Computers and Chemical Engineering 55 (2013) 83–96

Contents lists available at SciVerse ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

A new metaheuristic based approach for the design of sensor networks

Mercedes Carnero[a], José Hernández[a], Mabel Sánchez[b,*]

[a] Facultad de Ingenieria UNRC, Campus Universitario, 5800 Río Cuarto, Argentina
[b] Planta Piloto de Ingeniería Química (UNS-CONICET), Camino La Carrindanga Km 7, 8000 Bahía Blanca, Argentina

ARTICLE INFO

Article history:
Received 6 November 2012
Received in revised form 15 March 2013
Accepted 1 April 2013
Available online 16 April 2013

Keywords:
Sensor network design
Combinatorial optimization
Estimation of Distribution Algorithms
Tabu Search

ABSTRACT

The optimal design of sensor networks consists in selecting the type, number and location of sensors that provide the required quantity and quality of process information by optimizing an appropriate objective function. The problem is multimodal and involves many binary variables, therefore a huge combinatorial optimization problem results. In this work, the design is solved using a metaheuristic based approach. A strategy that combines the advantages of Tabu Search and Estimation of Distribution Algorithms is presented, which is able to solve high scale designs since it can be implemented to run in parallel. Application results of the methodology to the optimal selection of instruments for networks of incremental size are provided.

1. Introduction

Process information is the foundation upon which other plant activities (monitoring, regulatory and supervisory control, real time optimization, planning and scheduling, fault diagnosis, etc.) are based. To fulfill information requirements, regarding its quality and availability, it is essential to install an appropriate sensor network (SN) in the plant, and also data reconciliation and SN maintenance tasks should be executed during process operation.

The selection of the set of variables to be measured, which is optimal with respect to the specified criteria and simultaneously satisfies the information requirements, is called the sensor network design problem (SNDP). The designer should decide to measure a variable or not. These decisions are mathematically formulated in terms of binary variables, which indicate the presence or absence of sensors. A combinatorial optimization problem arises that is usually multimodal and involves many binary variables. To solve it, different deterministic and metaheuristic based approaches have been presented.

Regarding the design of sensor networks (DSNs) for monitoring purposes, Bagajewicz (1997) introduced the use of binary variables and formulated a MINLP problem to obtain cost optimal networks for linear systems subject to constraints on the precision and robustness of variable estimates. A depth-first tree search algorithm was implemented as solution scheme.

Chmielewski, Palmer, and Manousiouthakis (2002) dealt with the design of minimum cost SN subject only to constraints on estimates precision for linear systems. Unmeasured variables were replaced by measurements obtained using dummy sensors of large variance and no cost, and a convex MILP formulation was solved using a Branch and Bound algorithm. With the same purpose, Bagajewicz and Cabrera (2002) addressed an alternative MILP approach, and Kelly and Zyngier (2008) presented a MILP model based on the Schur complements theorem.

Remarkable improvements of the aforementioned depth-first tree search algorithm appeared later. In this sense, Gala and Bagajewicz (2006) proposed the combination of cutsets of the process graph instead of measurements to reduce the computational time, and used a tree enumeration search along with a graph decomposition technique as solution scheme. Nguyen and Bagajewicz (2008) tackled the SNDP of nonlinear systems replacing the cutsets by model equations, and the same authors recently presented a breadth-first/level transversal tree search method with the same purpose (Nguyen & Bagajewicz, 2011).

With respect to the DSN for fault diagnosis purposes, Bhushan and Rengaswamy (2000) proposed algorithms based on the signed directed graph representation of the process to satisfy fault observability and resolution criteria. Then the same authors presented SND formulations based on diagnosability and reliability criteria (Bhushan & Rengaswamy, 2002a, 2002b). Bagajewicz, Fuxman, and Uribe (2004) addressed minimum cost SNDPs subject to

* Corresponding author. Tel.: +54 291 4861700x249; fax: +54 291 4861600.
E-mail addresses: mcarnero@ing.unrc.edu.ar (M. Carnero), jlh@ing.unrc.edu.ar (J. Hernández), msanchez@plapiqui.edu.ar (M. Sánchez).

0098-1354/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compchemeng.2013.04.007

Nomenclature

B	best individual of the generation
c	acquisition cost vector
C	penalty function
D	matrix of solutions
E_{le}	degree of estimability of variable *le*
$f_{\mathbf{Q}}$	joint probability distribution
F	cumulative distribution function
FB	factibility bound
H	evaluation function
g	vector of constraints
h	Frequency based Tabu list
I_0	initial set of instruments
ic	cluster index
I_list	inhibition list
L	bound for SOTS
LR	learning rate
MS	mutation amount
n	number of process variables
nc	number of clusters
N	neighborhood of possible solutions
M	number of individuals for each instance of PBIL
NPBIL	number of instances of PBIL executed in parallel
# MaxGeneration	maximum number of iterations for pPBIL
# maxiter	maximum number of iterations for SOTS
nso	number of iterations between allowable calls to SOTS
p	probability vector
pt	Tabu Tenure Period
ph	number of iterations before **h** is reset
Pinteraction	crossover probability
PMUTA	mutation probability
Pr	probability
pso	application probability of SOTS
q	solution vector
Q	random n-dimensional vector
R	process model equations
S	set of key variables
S_σ	set of key variables subject to precision constraints
S_E	set of key variables subject to estimability constraints
t	Recency based Tabu list
u	vector of unmeasured variables
UC	number of unsatisfied constraints
x	vector of measured variables
z	vector of process variables
$\hat{\sigma}_{lp}$	standard deviation of the *lp*th variable estimate

variable-estimates precision constraints and fault observability and resolution requirements using a MILP method. Later on, a general framework to design robust SN was introduced by Bhushan, Narashimhan, and Rengaswamy (2008), and the optimization problem was solved using constraint programming (Kotecha, Bhushan, & Gudi, 2007).

Solution procedures based on metaheuristics were also developed for the DSNs. Initially techniques based on Genetic Algorithms (GAs) were proposed. Sen, Narasimhan, and Deb (1998) presented an evolutionary method devoted to the selection of flowmeters for nonredundant SNs, which optimizes single criteria such as cost, reliability or estimation accuracy based on concepts from graph theory. Carnero, Hernández, Sánchez, and Bandoni (2001) used a GA whose operators were modified based on linear algebra concepts to optimize single or multiple criteria for the same type of SNs. Furthermore, Viswanath and Narasimhan (2001) developed an evolutionary approach for the design of linear redundant SNs that maximizes the reliability of variable estimates.

Regarding the design of general SNs, Benqlilou, Graells, Musulin, and Puigjaner (2004) applied the GA toolbox of MATLAB program to solve the design and retrofit of reliable SNs, and Gerkens and Heyen (2005) presented two ways of parallelizing the GA to reduce the solution time. Then a hybrid procedure (HGA) was developed by Carnero, Hernández, Sánchez, and Bandoni (2005) to minimize the instrumentation cost subject to precision constraints on key variables. It uses a structured population in the form of neighborhoods and a local optimizer of the best current solutions, which provide a good balance between the algorithm capabilities of exploration and exploitation. Also, Gerkens and Heyen (2008) proposed a general approach for designing the cheapest SN able to detect and locate a set of specified faults and used a parallel implementation of the GA to select sensors and their locations.

In the last decade some applications of Tabu Search (TS) metaheuristic for the solution of chemical engineering problems have appeared. It was reported that TS has a more flexible and effective search behavior than other stochastic methods as consequence of the use of adaptive memory (Cavin, Fischer, Glover, & Hungerbuhler, 2004; Lin & Miller, 2004a, 2004b). Within the framework of TS, a Strategic Oscillation Technique around the feasibility boundary (SOTS) was applied by Carnero, Hernández, and Sánchez (2005) to develop a new strategy for the DSNs. The procedure efficiently searches the solution space, significantly reducing the number of required calls to the evaluation function in comparison with HGA and the Classic TS.

There exist other population based methodologies that have demonstrated a rewarding performance for solving complex combinatorial optimization problems, and emerge as an alternative to GAs. In this sense, Estimation of Distribution Algorithms (EDAs) have been recognized as a new computing paradigm in evolutionary computation (Emmendorfer & Pozo, 2009). There is neither crossover nor mutation operators in EDAs. Instead the new population is generated by sampling the probability distribution, which is estimated from a database containing selected individuals of the previous generation. Within the framework of EDAs, the concepts of competitive learning used in artificial neural networks were introduced to direct the search by Baluja (1994), who devised The Population Based Incremental Learning Algorithm (PBIL).

