus, the rank selection will be applied to the reconstruction of the real field tracer experiment.

4 Reconstruction of the 31 May 2001 Kori Tracer Experiment

We have applied the proposed in the previous sections reconstruction algorithm localize a source of the release with the use of the data from field tracer experiment conducted in May 2001 over the Kori nuclear site [23]. From 2000 to 2002 six field tracer experiments had been conducted at the Kori site to the east of Korea [24]. These experiments have been carried out for the purpose of analyzing the site-specific atmospheric dispersion characteristics and validating a real-time radiological dose assessment system FADAS (Following Dose Assessment System) [25].

In this paper, we use data from the 3rd experiment taken on 31 May 2001. A tracer gas SF_6 was released from the meteorological tower (58 m high) with an average rate 75.79 kg/h. Release started at 12:30 and lasted for 3.5 h. The sampling of the tracer has taken place every 10 min since 15:00 up to 16:00. The 140 tracer gas samplers were disposed in two lines along the roads with the radius of about 3 and 12 km, respectively from the release point. During the experiment, the meteorological data were also measured at several locations. The topography, meteorological towers, sampling points and release point locations present Fig. 8.

During the reconstruction, it is important to implement into the forward atmospheric dispersion model the information about parameters characterizing the state of the atmosphere that can have the most impact on the transport of the dispersed substance. In the case of the Kori tracer experiment, one of the most important factors is the wind field. This experiment took place close to the sea coast of the east and south direction. During the daytime, the wind blows from the east and south by the effects of land-sea breeze. Therefore, the wind patterns are very complicated in coupling with an elaborate hilly topography. Due to character of the experiment we have chosen the SCIPUFF model [16] as the forward model to predict the concentrations at the sensors locations. SCIPUFF can assimilate observational data ranging from a single wind measurement to multiple profiles that include turbulence measurements and/or boundary layer parameters such as Pasquill-Gifford-Turner stability class. Thus, during the reconstruction we have included the wind speed, wind direction and stability class recorded every 15 min since 12:30 up to 16:00 by 5 meteorological towers at a height of 10 m (see Fig. 9). Based on these data SCIPUFF was able to calculate the velocity field from interpolating observations.

Summarizing, the reconstruction problem undertaken in this section is characterized by: (1) the domain area of 15 km $\times$ 15 km, in which the contamination source is searched for; (2) the gas samplers registered the tracer concentration in six-time intervals during 10 min since 15:00 up to 16:00 (i.e. 2.5 h after starting the release); (3) six meteorological towers register the wind speed and its direction every 15 min at a height of 10 m; (4) very complex hilly terrain at the sea coast.

The task to localize the atmospheric contamination source based on the sensors data from the Kori 2001 field tracer experiment was quite challenging. The first problem was the localization of the experiment. The Kori site is located at the bottom of the protrusion and from three sides is surrounded by the sea, with bordering

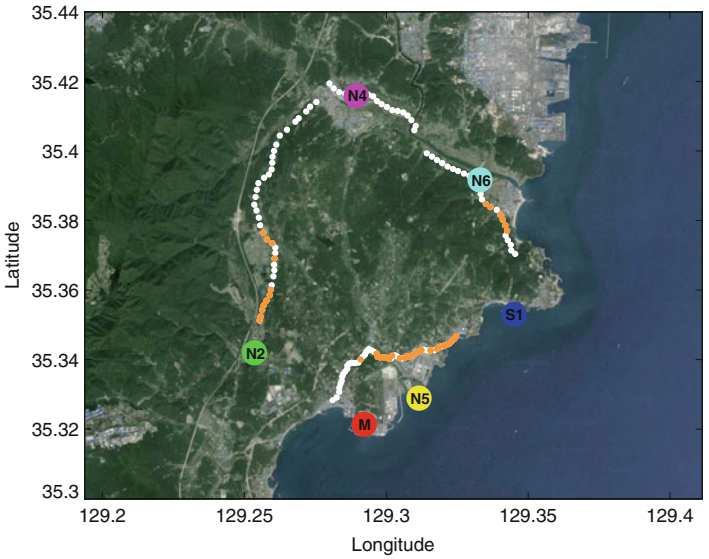


Fig. 8 Location of the release point *M* (*red dot*), sampling points (*white* and *orange dots*) and meteorological towers (*color dots*) during the field tracer experiment conducted on 31 May 2001 at the Kori nuclear site

