STOCHASTIC PROCEDURES FOR THE OPTIMAL SENSOR LOCATION IN CHEMICAL PLANTS

Mercedes C. Carnero\textsuperscript{a}, José L. Hernández\textsuperscript{a} and Mabel C. Sánchez\textsuperscript{b}

\textsuperscript{a}Facultad de Ingeniería, Universidad Nacional de Río Cuarto, Río Cuarto, Argentina
\textsuperscript{b}Planta Piloto de Ingeniería Química (UNS-CONICET), Bahía Blanca, Argentina

Introduction

Process information is the foundation upon which monitoring, control, optimization, planning and scheduling, fault diagnosis, etc. are based. To satisfy information requirements, regarding its quality and availability, it is essential to locate an appropriate sensor network (SN) in the plant.

The SN designer should decide to measure each process variable or not. These decisions are mathematically formulated in terms of binary variables. A combinatorial optimization problem results that is usually multimodal and involves many binary variables. Its solution has been addressed using tree search algorithms, MILP techniques and stochastic procedures.

Regarding the stochastic solution schemes, techniques based on Genetic Algorithms (GAs) were proposed initially. Benqlilou et al. (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 classic GA to reduce the solution time. A hybrid procedure was developed by Carnero et al. (2005a) to minimize the instrumentation cost subject to precision constraints on key variables, which makes use of a structured population in the form of neighborhoods and a local optimizer of the best current solutions. Furthermore, Carnero et al. (2005b) developed a new design strategy within the framework of Tabu Search (TS), which uses a Strategic Oscillation Technique around the feasibility boundary (SOTS).

Recently a metaheuristic approach based on the Estimation of Distribution Algorithms (EDAs) and SOTS was presented (Carnero et al., 2013). Application results of that procedure demonstrated that the combination of EDAs and SOTS advantages has a synergistic effect on the solution of the SN design problem (SNDP). The proposed solution scheme makes use of the Population Based Incremental Learning Algorithm (PBIL) developed by Baluja (1994), which assumes independent relationships among variables.

According to the model complexity, EDAs can be broadly divided into univariate, bivariate or multivariate approaches (Hauschild and Pelikan, 2011). To handle variable interdependencies, the second and third classes of EDAs require complex learning algorithms and significant additional computational resources. In this work, the utilization of probabilistic models that capture variable interdependencies, such as the marginal product factorization model (Santana et al., 2010), to solve the SNDP is analysed. A comparative performance study is conducted to evaluate the benefits of increasing the complexity of the distribution model.
Estimation of Distribution Algorithms

Evolutionary algorithms based on probabilistic models are recognized as a new computing paradigm in evolutionary computation. Different from GA, traditional crossover or mutation operations are not performed on the individuals in the population, but they adopt an evolutionary mode for searching the best solutions. This evolutionary mode first builds a probabilistic model about the distribution of good individuals in the search space and then samples a new generation of the population using that model. In this way EDAS succeed to acquire the knowledge for approaching the global optimum in the search space step by step.

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 and multivariate approaches (Hauschild and Pelikan, 2011).

Regarding discrete EDAs, the univariate techniques such as PBIL, cGA (Harik et al., 1999) and UMDA (Mühlenbein, 1998) assume that all variables are independent. Thus their joint probability can be factorized as the product of univariate marginal probabilities. Some algorithms learn the model from a population of solutions, while others update the model using only a few individuals.

The modelling capabilities of EDAs were extended using bivariate models. Efficient learning methods are used to represent pairwise dependencies between variables. Among these procedures the following should be mentioned: COMIT (Baluja and Davies, 1997) and BMDA (Pelikan and Mühlenbein, 1999).

Univariate and bivariate EDAs can be efficiently applied to separable problems or to problems with low degrees of dependency among the variables. But those approaches might rapidly lose their efficiency when the number of interactions among variables is larger. Therefore multivariate EDAs are devised to capture the dependencies among an arbitrary number of variables. Those techniques work by decomposing the joint probability distribution into factors that involve several variables of the problem, but their modelling flexibility comes at the expense of an extra computational work. In this sense, the methodology called EcGA (Harik et al., 2006) factorizes the joint probability distribution into a number of marginal distributions defined over non-overlapping subsets of variables, and the procedure AffEDA (Santana et al., 2010) uses affinity propagation with the same purpose. Furthermore Etxeberria and Larrañaga (1999) proposed the methodology called EBNA to learn a Bayesian network from the selected set of solutions in every generation.