Recently, Carnero, Hernández, and Sánchez (2009) presented a procedure for the design of SNs based on PBIL, which uses an improvement solution technique. The comparison between the design strategies based on PBIL and SOTS indicated that the last one strongly depends on the structure of the initial solution. If a good starting point is provided it produces a high quality solution with a low computational effort. In contrast, PBIL based methodology is more robust. It is capable of making replicas of the best solutions starting from lower quality initial points at the expense of an increment of the computational time. Because PBIL can be naturally run in parallel, the total elapsed time could be reduced for a given number of calls. Taking into account these results, in this work a new procedure for the DSNs is proposed that combines the advantages of SOTS and PBIL.

The rest of the paper is structured as follows. In Section 2, the general SNDP for monitoring purposes is stated and a particular one, that is, the minimum cost SNDP subject to precision and estimability constraints, is briefly revised. A new SND procedure is described in Section 3, and its rewarding performance for the location of sensors of incremental size systems is provided in Section 4. A conclusion and future works section ends this work.

2. Problem formulation

Let us assume **R** is a set of *r* non linear algebraic equations that represents the operation of a process under steady state conditions

$$\mathbf{R}(\mathbf{z}) = \mathbf{R}(\mathbf{x}, \mathbf{u}) = 0, \tag{1}$$

where **z** is the *n* dimensional vector of process variables, and **x** and **u** represent the vector of measured and unmeasured variables respectively. The optimal selection of instruments during plant design or upgrade consists in determining the optimal partition of vector **z** in vectors **x** and **u**, and can be formulated in general form as follows

$$\begin{aligned} &Min/Max \quad f(\mathbf{q}) \\ &s.t. \\ &\qquad g_l(\mathbf{q}) \le g_l^*(\mathbf{q}) \quad \forall l \in S \\ &\qquad q_i = 1 \quad \forall i \in I_0 \\ &\qquad \mathbf{q} \in \{0,1\}^{n-|I_0|} \end{aligned} \tag{2}$$

where **q** is the $(n - I_0)$ dimensional vector of binary variables such that: $q_i = 1$ if variable *I* is measured, and $q_i = 0$ otherwise. Furthermore $f(\mathbf{q})$ stands for a one dimensional objective function, $g_l(\mathbf{q})$ represents the constraint imposed on the quality of the *l*th key variable estimate, considering there exists only one constraint for each key variable, *S* is the set of key process variables, and I_0 stands for the initial set of instruments that is empty at the network design stage. In this formulation it is assumed that there is only one potential measuring device for each variable and, no instruments' localization restrictions are imposed.

In general nonlinear discrete optimization problems arise, and the dimension of the search space increases significantly for large scale processes. Consequently, the design turns out to be a huge combinatorial optimization problem that may have many local optima. In these cases, it is really valuable that the solution strategy provides at least a good solution, if not the global optima, and that it can be run in parallel computers to reduce execution times.

One popular formulation of Problem 2 is the minimum cost SNDP that satisfies precision and estimability constraints for a set of key variable estimates. It is formulated by Eq. (3) where $\mathbf{c^T}$ is the cost vector; $\hat{\sigma}_{lp}$ is the standard deviation of the *lp*th variable estimate after a data reconciliation procedure is applied and E_{le} stands for the degree of estimability of variable *le* (Bagajewicz & Sánchez, 1999). For this formulation E_{le} is set equal to one, consequently only a variable classification procedure run is needed to check its feasibility. Furthermore S_σ and S_E are the set of key process variables with requirements in precision and estimability respectively. It is assumed that a linearized algebraic model represents process operation and, measurements are subject to non correlated random errors.

$$\begin{aligned} &Min \quad \mathbf{c^T q} \\ &s.t. \\ &\qquad \hat{\sigma}_{lp}(\mathbf{q}) \le \sigma_{lp}^*(\mathbf{q}) \quad \forall lp \in S_\sigma \\ &\qquad E_{le}(\mathbf{q}) \ge 1 \quad \forall le \in S_E \\ &\qquad \mathbf{q} \in \{0,1\}^n \end{aligned} \tag{3}$$

3. New PBIL-SOTS based approach

3.1. An overview

Estimation of Distribution Algorithms constitute a new area of evolutionary computation. In these procedures, the use of classic crossover and mutation operators for the generation of the next population is replaced by the estimation of the joint probability distribution of the variables, and its posterior sampling. Therefore, the probabilistic model that guides the search is based on the knowledge of the relationships among variables. The use of EDAs provides other advantages, i.e., well developed statistical machine learning techniques can be readily employed in these methods; the search is more effective because they can learn and record more about the search space.

The relationships among variables are implicitly taken into account by Gas, instead they are explicitly considered in EDAs by means of the probabilistic model associated with the individuals of each generation. The estimation of that model constitutes the key issue of EDAs. According to the model complexity, EDAs can be broadly divided into univariate, bivariate or multivariate approaches (Hauschild & Pelikan, 2011). To handle variable interdependencies, the second and third classes of EDAs require complex learning algorithms and significant additional computational resources. Therefore it results convenient to analyze the performance of the first class of EDAs, although being simple, for solving the optimization problem under study.

One of the most popular EDAs that assumes independent relationships among variables is PBIL, which is the core of the proposed solution strategy. This procedure was originally devised to obtain the optimum of a function defined in the *n* dimensional binary space by Baluja (1994), and further improved by Baluja and Caruana (1995). Among others, Pang, Hu, and Hing (2006) and Wan and Qiu (2008) have shown good application results of that procedure for solving complex problems.

The pseudocode of PBIL is shown in Fig. 1. At first the probability vector $\mathbf{p}^0$ is initialized. Then an iterative procedure is run until a termination criterion is satisfied. At the *j*th iteration, a population made up of *M* individuals is generated by simulation taking into account the probabilities in $\mathbf{p}^{j-1}$. Each individual is evaluated using the fitness function, *H*, and the best one in terms of fitness, $\mathbf{B}^j$, is determined. Then the probability vector is updated, position by position, using $\mathbf{B}^j$ and the learning rate *LR* as follows

$$p_i^j = p_i^j(1 - LR) + B_i^j LR \quad (I = 1 : n) \tag{4}$$

where p_i^j and B_i^j represent the *i*th elements of the current probability and best solution vectors, respectively. After that, a mutation of $\mathbf{p}^j$ is performed, position by position, using the mutation probability *PMUTA* and the mutation amount *MS*. For each element of $\mathbf{p}^j$, a random number in the range [0 1] is generated and, if it is less than *PMUTA* the following equation is applied

$$p_i^j = p_i^j(1 - MS) + rand(0,1)MS \quad (I = 1 : n) \tag{5}$$

A parallel implementation of PBIL, pPBIL, which allows *NPBIL* instances being executed independently, has been proposed. After *NPBIL* subpopulations evolve one iteration, their **p** vectors are related by different mechanisms before sampling to exchange information among them.

The performance of pPBIL for solving the SNDP was enhanced by implementing a simple local search procedure by Carnero et al. (2009). The solution improvement method is based on the identification of constructive blocks which belong to high quality solutions and their permanence during the evolution. Furthermore subpopulation probability vectors are modified using uniform crossover with probability *Pinteraction*. The pseudocode of the procedure is included in Fig. 2. Application results show that the technique performs well for medium scale problems, but exhibits limitations for large ones. Therefore, high dimensional problems are addressed in this work by incorporating to pPBIL a more sophisticated local

```
Initiate the probability vector p
 while (stopping criteria = .FALSE.)
        Generate M individuals by simulation according to p
        Evaluate the fitness function H for each member of the population
        Select the best solution B
        Upgrade p using B and the learning rate LR
        Mutate p using a probability of mutation PMUTA and a mutation amount MS
 end
```

Fig. 1. Pseudocode of PBIL.

search algorithm. The metaheuristic SOTS, which has shown a rewarding performance for solving the SNDP by itself, is selected with that purpose.

Tabu Search is a metaheuristic approach used to solve optimization problems (Glover & Laguna, 1997). It uses a guided local search procedure to explore the entire solution space in a way that makes it difficult to be entrapped in local optima and prevents solution cycling. Previous solutions information, which is stored in Tabu lists, is used to efficiently guide the local search.

Given a current solution $\mathbf{q}$, a neighborhood of solutions N($\mathbf{q}$) is formed by modifying $\mathbf{q}$ through a move, which is a problem dependent operation. Then the elements of N($\mathbf{q}$) that correspond to solutions recently visited are excluded from it. The short term knowledge applied to update N($\mathbf{q}$) is provided by means of a Recency based Tabu list, where new solutions are incorporated at each iteration, and remained there as forbidden moves during the Tabu Tenure Period, *pt*. After that, the members of N($\mathbf{q}$) are evaluated to select the best one that becomes the starting point for the new iteration ($\mathbf{q}'$), even if it is worse than $\mathbf{q}$. At this point, the use of long term memory allows the inclusion of solutions not ordinarily found in the search, because the evaluation function of N($\mathbf{q}$) members associated with moves that have been done more often are penalized. These moves are contained in the Frequency based Tabu list, which is reset after *ph* iterations. Also the best solution ever found ($\mathbf{q}^*$) is saved. Aspiration criteria are applied to determine when the Tabu lists can be overridden to avoid missing good solutions.