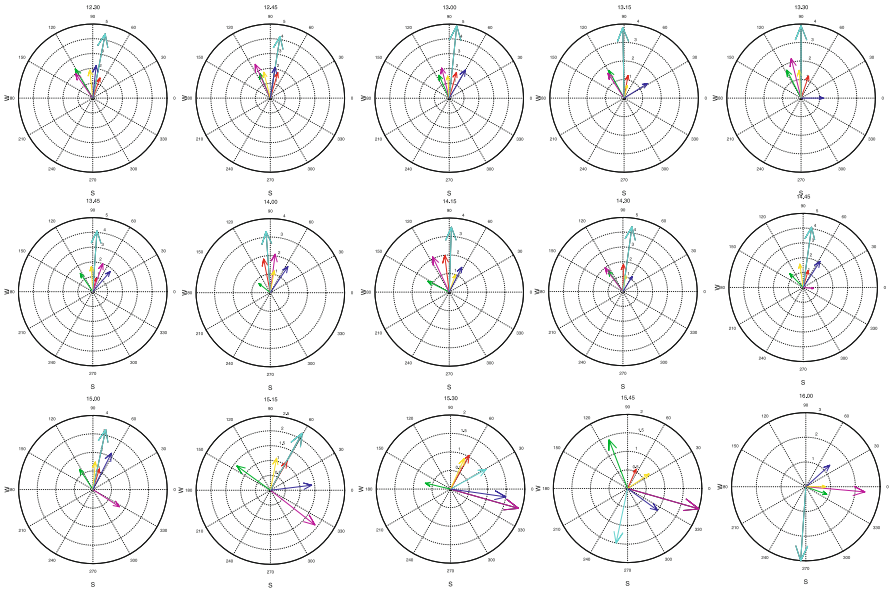


Fig. 9 Measurements of the wind speed and direction every 15 min since 12:30 up to 16:00 during the field tracer experiment conducted on 31 May 2001 at the Kori nuclear site. The *arrows color* corresponds to the meteorological towers presented in Fig. 8

land on the north side (Fig. 8). In this terrain the wind field configuration is very complicated. As presents Fig. 9 during the experiment the wind direction recorded by the meteorological towers changes significantly e.g. by station $N6$ even for 180° and more than 90° for other towers. The 140 automatic gas samplers were disposed along the traffic roads. However, during this experiment significant missing rate of air sampling took place. The main problem of the data is a large discrepancy between concentrations recorded by neighboring sensors i.e. at very close location one sensor returns the 'zero' concentration and the other a high rate. Thus, it was necessary to perform the preselection of the sensors. We have selected 40 sensors marked at the Fig. 8 by orange dots. The selected sensors reported entirely consistent data during the release, hence can be used during the reconstruction.

In this reconstruction the scanned parameters space was $M = \{x, y, Q\}$. The GA setup was identical as described in the Sect. 2.3, but we employed only the rank selection. We have also make use of the dynamic termination criteria. Consequently, the GA is terminated when some values of parameters x, y will be obtained with probability greater than 0.8.

Figure 10 present the posterior distributions for x, y and Q parameters obtained in the first three time steps. One can see that the target values of source coordinates were found by the algorithm quite fast. The dynamic termination criterion allowed to

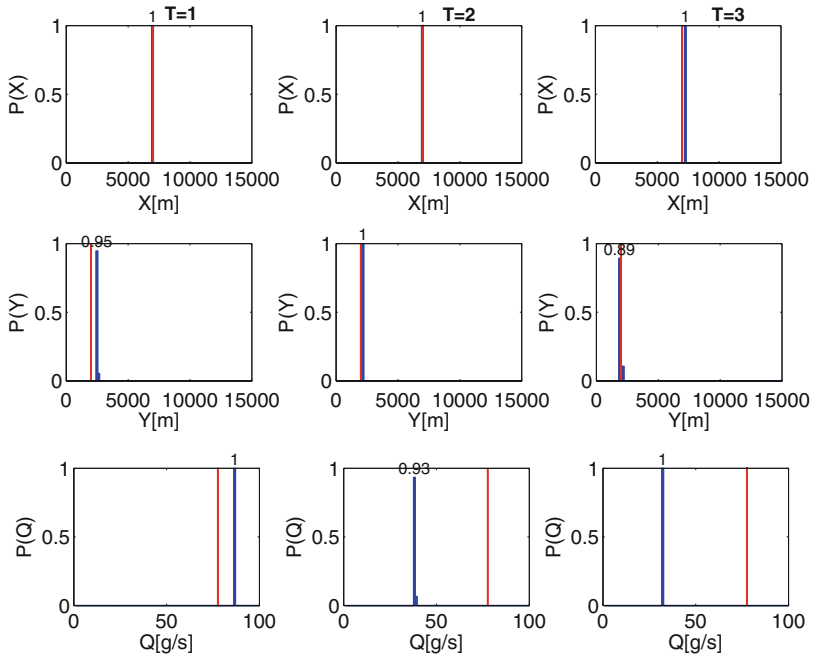


Fig. 10 Probability distributions of the models parameters x, y, and Q for the last generations in 1st, 2nd and 3rd time step for reconstruction of the field tracer experiment conducted on 31 May 2001 at the Kori nuclear site (rank selection). *Vertical red lines* represent the target value