Sensor Network Design Problem

The minimum cost SNDP that satisfies precision and estimability constraints for a set of key variable estimates is stated by Eq. (1), where $q$ is an $n$ dimensional vector of binary variables such that: $q_i = 1$ if variable $i$ is measured, and $q_i = 0$ otherwise, $e^T$ is the cost vector; $\hat{\sigma}_k$ is the estimate standard deviation of the $k$-th variable contained in $S_\sigma$ after a data reconciliation procedure is applied, and $E_l$ stands for the degree of estimability of the $l$-th variable included in $S_E$ (Bagajewicz and Sánchez, 1999). For this formulation $E_l$ is set equal to one, consequently only a variable classification procedure run is
needed to check its feasibility. Furthermore \( S_\sigma \) 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.

\[
\begin{align*}
\text{Min} & \quad e^Tq \\
\text{s.t.} & \quad \hat{\sigma}_k(q) \leq \sigma^*_k(q) \quad \forall k \in S_\sigma \\
& \quad E_l(q) \geq 1 \quad \forall l \in S_E \\
& \quad q \in \{0,1\}^n
\end{align*}
\]

(1)

**PBIL-SO Based Sensor Network Design Approach**

In this section an outline of the new metaheuristic procedure developed by Carnero et al. (2013) for the design of SNs (DSNs) is presented. The PBIL is the core of the solution strategy.

Regarding the basic PBIL procedure, initially a probability vector \( p^0 \) of size \( n \) 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 \( p^{j-1} \). Each individual is evaluated using the fitness function, and the best one in terms of fitness, \( B^j \), is determined. Then the probability vector is updated, position by position, using \( B^j \) and the learning rate \( LR \). After that, a mutation of \( p^j \) is performed, position by position, using the mutation probability \( PMUTA \) and the mutation amount \( MS \).

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 using a sophisticated local search algorithm. The procedure 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 and Lagura, 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. 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. Then move rules are modified to drive the search towards the feasible region until certain condition is satisfied. 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.
Comparative Studies between PBIL-SO and multivariate EDAs

In this section application results of the PBIL-SO methodology are compared with those provided by the strategy AffEDA (Santana, et al., 2009) for the DSNs of a process flowsheet made up of 47 units – 82 streams. It is assumed that variables are related only by mass balance equations. The standard deviation of flowmeters is 2.5% of the corresponding true flowrates. The following constraints are imposed to the minimum-cost optimization problem: $E \geq 1$ for streams 5 12 14 35 37 44 62 70 77; $\sigma^*_{10} = 1584.2$, $\sigma^*_{13} = 1359.6$, $\sigma^*_{35} = 200.7$, $\sigma^*_{39} = 1580.6$, $\sigma^*_{56} = 122.72$, $\sigma^*_{69} = 1284.4$.

The procedures were executed using a Processor Intel ® Core (TM) i7 CPU 920 @ 2.67 GHz, 6GB RAM, using MatLab Release 14. As parallel computers are unavailable, the parallel implementation of the PBIL-SO algorithm is simulated by sequentially running NPBIL instances of that procedure and updating their probability vectors using uniform crossover. Regarding parameter settings, Table 1 presents parameter values for the PBIL-SO algorithm.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPBIL</td>
<td>4</td>
<td>$P_{so}$</td>
<td>0.05</td>
</tr>
<tr>
<td>$M$</td>
<td>12</td>
<td>$N_{so}$</td>
<td>25</td>
</tr>
<tr>
<td>$# MaxGeneration$</td>
<td>200</td>
<td>$N_c$</td>
<td>2</td>
</tr>
<tr>
<td>LR</td>
<td>0.1</td>
<td>$# maxiter$</td>
<td>120</td>
</tr>
<tr>
<td>PMUTA</td>
<td>0.02</td>
<td>$P_t$</td>
<td>$1.5\sqrt{n}$</td>
</tr>
<tr>
<td>$MS$</td>
<td>0.05</td>
<td>$P_h$</td>
<td>$n/2$</td>
</tr>
<tr>
<td>$P_{interaction}$</td>
<td>0.25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where $P_{interaction}$ is the Crossover probability, $P_{so}$ stands for the Application probability of SOTS, $N_{so}$ is the Number of iterations between allowable calls for SOTS, $N_c$ represents the Number of clusters, $\# maxiter$ is the Maximum Number of Iterations for SOTS, $P_t$ stands for the Tabu Tenure Period, and $P_h$ is the Number of iterations before the Frequency based Tabu list is reset.

For the AffEDA procedure, different population sizes are analyzed to determine the impact of this parameter on the solution quality ($M=50$, $M=200$ and $M=300$). Furthermore the tournament selection is applied to 50% of population individuals at the learning stage, and the maximum generations number ($\# MaxGeneration$) which is used as stop criterion is equal to 200.