Within the framework of TS, other procedures are incorporated for search intensification and diversification such as SO, which consists of a sequence of destructive and constructive phases. Given a feasible solution, the search is strategically driven to cross the feasibility boundary, and to continue in the infeasible region until certain depth. The destructive phase finishes when Condition 1 is satisfied; then move rules are modified to drive the search towards the feasible region (constructive phase) until Condition 2 is fulfilled. The process of repeatedly crossing the feasibility boundary from different directions originates an oscillatory behavior. Standard TS mechanisms are applied to avoid going back over previous trajectories. In Fig. 3, the general pseudocode of SOTS is included.

3.2. Special features of pPBIL-SOTS strategy for sensor network design

In this subsection, specific issues of the algorithm proposed for the DSNs are addressed.

```
Initiate NPBIL probability vectors p_k (k =1,…, NPBIL)
while (stopping criteria = .FALSE.)
  for k = 1,…, NPBIL
      Generate M individuals by simulation according to p_k
      Evaluate the fitness function H for each member of the population
      Apply the Improvement Solution Method to each member of the population
      Select the best solution B_k
      Upgrade p_k using using B_k and the learning rate LR
      Mutate p_k using a probability of mutation PMUTA and a mutation amount MS
  end
  Set OffSpring= Ø
  for k = 1,…, NPBIL step 2
      Select 2 individuals (parents) among all vectors p
      if random<Pinteraction
          Use uniform crossover to calculate two children
          Add children to OffSpring
      else
          Add parents to OffSpring
      end
   end
   for k=1,..., NPBIL
        Set p_k=OffSpring(k)
   end
end
```

Fig. 2. Pseudocode of pPBIL using a simple local search procedure.

```
Given: initial solution q0

if q0 is feasible
    Set Phase=1 /* destructive phase */
else
    Set Phase=0  /* constructive phase */
end
Set q*=q= q0
for i =1 to  # maxiter
    if  Phase =1
        Generate neighbourhood N1(q) and evaluate
        Get  best neighbour q'
        if condition1=true
            Phase = 0
        end
    else
        Generate neighbourhood N2(q) and evaluate
        Get  best neighbour q'
        if  condition2=true
            Phase=1
        end
    end
    q=q'
    Update Recency and Frequency Tabu lists
    if q ≺ q*
        q*=q
    end
end
return q*
```

Fig. 3. Pseudocode of SOTS algorithm.

3.2.1. Evaluation function

In this work the following fitness function is applied

$$H = \begin{cases} \sum_{i=1}^{n} c_i q_i & \text{if } \mathbf{q} \text{ is feasible} \\ \sum_{i=1}^{n} c_i (1 + C(\mathbf{q})) & \text{if } \mathbf{q} \text{ is infeasible} \end{cases} \quad (6)$$

where $\sum_{i=1}^{n} c_i$ is the upper bound of the SNDP objective function, $C(\mathbf{q})$ takes into account constraint violations as follows

$$C(\mathbf{q}) = \frac{1}{UC} \sum_{l=1}^{UC} \frac{g_l - g_l^*}{g_l} \quad (7)$$

and, *UC* stands for the number of unsatisfied constraints. The value $\sum_{i=1}^{n} c_i$ is the cost of the SN when all variables are measured, and is called Feasibility Bound (*FB*).

3.2.2. Initialization of probability vectors

Let us consider a solution vector $\mathbf{q} = \{q_1, q_2, \ldots, q_n\}$ is a realization of an n dimensional random vector $\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_n\}$, where Q_i is a binary variable. If the joint probability distribution of $\mathbf{Q}$ is denoted as $f_{\mathbf{Q}}$, then $\Pr(\mathbf{Q} = \mathbf{q}) = f_{\mathbf{Q}}(\mathbf{q})$. Furthermore, if the marginal distribution function of each unidimensional random variable Q_i is p_i, then $\Pr(Q_i = q_i) = p_i(q_i)$.

Taking into account previous notation, the probabilistic model assumed by PBIL can be factorized as the product of the marginal probability distribution of each variable since they are considered independent

$$f_{\mathbf{Q}}(q_1, q_2, \ldots, q_n) = \prod_{i=1}^{n} p_i(q_i) \quad (8)$$

In early works about PBIL, the elements of $\mathbf{p}$ are initialized to 0.5, therefore sampling from this vector yields random solutions. In contrast, the proposed strategy incorporates specific problem knowledge to build the probabilistic model. For each pPBIL subpopulation, a set D_0 of feasible solutions regarding the estimability of key process variables is formed, which guarantees a certain degree of structural diversity. Then the initial probability vector, $\mathbf{p}^0$, is estimated from D_0. The procedure is made up of three steps:

(a) Generation of a previous subpopulation of solutions *PP* that satisfy estimability constraints on key variables. The methodology described by Carnero, Hernández, Sánchez, and Bandoni (2005) is applied.
(b) Selection of solutions to include in D_0. At first the $M/2$ solutions in *PP* that have the best fitness values are incorporated to D_0. The remaining $M/2$ members of D_0 are chosen by measuring the Hamming distance between each candidate solution in *PP* and the existing members of D_0 and selecting the most distant ones.
(c) Estimation of the initial probability vector. Since Q_i follows a Bernoulli distribution, p_i is equal to the expected value of Q_i. A maximum likelihood estimate of this value is the sample mean of Q_i, which is used to initialize the ith element of $\mathbf{p}^0$. Therefore p_I^0 is calculated from D_0 as follows

# individual	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	fitness
1	1	1	1	1	1	1	1	1	13000
2	1	0	0	0	0	1	0	0	16500
3	1	1	1	0	0	1	0	0	7500
4	0	1	0	0	0	1	0	1	5500
5	1	1	0	0	0	0	0	1	17000
6	1	1	0	1	0	1	1	1	9000
7	1	1	0	0	0	1	0	0	18000
8	1	0	1	0	1	1	1	0	8500
9	1	1	1	0	0	1	0	0	7500
10	1	1	1	1	0	0	1	0	20500
11	1	0	0	1	1	1	0	0	6500

PP

# individual	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	fitness
4	0	1	0	0	0	1	0	1	5500
11	1	0	0	1	1	1	0	0	6500
3	1	1	1	0	0	1	0	0	7500
8	1	0	1	0	1	1	1	0	8500
1	1	1	1	1	1	1	1	1	13000
10	1	1	1	1	0	0	1	0	20500

D_0

Fig. 4. Building of the initial population from a previous set.

# individual	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	fitness
4	0	1	0	0	0	1	0	1	5500
11	1	0	0	1	1	1	0	0	6500
3	1	1	1	0	0	1	0	0	7500
8	1	0	1	0	1	1	1	0	8500
1	1	1	1	1	1	1	1	1	13000
10	1	1	1	1	0	0	1	0	20500

D_0

$$p_i = \frac{1}{M}\sum_{1}^{M} q_i \quad (i = 1:n)$$

0.83	0.67	0.67	0.50	0.50	0.83	0.50	0.33

$\mathbf{p}^0$

Fig. 5. Estimation of the initial probability vector.

$$p_i = \frac{1}{M}\sum_{1}^{M} q_i \quad (i = 1:n) \tag{9}$$

Example 1. Let us consider a process whose operation is formulated using a set of mass balance equations represented by the incidence matrix **R**, and assume that estimability and precision constraints are imposed on sets $S_E = \begin{bmatrix} z_1 & z_2 & z_6 \end{bmatrix}$ and $S_\sigma = \begin{bmatrix} z_2 & z_6 \end{bmatrix}$, respectively.

$$\mathbf{R} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 \end{bmatrix}$$

Fig. 4 shows a set *PP* of potential solutions. To satisfy the constraint $E \geq 1$, a variable should be measured or unmeasured but observable. Consequently, each member of *PP* represents a set of sensors that allows the estimation of all required variables using measurements and mass balance calculations. In the same figure, the initial population D_0 for $M = 6$ is shown. Furthermore, the estimation of the initial probability vector in terms of D_0 members is represented in Fig. 5.