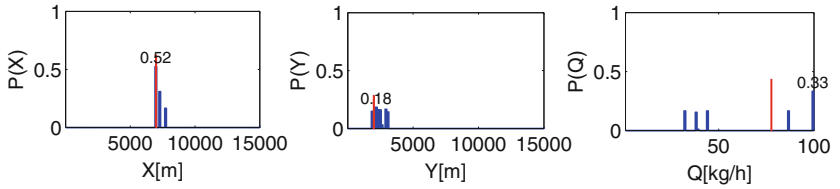


Fig. 11 Cumulative probability distributions of the models parameters x, y, and Q averaged over all time steps for reconstruction of the field tracer experiment conducted on 31 May 2001 at the Kori nuclear site (rank selection). *Vertical red lines* represent the target value

finish the reconstruction in just 45 generations resulting in 6750 runs of the SCIPUFF model (i.e.: 10 generations in T = 1; 9 generations in T = 2; generations in 8 generations in T = 3; and only 6 generations in T = 4, 5, 6). The cumulative posterior distributions are presented in Fig. 11. One can see that the location of the release source was found quite well. Reconstruction algorithm pointed as the most probable $P(x = 6900 \pm 75) = 0.52$, $P(y = 2175 \pm 75) = 0.18$ while the true values were $x = 7000$ and $y = 2000$. The worst situation is with the release rate, because it was over estimated by 20 kg/h. There are several reasons for this mismatch between the predictions and the measurements for the source reconstruction during Kori tracer experiment. First of all, all concentrations measured during the experiment were averaged values from a 10 min release, whereas model predictions are steady-state results. The second reason is that the height of the release was 58 m while during the reconstruction it was fixed at the sensor height i.e. 10 m. It is also important to notice, that very hilly terrain topography was not incorporated into the SCIPUFF model. In consequence the reconstruction was not able to take into account the diversity of the measured by the gas sampler concentrations owing to the altitude difference connected with the placing the gas samplers along the road in the valley between the hills. However, taking the above mention we can conclude that the proposed reconstruction algorithm even in the so complicated domain was able to quite fast find the release source location, which is one of the primary goals in the real-time reconstructions.

5 Conclusion

We have presented a methodology to reconstruct a source causing an area of contamination, based on a set of measurements. The method combines Bayesian inference with the genetic algorithm and produces posterior probability distributions of the parameters describing the unknown source. Developed dynamic data-driven event reconstruction model couples data and pollutant dispersion simulations through Bayesian inference. This approach successfully provide the solution to the stated

inverse problem i.e. having the downwind concentrations and knowledge of the wind field algorithm found the most probable location of the source and release strength.

We have proposed the dynamical termination criteria for the genetic algorithm. This criteria reflects the probabilistic aspect of the obtained solution i.e. the GA is terminated when some of the searched parameters are pointed with satisfactorily probability. This approach allows to optimize the algorithm computational time. We show that in the presented problem the rank selection is more efficient than the hard tournament selection. The developed algorithm was also employed to perform the reconstruction with use of the real field tracer experiment data. In the case of the Kori experiment reconstruction algorithm correctly assessed the x and y coordinates and overestimated release rate.

The probabilistic aspect of the solution optimally combines a probable answer with the uncertainties of the available data. Among several possible solutions, the Bayesian source reconstruction points the values of the model parameters that are more consistent with the currently available data.

Acknowledgments This work was supported by the Welcome Programme of the Foundation for Polish Science operated within the European Union Innovative Economy Operational Programme 2007–2013. Authors thank Piotr Kopka for help in preparing Figs. 8 and 9.