Table 2 shows the statistics of the best solutions for 100 runs of each algorithm. It can be seen that PBIL-SO outperforms AffEDA, even when its population size has been increased to 300 individuals. The minimum value of the best solutions obtained by the PBIL-SO procedure is 50845.16, and the median and the coefficient of variation values are 50845.37 and 1%, respectively. The 99 % of the best solutions are lower than 52909.78 for PBIL-SOTS, but the same measure for AffEDA when $M=300$ is 77391.18. A summary of the evaluation function values of the best solutions is displayed using box-plots in Figure 1, and those values are represented in Figure 2 in ascending order.
Table 2: Best Solutions statistics

<table>
<thead>
<tr>
<th>Method</th>
<th># Runs</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Coefficient of Variation</th>
<th>Minimum</th>
<th>Median</th>
<th>Maximum</th>
<th>P99</th>
</tr>
</thead>
<tbody>
<tr>
<td>AffEDA, M=48i</td>
<td>100</td>
<td>104055.68</td>
<td>7142.19</td>
<td>0.07</td>
<td>87282.49</td>
<td>105128.89</td>
<td>122632.53</td>
<td>121365.27</td>
</tr>
<tr>
<td>AffEDA, M=200i</td>
<td>100</td>
<td>58187.47</td>
<td>8288.00</td>
<td>0.14</td>
<td>50959.56</td>
<td>53901.30</td>
<td>82330.96</td>
<td>80959.77</td>
</tr>
<tr>
<td>AffEDA, M=300i</td>
<td>100</td>
<td>56176.08</td>
<td>7981.59</td>
<td>0.14</td>
<td>50846.18</td>
<td>51657.44</td>
<td>80547.80</td>
<td>77391.18</td>
</tr>
<tr>
<td>PBIL-SOTS, M=12x4i</td>
<td>100</td>
<td>50886.63</td>
<td>412.88</td>
<td>0.01</td>
<td>50845.16</td>
<td>50845.37</td>
<td>54974.18</td>
<td>52909.78</td>
</tr>
</tbody>
</table>

Figure 1: Box-plot of the Evaluation Function values

Figure 3 shows the execution times in ascending order, and Table 3 provides an analysis of those elapsed times. When the population size increases for AffEDA, a proportional increment of the execution times is observed. Furthermore the elapsed times of PBIL-SOTS and AffEDA with M=300 i are similar, which is a sensible result taking into account that the performance of AffEDA for that population size approximates the behaviour of PBIL-SOTS.

Tabla 3: Elapsed time statistics (min)

<table>
<thead>
<tr>
<th>Method</th>
<th># Runs</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Coefficient of Variation</th>
<th>Minimum</th>
<th>Median</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>AffEDA, M=48i</td>
<td>100</td>
<td>5,90274</td>
<td>0,03953</td>
<td>0,0067</td>
<td>5,82433</td>
<td>5,901</td>
<td>6,07017</td>
</tr>
<tr>
<td>AffEDA, M=200i</td>
<td>100</td>
<td>18,98687</td>
<td>0,07294</td>
<td>0,00384</td>
<td>18,83967</td>
<td>18,97925</td>
<td>19,2125</td>
</tr>
<tr>
<td>AffEDA, M=300i</td>
<td>100</td>
<td>29,26885</td>
<td>0,1394</td>
<td>0,00476</td>
<td>28,91983</td>
<td>29,26075</td>
<td>29,69017</td>
</tr>
<tr>
<td>PBIL-SOTS, M=12x4i</td>
<td>100</td>
<td>25,12669</td>
<td>2,65785</td>
<td>0,10578</td>
<td>17,62417</td>
<td>24,89975</td>
<td>30,99017</td>
</tr>
</tbody>
</table>

5
Figure 2: Evaluation function values for the best solutions using AffEDA and PBIL-SOTS

Figure 3: Execution times for AffEDA and PBIL-SOTS
The algorithm EBNA, which learns a Bayesian network from the selected set of solutions, is also used to solve the proposed problem. One hundred runs are performed considering $M=50$ individuals. A better performance of PBIL-SO with respect to EBNA is achieved taking into account the quality of the solution and the execution time. The minimum value of the best solutions obtained using EBNA is 105643.06 and the average of those solutions is 184736.068. Furthermore EBNA’s execution times are significantly higher than PBIL-SO’s elapsed times. Results are included in Figures 4 and 5.

![Figure 4: Evaluation function values for the best solutions using EBNA](image1.png)

![Figure 5: Execution times for EBNA](image2.png)
Conclusions

In this work the performance of the strategy PBIL-SO, which is an univariate EDA enhances using local search, is compared with the behaviour of some multivariate EDAs for the solution of a SNDP. For the analysed case studies, PBIL-SO outperforms the other procedures taking into account the quality of the solution and the elapsed computational time. In future works the analysis will be extended to other process flowsheets and design problems.

References