3.2.3. Sampling

A new population is generated at the *j*th iteration of PBIL by sampling the $\mathbf{p}^{j-1}$ probability vector. For this purpose a simulation is carried out according to the inverse transformation method, which uses a generator of uniformly distributed random numbers, *U*, to simulate a sample of the random variable of interest.

Let us consider that the binary variable Q_i assumes values 0 or 1 with the following probability

$$\Pr(Q_i = 1) = p_i^{j-1}$$

$$\Pr(Q_i = 0) = 1 - p_i^{j-1}$$

Therefore its cumulative distribution function F can be stated as

$$\mathrm{F}(q_i) = \begin{cases} 0 & \text{if } q_i < 0 \\ 1 - p_i^{j-1} & \text{if } 0 \leq q_i < 1 \\ 1 & \text{if } 1 \leq q_i \end{cases}$$

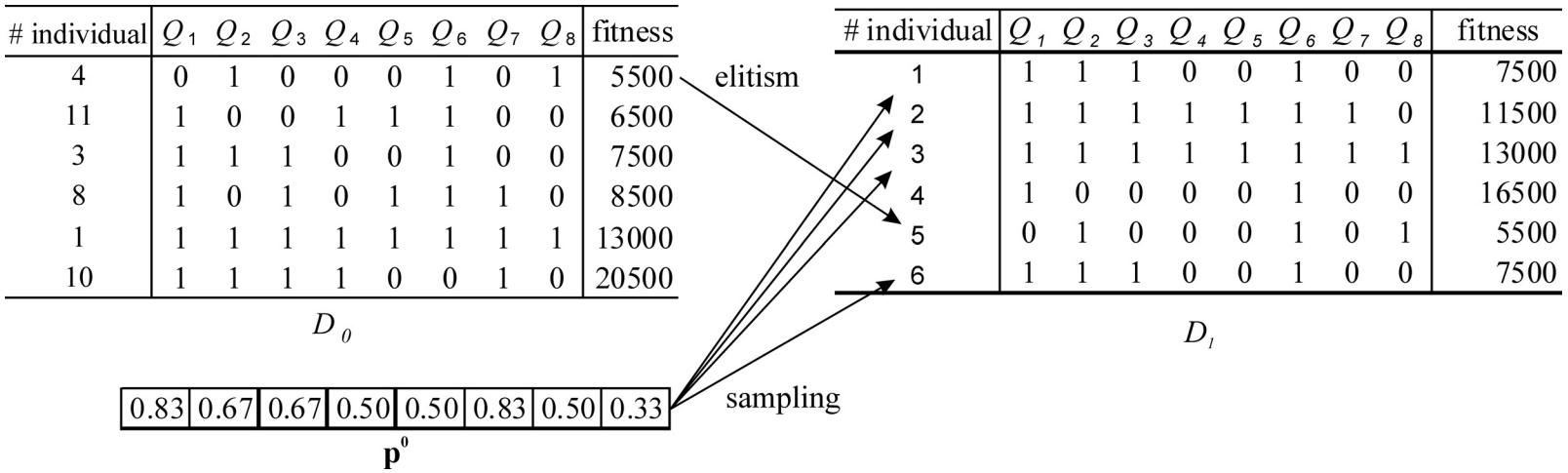
A random sample of Q_i can be generated according to the following scheme

$$Q_i = \begin{cases} 0 & \text{if } U \leq 1 - p_i^{j-1} \\ 1 & \text{if } 1 - p_i^{j-1} < U \leq 1 \end{cases}$$

The D_j population is made up of $(M-1)$ samples of vector $\mathbf{p}^{j-1}$ and the best individual of the $(j-1)$th population. In this way a mechanism of elitism is applied to avoid losing good solutions. In Fig. 6, the generation of population D_1 by sampling the initial probability vector $\mathbf{p}^0$ corresponding to Example 1 is shown.

3.2.4. Local search

The benefits associated with the implementation of a local search procedure strongly depend on the quality of the initial solution and the computational cost of the technique. Therefore, the selection of which and how many of the solutions managed by PBIL will become starting solutions for the local search, and the



# individual	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	fitness
4	0	1	0	0	0	1	0	1	5500
11	1	0	0	1	1	1	0	0	6500
3	1	1	1	0	0	1	0	0	7500
8	1	0	1	0	1	1	1	0	8500
1	1	1	1	1	1	1	1	1	13000
10	1	1	1	1	0	0	1	0	20500

# individual	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	fitness
1	1	1	1	0	0	1	0	0	7500
2	1	1	1	1	1	1	1	0	11500
3	1	1	1	1	1	1	1	1	13000
4	1	0	0	0	0	1	0	0	16500
5	0	1	0	0	0	1	0	1	5500
6	1	1	1	0	0	1	0	0	7500

0.83	0.67	0.67	0.50	0.50	0.83	0.50	0.33

Fig. 6. Generation of D_1 from $\mathbf{p}^0$ vector.

```
Given: initial solution q and best so far q*
Set N1(q) = {∅}
for i =1 to length(q)
    q'=q
    if q'(i)=1
        q'(i)=0
        if movement(i) is tabu
            if q' ≺ q*
                add q' to N1(q)
            end
        else
            add q' to N1(q)
        end
    end
end
return N1(q)
```

Fig. 7. Pseudocode for neighborhood $N_1(\mathbf{q})$ generation.

frequency of calls to SOTS by the main algorithm are important issues to be solved.

Regarding the selection of individuals that undergo local search (candidate solutions), the central idea is to inspect by means of SOTS different regions within each subpopulation. Hence the following steps are executed for each one: (a) identification of its best current solution; (b) grouping its members into *nc* clusters using the Hamming distance as structural similarity metric and the algorithm K-means; (c) selection of its best current individual as a candidate solution; (d) random selection of one candidate solution for each cluster except for the one that contains the best individual.

Given a candidate solution $\mathbf{q}$, N_1 or N_2 neighborhoods are built depending on the oscillatory behavior is associated to a destructive or constructive phase, respectively. In the first case, $N_1(\mathbf{q})$ is defined as the set of solutions obtained by eliminating from $\mathbf{q}$ one measurement when this move is not tabu. The Recency based Tabu list is formed by an n dimensional vector $\mathbf{t}$. A non zero element of $\mathbf{t}$ indicates that this variable move is forbidden because it was performed to obtain a recent solution. Furthermore its value is the number of remaining iterations until *pt* for this move is elapsed. If the best neighbor is in a tabu area but has a better evaluation function value than $\mathbf{q}^*$ then its tabu property is overridden. Fig. 7 shows the pseudocode used to build $N_1(\mathbf{q})$.

During the destructive phase, the search crosses the *FB* and proceeds in the infeasible region until H reaches the bound $L = \sum_{i=1}^{n} c_i + \max(\mathbf{c})$. Thus the depth of the exploration in that region is equal to the cost of the most expensive instrument. Then the constructive phase is initiated by incorporating measurements. In contrast to the previous phase, $N_2(\mathbf{q})$ is defined as the set of solutions obtained by adding one measurement to $\mathbf{q}$ when this move is not tabu. When the search returns to the feasible region the constructive phase finishes.

The Frequency based Tabu list is represented by an n dimensional vector $\mathbf{h}$. The ith component of $\mathbf{h}$ reports the number of moves of variable i used to generate the next solution during *ph* iterations. The evaluation function corresponding to the ith allowable move is penalized in proportion to h_i in order to direct the search to unvisited areas or regions visited less frequently. After *ph* iteration vector $\mathbf{h}$ is reset.

In Fig. 8 the pseudocode of the local search procedure implemented for each candidate solution is presented.

With respect to the number of calls to SOTS, it is well known that an appropriate balance between the solution-space explorative capacity of the algorithm and the exploitation of the more promising regions of that space is necessary. An underlying assumption of metaheuristic procedures is that good solutions have similar structures. Therefore in the extreme case that all subpopulation members undergo local search, similar genotypic solutions are obtained inducing a premature convergence of PBIL to a local optima. In contrast, if the number of calls to SOTS is very low, no advantage would be taken of the enormous exploitative capacity that it exhibits.

Furthermore, it should be noted that there exists a linear relationship between the computational cost of the calls to SOTS by PBIL and the size of the problem. The cardinality of the neighborhood structure of a solution is of order n, both in the constructive as well as in the destructive phases, and each element of this neighborhood is evaluated *maxiter* number of times, therefore each execution of SOTS consumes a number of function evaluations of order (*maxiter***n*). The computation time associated with this task can be excessive for large scale problems, consequently the frequency of calls to the local search procedure should be controlled.

In the proposed algorithm, the frequency of application of the local search technique is governed by a parameter *pso*, which defines the probability of occurrence of that event for a subpopulation. Once it occurs, this subpopulation is marked to inhibit the call to SOTS during *nso* iterations.