References

1. A. Keats, E. Yee, F.-S. Lien, Bayesian inference for source determination with applications to a complex urban environment. Atmos. Environ. **41**, 465–479 (2007). doi:10.1016/j.atmosenv.2006.08.044
2. J.A. Pudykiewicz, Application of adjoint tracer transport equations for evaluating source parameters. Atmos. Environ. **32**, 303–3050 (1998). doi:10.1016/S1352-2310(97)00480-9
3. G. Johannesson et al., Sequential Monte-Carlo based framework for dynamic data-driven event reconstruction for atmospheric release, in *Proceedings of the Joint Statistical Meeting*, (American Statistical Association and Cosponsors, Minneapolis, 2005), pp. 73–80
4. M. Borysiewicz, A. Wawrzynczak, P. Kopka, Stochastic algorithm for estimation of the model's unknown parameters via Bayesian inference, in *Proceedings of the Federated Conference on Computer Science and Information Systems* (IEEE Press, Wroclaw, 2012), pp. 501–508. ISBN 978-83-60810-51-4
5. M. Borysiewicz, A. Wawrzynczak, P. Kopka, Bayesian-based methods for the estimation of the unknown model's parameters in the case of the localization of the atmospheric contamination source. Found. Comput. Decis. Sci. **37**(4), 253–270 (2012). doi:10.2478/v10209-011-0014-9
6. A. Wawrzynczak, P. Kopka, M. Borysiewicz, Sequential Monte Carlo in Bayesian assessment of contaminant source localization based on the distributed sensors measurements. Lecture Notes in Computer Sciences. PPAM 2013, vol 8385, Part II, Chap. 38, pp. 407–417 (2014). doi:10.1007/978-3-642-55195-6_38
7. J.H. Holland, *Adaptation in Natural and Artificial Systems*, 2nd edn. (MIT Press, Cambridge, 1992)
8. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison Wesley Longman, London, 2006)
9. P.J. Fleming, R.C. Purshouse, Genetic algorithms in control systems engineering, in *Proceedings of the 12th IFAC World Congress*, pp. 383–390 (2001)

10. R.M. Goodall, K. Michail, J.F. Whidborne, A.C. Zolotas, Optimised configuration of sensing elements for control and fault tolerance applied to an electro-magnetic suspension. Ph.D. thesis, Loughborough University, UK (2009)
11. P. Rustem (ed.), Genetic algorithms in applications. InTech, Chapters published 21 March 2012 under CC BY 3.0 license (2012). doi:10.5772/2675. ISBN 978-953-51-0400-1
12. C.T. Allen, S.E. Haupt, Source characterization with a genetic algorithm-coupled dispersion-backward model incorporating SCIPUFF. Department of Meteorology, The Pennsylvania State University (2006). doi:10.1175/JAM2459.1
13. A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. **3**(2), 121–141 (1999). doi:10.1109/4235.771166
14. A. Saremi, T.Y.E. Mekkawy, G.G. Wang, Tuning the parameters of a memetic algorithm to solve vehicle routing problem with backhauls using design of experiments. Int. J. Oper. Res. **4**(4), 206–219 (2007)
15. O. Roeva, S. Fidanova, M. Paprzycki, Influence of the population size on the genetic algorithm performance in case of cultivation process modelling, in *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems*, pp. 371–376 (2013)
16. R.I. Sykes et al., PC-SCIPUFF Version 1.2PD Technical Documentation. ARAP report no. 718. Titan Corporation (1998)
17. A. Gelman, J. Carlin, H. Stern, D. Rubin, *Bayesian Data Analysis* (Chapman & Hall/CRC, Boca Raton, 2003)
18. D.B. Turner, *Workbook of Atmospheric Dispersion Estimates* (Lewis Publishers, Boca Raton, 1994)
19. F. Pasquill, The estimate of the dispersion of windborne material. Meteorol. Mag. **90**(1063), 33–49 (1961)
20. F.A. Gifford Jr., Atmospheric dispersion calculation using generalized Gaussian plum model. Nucl. Saf. **2**(2), 56–59, 67–68 (1960)
21. A. Wawrzynczak et al., Recognition of the atmospheric contamination source localization with the genetic algorithm, Studia Informatica, UPH, Siedlce (in print) (2015)
22. A. Wawrzynczak, M. Jaroszyski, M. Borysiewicz, Data-driven genetic algorithm in Bayesian estimation of the abrupt atmospheric contamination source, in P*roceedings of the 2014 Federated Conference on Computer Science and Information Systems*. Annals of Computer Science and Information Systems, vol. 2, pp. 519–527 (2014). doi:10.15439/2014F272
23. K.S. Suh, E.H. Kim, W.H. Hwang, H.J. Jeong, M.H. Han, Atmospheric dispersion modeling over the Kori nuclear site, in *11th International Congress of the International Radiation Protection Association*, Madrit, Spain (2004)
24. M.H. Han, E.H. Kim, K.S. Suh et al., Simulation of the dispersion of radioactive effluents over the Kori site using field tracer experiment. J. Nucl. Sci. Technol., Supplement **4**, 423–426 (2004)
25. M.H. Han, E.H. Kim, K.S. Suh et al., Field tracer experiments over nuclear sites for the validation of a Korean real-time atmospheric dispersion and dose assessment system (FADAS). Int. J. Environ. Pollut. **16**, 1–6 (2001)