Example 2. In this example the design of instrumentation for a plant with 17 streams and 6 units is addressed. The set of key variables is $S_E = \{z_1, z_2, z_6\}$. Fig. 9 outlines the application of the local search procedure to a subpopulation, with $M = 12$ and $nc = 2$, at a certain iteration. The individual #7, which belongs to the first cluster ($ic = 1$), is the best solution of the current subpopulation, therefore it is selected as the candidate solution for that cluster. Regarding the second cluster, a random selection is performed to determine its candidate solution among its members. This corresponds to individual #5. Both members of the subpopulation undergo the SOTS procedure, which produces a remarkable improvement in the solution. In Fig. 10 the path followed by the solutions corresponding to the constructive and destructive phases of the SOTS algorithm is represented in terms of H values for *maxiter* = 120. Fig. 11 shows the path of feasible and infeasible solutions.

3.3. Pseudocode

The pseudocode of the proposed algorithm is presented in Fig. 12. Each subpopulation is characterized by the following variables: $\mathbf{p}$ is the probability vector, $\mathbf{D}$ stands for the matrix

```
Given: initial solution q0
Set L = Σ_i^n c_i + max(c) and FB = Σ_i^n c_i
if q0 is feasible
    Set Phase=1 /* destructive phase */
else
    Set Phase=0 /* constructive phase */
end
set q*=q= q0
for i =1 to # maxiter do
    if Phase =1
        Generate neighbourhood N1(q) and evaluate
        Get best neighbour q'
        if H(q') > L
            Phase = 0
        end
    else
        Generate neighbourhood N2(q) and evaluate
        Get best neighbour q'
        if H(q') ≤ FB
            Phase=1
        end
    end
    q=q'
    Update Recency and Frequency Tabu lists
    if q ≺ q*
        q*=q
    end
end
return q*
```

Fig. 8. Pseudocode of SOTS algorithm for solving the SNDP.

of solutions, **fit** is the vector that contains *H* values for each solution in **D**, **B** is the best solution in **D, I_list** is an *NPBIL* dimensional vector whose elements record the number of remaining iterations until the application of SOTS becomes again allowable for a given subpopulation.

4. Application examples

In this section application results of the proposed methodology to systems of incremental size are reported. The analysis of the results takes into account solution quality and variability, the

# individual	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}	Q_{16}	Q_{17}	fitness	*ic*
1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2870.60	2
2	1	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	2790.60	2
3	1	1	0	1	0	1	0	0	0	1	1	1	1	1	1	1	1	813.80	1
4	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	2195.70	2
5	1	1	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	2732.10	2
6	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1000.10	2
7	1	1	1	0	1	1	0	0	0	0	0	1	1	1	1	1	1	799.10	1
8	1	1	0	1	0	1	0	0	0	0	1	0	0	1	1	1	1	2335.60	1
9	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	2333.60	1
10	1	0	0	1	1	1	0	0	0	1	1	0	1	1	1	1	1	2589.60	1
11	0	1	0	1	0	1	0	0	0	0	1	0	1	1	1	1	1	2462.30	1
12	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	2745.70	2

clustering

# individual	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}	Q_{16}	Q_{17}	fitness	*ic*
3	1	1	0	1	0	1	0	0	0	1	1	1	1	1	1	1	1	813.80	1
7	1	1	1	0	1	1	0	0	0	0	0	1	1	1	1	1	1	799.10	1
8	1	1	0	1	0	1	0	0	0	0	1	0	0	1	1	1	1	2335.60	1
9	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	2333.60	1
10	1	0	0	1	1	1	0	0	0	1	1	0	1	1	1	1	1	2589.60	1
11	0	1	0	1	0	1	0	0	0	0	1	0	1	1	1	1	1	2462.30	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2870.60	2
2	1	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	2790.60	2
4	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	2195.70	2
5	1	1	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	2732.10	2
6	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1000.10	2
12	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	2745.70	2

TS-SO

TS-SO

# individual	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}	Q_{16}	Q_{17}	fitness	*ic*
3	1	1	0	1	0	1	0	0	0	1	1	1	1	1	1	1	1	813.8	1
7	1	1	0	0	0	1	1	1	1	1	0	0	1	0	0	0	0	534.76	1
8	1	1	0	1	0	1	0	0	0	0	1	0	0	1	1	1	1	2335.6	1
9	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	2333.6	1
10	1	0	0	1	1	1	0	0	0	1	1	0	1	1	1	1	1	2589.6	1
11	0	1	0	1	0	1	0	0	0	0	1	0	1	1	1	1	1	2462.3	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2870.6	2
2	1	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	2790.6	2
4	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	2195.7	2
5	1	1	0	0	0	1	1	1	1	1	0	0	1	0	0	0	0	534.76	2
6	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1000.1	2
12	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	2745.7	2

Fig. 9. Clustering and improvement using SOTS.

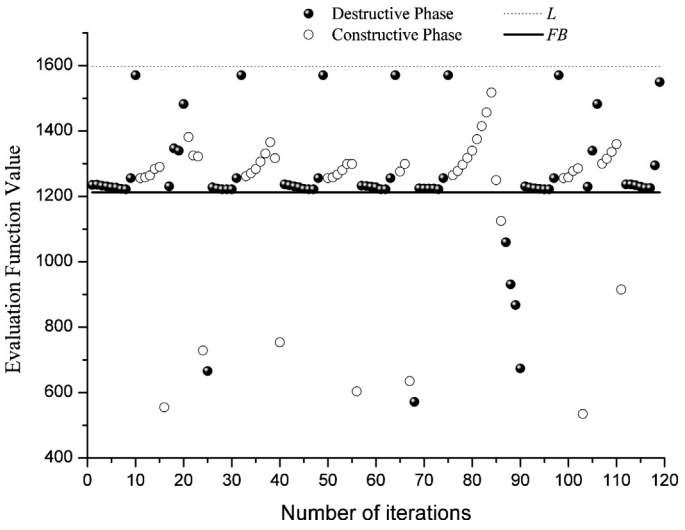


Fig. 10. Solution path followed by the SOTS algorithm in terms of *H*.

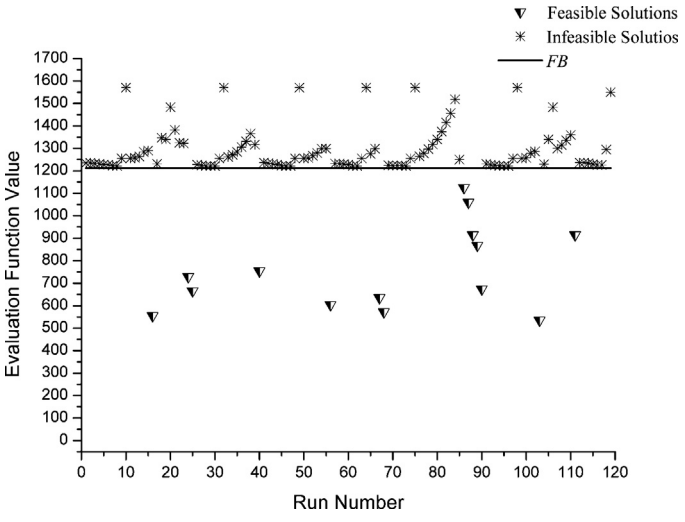


Fig. 11. Feasible and infeasible solution path.

procedure robustness and its scalability behavior to tackle the SNDP for large scale plants.

Three process flowsheets (Cases 1–3) of dimension: 11 units – 28 streams, 19 units – 52 streams, and 47 units – 82 streams are considered. It is assumed that variables are related only by mass balance equations. The standard deviation of flowmeters is 2.5%, 2% and 2% of the corresponding true flowrates for Cases 1–3, respectively. Table 1 shows the complexity of the set of constraints imposed on each Case, and Table 2 reports the parameter settings.

The next table presents the best solutions of Cases 1–3 for 100 runs of the algorithm. The procedure was executed using a Processor Intel® Core (TM) i7 CPU 920 @ 2.67 GHz, 6 GB RAM, using

Table 1
Set of constraints for Cases 1–3.

Case	Constraints
1	$E \geq 1$ for streams 3 8 9 17 20 21 24 25 $\sigma_3^* = 2.23$ $\sigma_8^* = 3.28$ $\sigma_{21}^* = 1.74$ $\sigma_{24}^* = 0.93$
2	$E \geq 1$ for streams 2 5 15 29 31 32 38 39 40 44–52 $\sigma_{15}^* = 12,410$ $\sigma_{31}^* = 13,750$ $\sigma_{32}^* = 126$ $\sigma_{40}^* = 1378$ $\sigma_{44}^* = 568$ $\sigma_{45}^* = 595$ $\sigma_{46}^* = 716$ $\sigma_{47}^* = 546$ $\sigma_{49}^* = 1442$
3	$E \geq 1$ for streams 4 11 21 24 30 32 35 39 43 44 52 58 60 68 75 80 81 $\sigma_{11}^* = 1423$, $\sigma_{35}^* = 172$, $\sigma_{39}^* = 1422$, $\sigma_{58}^* = 27$, $\sigma_{60}^* = 579$, $\sigma_{81}^* = 425$

Table 2
Parameter setting.

Parameter	Value	Parameter	Value
# PBIL	4	*pso*	0.05
M	12	*nso*	25
# MaxGeneration	200	*nc*	2
LR	0.1	*# maxiter*	120
PMUTA	0.02	*pt*	$1.5\sqrt{n}$
MS	0.05	*ph*	$n/2$
Pinteraction	0.25		

MatLab Release 14. As parallel computers are unavailable, the parallel implementation of the algorithm is simulated by sequentially running *NPBIL* instances of the procedure PBIL-SOTS and updating their probability vectors using uniform crossover. Therefore the computation time of each Case is only reported for illustrative and comparative purposes (Table 3).

In Case 1 the best solution is obtained for 100% of the runs, and the average and minimum execution times are 3.26 and 2.50 min, respectively. Figs. 13 and 14 show the behavior of the proposed algorithm for this case study. It is also solved using PBIL with only one population, made up of 48 individuals, and no local search. In Fig. 15, *H* values attained using PBIL are reported. Although the best value of *H* is identical to the one achieved by pPBIL-SOTS (1297.39), the average of *H* values is 1424.34, and their standard deviation is 113.62. These indicate a poorer behavior of PBIL with respect to pPBIL-SOTS.

In Case 2 the best solution is also obtained for 100% of the runs, and the average time is 10.15 min. Figs. 16 and 17 represent the performance of the proposed algorithm for that case study. In Fig. 18, *H* values provided by PBIL for the same case study are displayed. The average of those values is 1408.28 and their standard deviation is 381.17, which are significantly greater than the ones corresponding to pPBIL-SOTS.

In Case 3 the average of *H* values obtained using pPBIL-SOTS is 107,377.32 and their standard deviation is very low, 0.06. The average run time is 23.68 min. Figs. 19 and 20 show the behavior of pPBIL-SOTS for solving that case. The *H* values obtained when only PBIL metaheuristic is used for the same purpose are contained in Fig. 21. The average of those values is 128,824.61 and their standard deviation is 18,919.26. The analysis of these results clearly indicates PBIL no adequately exploits the promising areas of the search space when the size of the problem increases.

Table 3
Results.

Case	Optimal solution	Estimated deviation	Cost
1	1, 2, 4, 7, 9–11, 13–24	$\hat{\sigma}_3 = 2.16$, $\hat{\sigma}_8 = 2.54$, $\hat{\sigma}_{21} = 1.5$, $\hat{\sigma}_{24} = 0.92$	1297.39
2	10, 16, 31–33, 35, 37, 39–41, 43–48, 50–52	$\hat{\sigma}_{15} = 2510$, $\hat{\sigma}_{31} = 916$, $\hat{\sigma}_{32} = 84$, $\hat{\sigma}_{40} = 919$, $\hat{\sigma}_{44} = 379$, $\hat{\sigma}_{45} = 397$, $\hat{\sigma}_{46} = 478$, $\hat{\sigma}_{47} = 364$, $\hat{\sigma}_{49} = 566$	1154.34
3	1, 2, 5, 15, 21, 22, 25, 28, 31, 33–35, 37 44–46, 49–51, 53–55, 60–63, 67, 68, 72–77, 79, 80–82	$\hat{\sigma}_{11} = 698$, $\hat{\sigma}_{35} = 172$, $\hat{\sigma}_{39} = 978$, $\hat{\sigma}_{58} = 22$, $\hat{\sigma}_{60} = 443$, $\hat{\sigma}_{81} = 419$	107,377.13

```
/* Initiate PBILs */
I_list=0          /*Initiate Inhibit list */
for k=1 to NPBIL
        PP=generator1(R, rows_PP)
        D0 (1..M/2)=Best_Solutions(PP,M/2)
        D0 (M/2+1..M)=Far_Solutions(PP,M/2)
        Dk= D0
        for i=1 to n
                pk(i) = sum(Dk(m,i))/ M   m=1..M
        end
end

for j=1 to MaxGenerations
        for k=1 to NPBIL
                Dk =sample(pk)
                for m=1 to M
                        fitk=H(rowm(Dk))
                end
                Bk=best individual in Dk
                if PBILk is not inhibit and randnumber< pso
                        Divide Dk in nc clusters
                        set clbest=cluster which belong Bk
                        Apply SOTS to Bk
                        for w=1 to nc
                                if w<>clbest
                                        Select randomly a solution qw
                                        Apply SOTS to qw
                                end
                        end
                        update Dk
                        update Bk
                        I_list(k)= nso     /* inhibit SOTS on PBILk for nso iterations */
                end

                Update probability vector pk with learning rate LR
                Mutate pk using a probability of mutation PMUTA and a quantity of mutation
                        MS
                Update I_list
        end
        if randnumber<Pinteraction
                Modify pk vectors by crossover operator
        end
end
sol_out=best(B)
```

Fig. 12. Pseudocode of pPBIL-SOTS algorithm.

The following comments arise from the analysis of the previous results:

(1) Good quality and reproducibility of solutions is achieved using the proposed methodology. Better solutions are obtained with respect to those provided by PBIL. Furthermore, given the probabilistic nature of pPBIL-SOTS, the null or extremely low dispersion of the set of possible solutions indicates an appropriate behavior of the procedure for solving the combinatorial problem.
(2) The procedure allows solving SNDPs of incremental size. When Case 3 which involves 82 binary variables is addressed, the algorithm also provides good solutions, and the dispersion of the solution set evaluated in terms of the variation coefficient is only 5.8e−004.
(3) The use of a local search procedure based on SOTS and various instances of PBIL significantly enhance the solution strategy performance with respect to the basic PBIL.
(4) A direct relationship obviously exists between the computation time and the problem size. As mentioned before, the computation times are only reported for comparative purposes, since the procedure is executed by an interpreter program such as MatLab and, the parallelization capability of the procedure has not been properly exploited because PBIL instances are run sequentially. Based on current knowledge, the implementation of the procedure using parallel computers and a compiled language will significantly reduce those computation times. It is expected that they will be in the range of seconds.
(5) Even though application examples show the solution of SNDPs represented by Eq. (3), the proposed algorithm is sufficiently

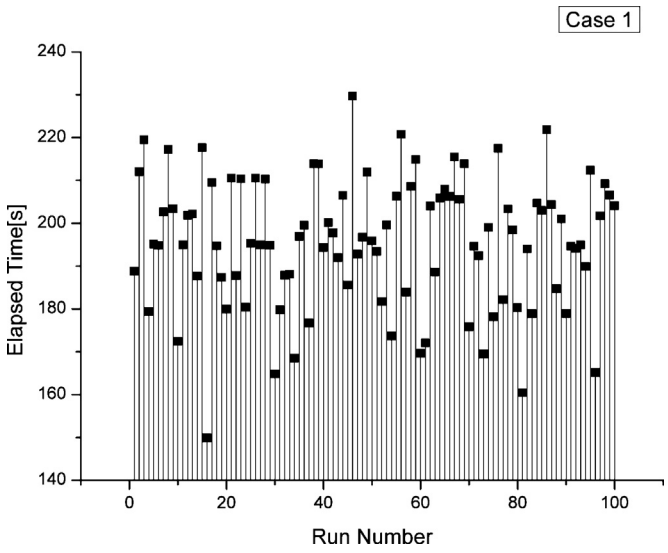


Fig. 13. Running time of pPBIL-SOTS for Case 1.

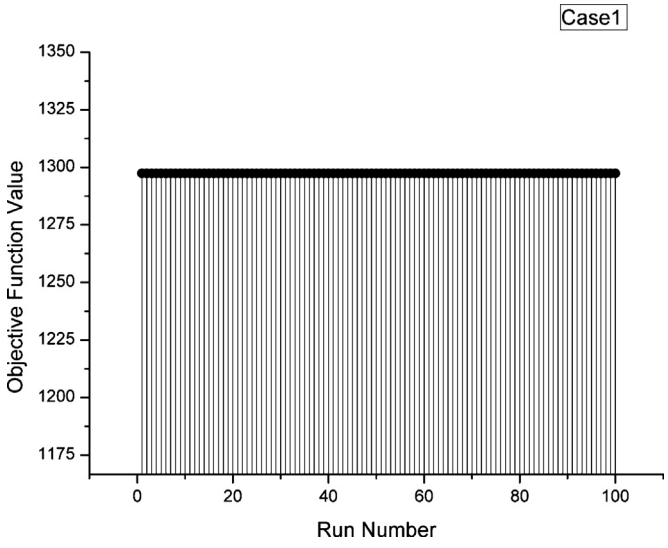


Fig. 14. Objective function value of pPBIL-SOTS for Case 1.

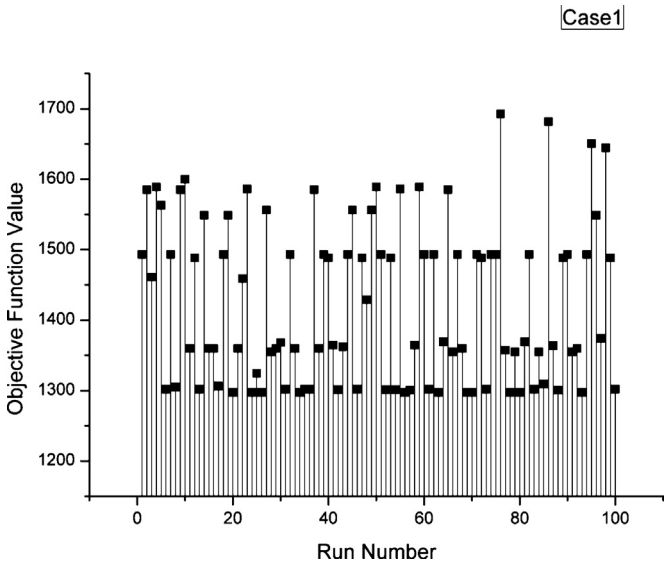


Fig. 15. Objective function value of PBIL for Case 1.

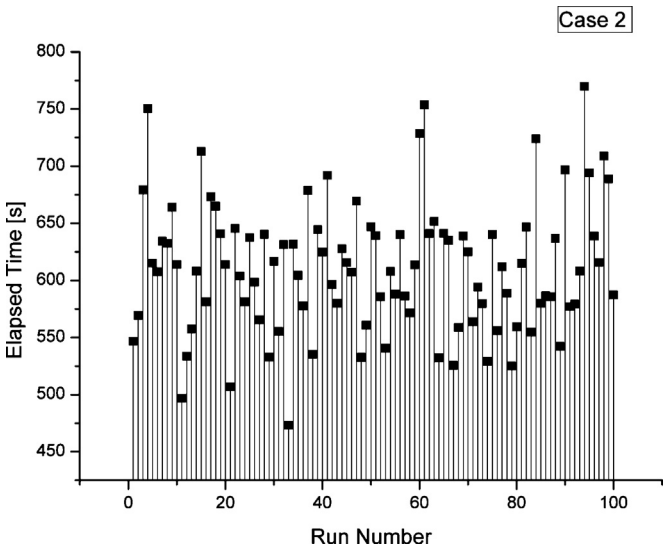


Fig. 16. Running time of pPBIL-SOTS for Case 2.

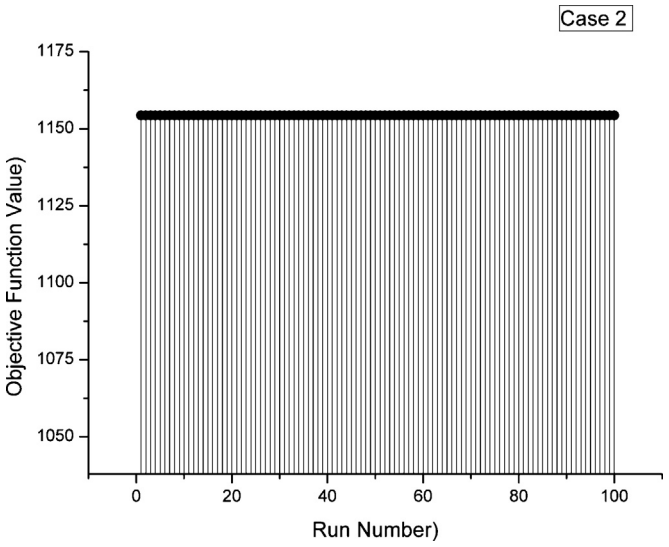


Fig. 17. Objective function value of pPBIL-SOTS for Case 2.

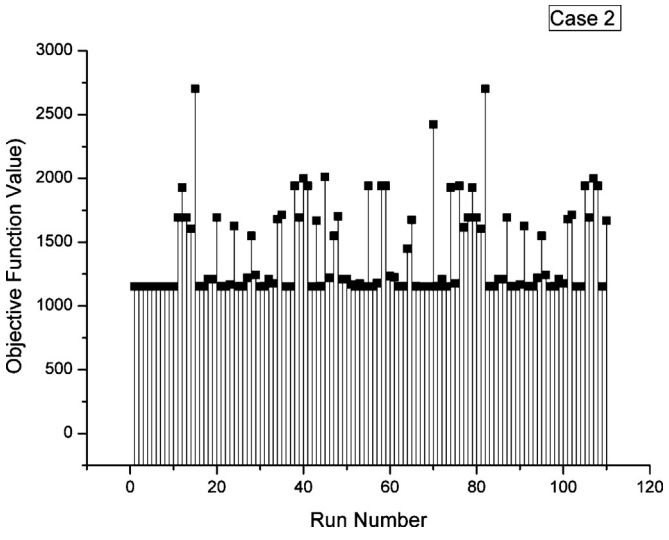


Fig. 18. Objective function value of PBIL for Case 2.

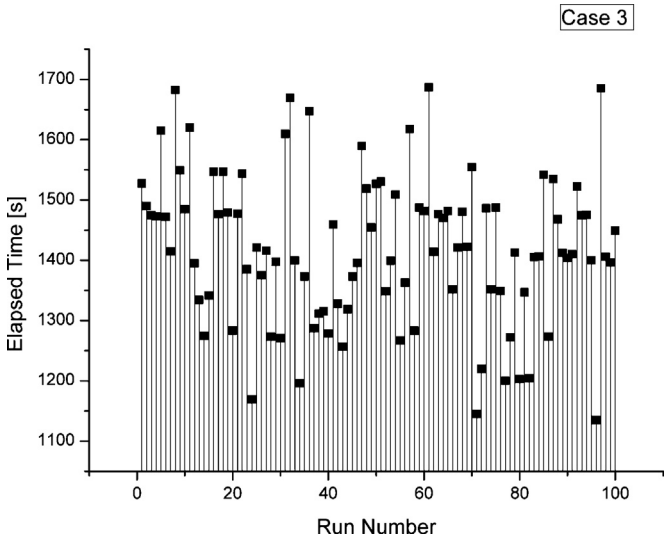


Fig. 19. Running time of pPBIL-SOTS for Case 3.

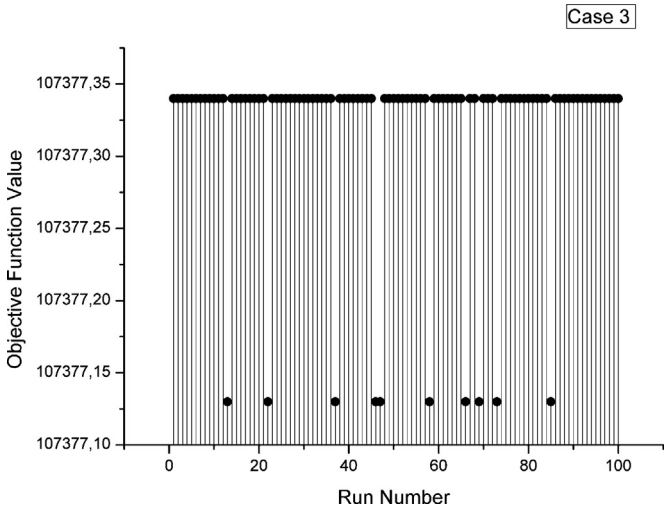


Fig. 20. Objective function value of pPBIL-SOTS for Case 3.

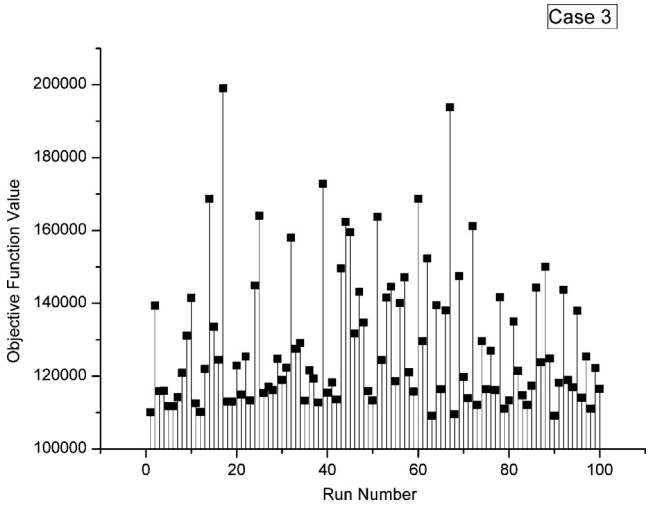


Fig. 21. Objective function value of PBIL for Case 3.

robust to address optimal designs formulated in terms of others objective functions and constraints.

5. Conclusion and future works

In this work a metaheuristic approach based on EDAs and SOTS is presented for the DSNs. Even though diverse combinatorial optimization problems have been addressed using EDAs, just a few ones correspond to the interest of chemical engineers. In this sense, this work provides a new strategy to make decisions associated with the structure of a process system, such as the SN, and an analysis about the performance of the metaheuristics involved in the solution scheme.

The combination of EDAs and SOTS advantages has a synergistic effect on the solution of the SNDP: Thus the proposed method can tackle designs with a high number of binary variables, obtains good quality and reproducibility of the solutions, and is sufficiently robust to address different formulations of the combinatorial optimization problem. The parallel implementation of the algorithm using various subpopulations that evolve independently and interchange information by a crossover mechanism originates a very important reduction of the computation time.

Another issues associated with algorithm modifications to reduce the amount of parameters, alternative schemes to assign the frequency of calls to the local search procedure, and to characterize subpopulations will be addressed in future works. They will also be extended to the utilization of probabilistic models that capture variable interdependencies, such as, the marginal product factorization model.

Given the intrinsic parallelizable nature of the procedure, the parallel implementation of the proposed algorithm using a cluster of computers, which is under construction, will significantly reduce the computation time. It is estimated that the communication time between the computers of the cluster will be low. Furthermore, the outline of future works includes the development of a parallel version of the strategy using a Compute Unified Device Architecture and the extension of the algorithm to address nonlinear process models.

Acknowledgments

The authors wish to thank the financial support of CONICET (National Research Council of Argentina), ANPCyT (National Agency for the Science and Technological Promotion, Argentina), UNRC (Universidad Nacional de Río Cuarto, Río Cuarto, Argentina) and UNS (Universidad Nacional del Sur, Bahía Blanca, Argentina).

References

Bagajewicz, M. (1997). Design and retrofit of sensor networks in process plants. *AIChE Journal*, *43*, 2300–2306.

Bagajewicz, M., & Cabrera, E. (2002). New MILP formulation for instrumentation network design and upgrade. *AIChE Journal*, *48*, 2271–2322.

Bagajewicz, M., Fuxman, A., & Uribe, A. (2004). Instrumentation network design and upgrade for process monitoring and fault detection. *AIChE Journal*, *50*, 1870–1880.

Bagajewicz, M., & Sánchez, M. (1999). Cost optimal design and upgrade of non-redundant and redundant linear sensor networks. *AIChE Journal*, *45*, 1927–1938.

Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning*. Technical Report CMU-CS-94-163. Carnegie Mellon University.

Baluja, S., & Caruana, R. (1995). *Removing the genetics from the standard genetic algorithm*. Technical Report CMU-CS-95-141. Carnegie Mellon University.

Benqlilou, C., Graells, M., Musulin, E., & Puigjaner, L. (2004). Design and retrofit of reliable sensor networks. *Industrial and Engineering Chemistry Research*, *43*, 8026–8036.

Bhushan, M., Narashimhan, S., & Rengaswamy, R. (2008). Robust sensor network design for fault diagnosis. *Computers & Chemical Engineering*, *32*, 1067–1084.

Bhushan, M., & Rengaswamy, R. (2000). Design of sensor network based on the signed directed graphs of the process for efficient fault diagnosis. *Industrial and Engineering Chemistry Research*, *39*, 999–1019.

Bhushan, M., & Rengaswamy, R. (2002a). Comprehensive design of sensor network for chemical plants based on various diagnosability and reliability criteria. I. Framework. *Industrial and Engineering Chemistry Research*, *41*, 1826–1839.

Bhushan, M., & Rengaswamy, R. (2002b). Comprehensive design of sensor network for chemical plants based on various diagnosability and reliability criteria. II. Applications. *Industrial and Engineering Chemistry Research*, *41*, 1840–1860.

Carnero, M., Hernández, J., & Sánchez, M. (2005). Optimal sensor network design and upgrade using Tabu Search. *Computer Aided Chemical Engineering*, *20*, 1447–1452.

Carnero, M., Hernández, J., & Sánchez, M. (2009). Design of sensor networks for chemical plants based on metaheuristics. *Algorithms*, *2*, 259–281.

Carnero, M., Hernández, J., Sánchez, M., & Bandoni, A. (2001). An evolutionary approach for the design of non-redundant sensor networks. *Industrial and Engineering Chemistry Research*, *40*, 5578–5584.

Carnero, M., Hernández, J., Sánchez, M., & Bandoni, A. (2005). On the solution of the instrumentation selection problem. *Industrial and Engineering Chemistry Research*, *44*, 358–367.

Cavin, L., Fischer, U., Glover, F., & Hungerbuhler, K. (2004). Multiobjective process design in multi-purpose batch plants using a Tabu Search optimization algorithm. *Computers and Chemical Engineering*, *28*, 459–478.

Chmielewski, D., Palmer, T. E., & Manousiouthakis, V. (2002). On the theory of optimal sensor placement. *AIChE Journal*, *48*, 1001–1012.

Emmendorfer, L., & Ramirez Pozo, A. (2009). Effective linkage learning using low-order statistics and clustering. *IEEE Transactions on Evolutionary Computation*, *13*, 1233–1246.

Gala, M., & Bagajewicz, M. (2006). Efficient procedure for the design and upgrade of sensor networks using cutsets and rigorous decomposition. *Industrial and Engineering Chemistry Research*, *45*, 6687–6697.

Gerkens, C., & Heyen, G. (2005). Use of parallel computers in rational design of redundant sensor networks. *Computers and Chemical Engineering*, *29*, 1379–1387.

Gerkens, C., & Heyen, G. (2008). Sensor placement for fault detection and localization. *Computer Aided Chemical Engineering*, *25*, 355–360.

Glover, F., & Laguna, M. (1997). *Tabu Search* (1st ed.). Norwell: Kluwer Academic Publishers.

Hauschild, M., & Pelikan, M. (2011). An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, *1*, 111–128.

Kelly, J. D., & Zyngier, D. (2008). A new and improved MILP formulation to optimize observability, redundancy and precision for sensor network problems. *AIChE Journal*, *54*, 1282–1292.

Kotecha, P. R., Bhushan, M., & Gudi, R. D. (2007). Constrained programming based robust sensor network design. *Industrial and Engineering Chemistry Research*, *46*, 5985–5999.

Lin, B., & Miller, D. C. (2004a). Solving heat exchanger network synthesis problems with Tabu Search. *Computers and Chemical Engineering*, *28*, 1451–1464.

Lin, B., & Miller, D. C. (2004b). Tabu Search algorithm for chemical process optimization. *Computers and Chemical Engineering*, *28*, 2287–2306.

Nguyen, D., & Bagajewicz, M. (2008). Design of nonlinear sensor networks for process plants. *Industrial and Engineering Chemistry Research*, *47*, 5529–5542.

Nguyen, D., & Bagajewicz, M. (2011). New efficient breadth_first/level traversal tree search method for the design and upgrade of sensor networks. *AIChE Journal*, *57*, 1302–1309.

Pang, H., Hu, K., & Hing, Z. (2006). Adaptive PBIL algorithm and its application to solve scheduling problems. In *IEEE conference on computer aided control system design* Munich, Germany, (pp. 784–789).

Sen, S., Narasimhan, S., & Deb, K. (1998). Sensor network design of linear processes using genetics algorithms. *Computers and Chemical Engineering*, *22*, 385–390.

Viswanath, A., & Narasimhan, S. (2001). Multi-objective sensor network design using genetic algorithms. In *4th IFAC workshop on on-line fault detection and supervision in the chemical process industries* Seoul, Korea, (pp. 265–270).

Wan, S., & Qiu, D. (2008). Vehicle routing optimization problem with time constraint using advanced PBIL algorithm. In *IEEE international conference on service operations and logistics, and informatics, Vol. 1* Beijing, China, (pp. 1394–1398).